Computer Graphics International 2004 conference proceedings, IEEE, ISSN 1530-1052, pp. 270-273, 2004

A New Line Clipping Algorithm with Hardware Acceleration

Vaclav Skala[1]
University of West Bohemia, Pilsen
Department of Computer Science and Engineering
Czech Republic
skala@kiv.zcu.cz

Abstract

Algorithms for line clipping against convex polygon have been studied for a long time and many research papers have been published so far. In spite of the latest graphical hardware development and significant increase of performance the clipping is still a bottleneck of the graphical pipeline. This paper presents a new robust and fast algorithm for line clipping by a convex polygon. The algorithm uses a small pre-processing in order to obtain significant speed up. The proposed algorithm is especially convenient for applications where points or lines are represented in homogeneous coordinates. The algorithm does not use division in floating point representation as the resulting points are in homogeneous coordinates. The algorithms will benefit if vector-vector hardware supported operations can be used.

1. Introduction

There are many algorithms devoted to the line and line segment clipping in E2 and E3. Generally algorithms have been developed and modified for line or line segment or polygon clipping against rectangular window, convex polygon or polygonal clipping area [4], [7], [8], [9], [10], [11]. Some modifications have been developed also for self-intersecting clipping polygon or for areas with linear or quadric edges [12].

Also some algorithms were specially developed or tuned for some special cases like small windows or for some specific characteristics of input data set. Some algorithms and their efficiency are very sensitive to input data, i.e. geometrical distribution of lines or line segments, too.

Algorithms used in E2 space are mostly based on Eucledian space representation while homogeneous coordinates in spite of the fact that positive projective space representation is more natural in some cases. In many applications the clipping window or polygon are constant for many clipped lines. In this case a pre-processing might be another factor speeding the algorithm, too.

In spite of the latest graphical hardware development and significant increase of performance the clipping is still a bottleneck of the graphics pipeline.

This paper presents a new fast and robust algorithm for line clipping against convex polygon and line clipping against rectangular window. The proposed algorithm is compared with well-known Cyrus-Beck algorithm [3], [4]. Experimental evaluation, verification and comparison of the proposed algorithm are presented.

2. Projective geometry and duality

Homogeneous coordinates are widely used in computer graphics applications, usually connected with geometric transformations like rotation, scaling, translation and projection, etc. The "geometrical" interpretation is shown at Figure 1.

The point $\boldsymbol{x}$ is defined as a point in E^2 with coordinates (X,Y) or as a point with homogeneous coordinates $[x, y, w]^T$, where $w = 1$ usually. The point $\boldsymbol{x}$ is actually a "line" without the origin in the projective space $P^2(x,y,w)$, and $X = x / w$ and $Y = y / w$. It can be seen that a line $p \in E^2$ is actually a plane ρ without the origin in the projective space P^2, i.e. the line p is defined as

$$ax + by + cw = 0, \quad w \neq 0$$

[1] Supported the Ministry of Education of the Czech Republic-project LA240 and Microsoft Research - project No.2003-178

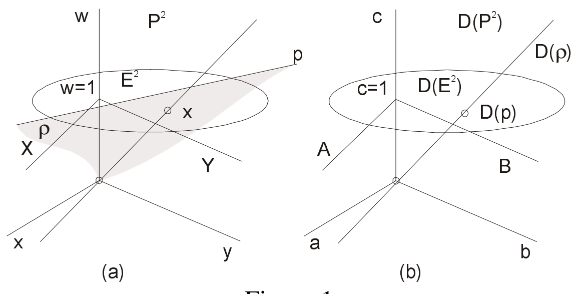

Figure 1

The line *p* is actually geometric interpretation of that. The equation can be divided by any $\xi \neq 0$ without any effect to the geometry. In dual representation the plane ρ can be represented as a line $D(\rho) \in D(P^2)$ or as a point $D(\rho) \in D(E^2)$, when a projection is made, e.g. for $c = 1$. There is a complete theory on projective spaces, see [5], [13] for details.

On the other hand, there is a phenomenon of principle of duality that can be used for derivation of some useful formula. The principle states that any theorem remains true when we interchange the words "point" and line", "lie on" and "pass through", "join" and "intersection", "collinear" and "concurrent" and so on. Once the theorem has been established, the dual theorem is obtained as described above, for details see [Cox61a], [John96a]. In other words, the duality says that in all theorems it is possible to substitute a term "point" by a term "line" and term "line" by the term "point" and the given theorem stays valid. This helps a lot in solution of some geometrical cases.

Definition$_1$

The cross product of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is defined as

$$\mathbf{x}_1 \text{ x } \mathbf{x}_2 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix}$$

where $\mathbf{i} = [1,0,0]^T$, $\mathbf{j} = [0,1,0]^T$, $\mathbf{k} = [0,0,1]^T$

Theorem$_1$

Let two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are given in the projective space. Then a line $\boldsymbol{p}$, that is defined by those two points, is determined as the cross product $\boldsymbol{p} = \boldsymbol{x}_1 \, x \, \boldsymbol{x}_2$

Note: It can be seen that it is valid also for cases if $w \neq 1$ & $w \neq 0$.

Theorem$_2$

Let two lines $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are given in the projective space. Then a point $\boldsymbol{x}$, that is defined as intersection of those two lines, is determined as a cross product $\boldsymbol{x} = \boldsymbol{p}_1 \, x \, \boldsymbol{p}_2$.

These two theorems are very important as they enable us to handle some problems defined in homogeneous coordinates and make computations quite effective.

The Cyrus-Beck algorithm (CB) is the most famous algorithm [3] for line and line segment clipping by convex polygon. There are also several algorithms for line clipping against the given convex polygon, see [1], [8], [9], [12] that claimed some advantages over the CB algorithm. Nevertheless the CB algorithm is very stable and its performance is nearly independent from "geometrical" distribution of clipped primitives. The CB algorithm assumes that the clipping polygon is convex,

It is obvious that the CB algorithm computes *N* intersection points, but only two are actually needed if an intersection exists.

3. The proposed algorithm

A new approach to the line clipping by convex polygon in E^2 based on "function of separation" is presented here. Main advantages of the proposed algorithm are:

- robustness and stability,
- significant speed up,
- it works in homogeneous coordinates natively.

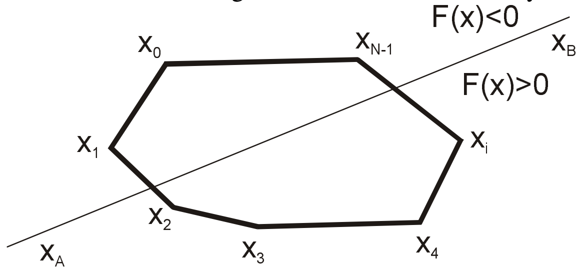

Figure 2

Let us assume a convex polygon *P*, see Figure 2, and a line *p* given as $F(x) = ax + by + c = 0$. The line *p* splits space into two half spaces, i.e. $F(x) < 0$ and $F(x) \geq 0$

It can be seen that the function *F(x)* can be evaluated for each vertex of the given clipping convex polygon and for the i-th vertex the value c_i is obtained as follows:

$$c_i = \begin{cases} 1 & \text{if } F(x_i) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

It means that each vertex is classified whether it is on the "left" or the "right" side of the clipped line *p*.

For simplicity of explanation, let us assume a rectangular window, i.e. a polygon with 4 edges, see Figure 3.

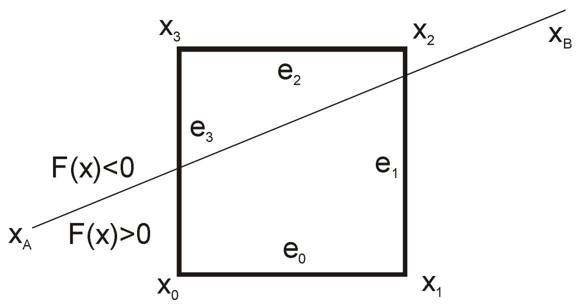

Figure 3

In this case the vector $\boldsymbol{c}$ consists of bits

$$\boldsymbol{c} = [\, c_4 , c_3 , c_2 , c_1 \,]^T$$

Let us construct a table TAB of all possible values of the vector $\boldsymbol{c}$, see Table 1. If all the combinations are interpreted geometrically, it can be seen that for each line of the table the indices of the intersected edges can be pre-computed. It means that the function *F(x)* can be evaluated for each vertex of the clipping polygon and indices of intersected edges are stored in vectors TAB1 and TAB2, see Table 1. Some combinations are impossible, like $[\, 0, 1, 0 , 1 \,]^T$, and these combinations are signed as N/A. Also it can be seen that the vectors TAB1 and TAB2 are "symmetrical", i.e. the **orientation of the clipping polygon *P* is not needed**.

The extension for a general case when the given line *p* is clipped by the convex polygon *P* is straightforward, i.e. the vector $\boldsymbol{c}$ has *N* bits in this case and similar vectors TAB1 and TAB2 can be generated for general case as well. Of course, it is necessary to determine which edges are intersected for each possible combination of the vector $\boldsymbol{c}$.

c	c_3	c_2	c_1	c_0	**TAB1**	**TAB2**	**MASK**
0	0	0	0	0	None	None	None
1	0	0	0	1	0	3	0100
2	0	0	1	0	0	1	0100
3	0	0	1	1	1	3	0010
4	0	1	0	0	1	2	0010
5	0	1	0	1	N/A	N/A	N / A
6	0	1	1	0	0	2	0100
7	0	1	1	1	2	3	1000
8	1	0	0	0	2	3	1000
9	1	0	0	1	0	2	0100
10	1	0	1	0	N/A	N/A	N / A
11	1	0	1	1	1	2	0010
12	1	1	0	0	1	3	0010
13	1	1	0	1	0	1	0100
14	1	1	1	0	0	3	0100
15	1	1	1	1	None	None	None

Table 1

A reader can find out that those indices of intersected edges can be determined easily. When the two following bits in the vector $\boldsymbol{c}$ are different, i.e. $c_i \neq c_{i+1}$ (+ means with **mod** *N*) the edge $\boldsymbol{x}_i\boldsymbol{x}_{i+1}$ will be intersected. Of course there will be more impossible cases, assigned recently as N/A, because in the case of convex clipping polygon the edges are actually split to:

- **one segment** of edges on the "right" side of the line *p*,
- **one segment** of edges on the "left" side of the line *p*,
- edges intersected by the line *p*.

It can be seen that the table construction can be made once for all clipped lines if the clipping polygon *P* is not changed. In case of necessity the vectors TAB1 and TAB2 can be interpreted by **if** statements "on the fly". The Algorithm 1 describes the proposed algorithm.

```
procedure CLIP_L;
{ $x_A$ , $x_B$ – can be in homogeneous coordinates }
{ The EXIT statement ends the procedure }
{ input: $x_A$ , $x_B$ }
begin { $x_A=[x_A,y_A,1]^T$ }
{1}    $p := x_A$ x $x_B$;  { $ax+by+c = 0$; $p = [a,b,c]^T$ }
{2}    for k:=0 to N-1 do { $x_k=[x_k,y_k,1]^T$ }
{3}            if $p^T x_k \geq 0$ then $c_k$:=1 else $c_k$:=0;
{4}    if c = $[0000]^T$ or c = $[1111]^T$ then EXIT;
{5}    i:= TAB1[c];        j:= TAB2[c];
{6}    $x_A := p$ x $e_i$ ;        $x_B := p$ x $e_j$ ;
{7}    DRAW ($x_A$; $x_B$ )
end {CLIP_L};
```

Algorithm 1

It can be seen that the algorithm is very simple. The N/A cases will be never used.

Because all operations in the Algorithm 1 are valid in homogeneous coordinates, this algorithm can be used if input and output points are in homogeneous coordinates. Note that **cross-product processor** can perform steps {1,6} and steps {2,3} can be made in parallel. The cross-product can be replaced by dot product with matrix multiplication

$$\mathbf{x} = \mathbf{x}_1 \, x \, \mathbf{x}_2 , \quad \text{where } \mathbf{x} = [\, x , y , w]^T$$

Then

$$x = \mathbf{x}_1 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \mathbf{x}_2 \qquad y = \mathbf{x}_1 \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}_2$$

$$w = \mathbf{x}_1 \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{x}_2$$

4. Experimental results

Both, the CB and the proposed CLIP_L algorithms were implemented in Pascal (Delphi) and C++. Their efficiency compared on PC 750 MHz. All experiments were made with $w = 1$. Let us define the speed-up as

$$v = \frac{T_{CB}}{T_{CLIP\_L}}$$

where: T_{CB}, resp. T_{CLIP_L} is time spent by CB, resp. by the new proposed CLIP_L algorithm.

The experiments proved that

$$v \geq 1,9$$

and it slightly grows with value N, where N is the number of edges of the given convex clipping polygon. For very high N the value v reached value

$$v = 2,1$$

Several aspects, e.g. caching, additional instructions, cycles etc., cause some differences between theoretical estimation and experimental results.

The main advantages of the proposed CLIP_L algorithm are:

- robustness of the algorithm,
- ability to work with homogeneous representation,
- the vector-vector and parallel processing (the cycle can be made in parallel) speed up the processing significantly,
- simple implementation in hardware.

5. Conclusion

This paper describes a new approach to the line clipping problem using projective space and homogeneous coordinates. The new proposed algorithm is robust and fast. It uses separation function that ensures very high robustness.

The proposed algorithm CLIP_L for line clipping by a convex polygon is significantly faster than the original Cyrus-Beck algorithm and does not need predefined clipping polygon orientation. Additional speed-up has been obtained for general case, when $w \neq 1$. It can be used for cases when a line is given by end-points in homogeneous coordinates. The CLIP_L algorithm seems to be convenient for hardware implementation, too.

The proposed algorithm will benefit in speeding up if vector-vector operations or parallel processing can be used. It can be seen that the cross-product processor will increase the performance as well. Also the algorithms increases performance if the clipping window is will be $< 0 , 1 > \times < 0 , 1 >$.

6. Acknowledgments

The author would like to express his thanks to students and colleagues at the University of West Bohemia for recommendations, constructive discussions and hints that helped to finish the work and improve this manuscript a lot. Thanks especially belong to Jana Hajkova and Petr Cizek for implementation verification in C++ and Delphi (Pascal), to Ivo Hanak for some feature evaluation in HLSL and Cg using ATI X1 card as well.

7. References

[1] Bui,D.H., Skala,V.: Fast Algorithms for Clipping Lines and Line Segments in E^2, *The Visual Computer* , Vol.14, No.1 , pp.31-37 , 1998.

[2] Coxeter,H.S.M.: Introduction to Geometry, Jihn Wiley, 1961.

[3] Cyrus,M., Beck,J.: Generalized Two- and Three_Dimensional Clipping, Computers&Graphics, Vol.2., No.1, pp.23-28, 1978.

[4] Foley,D.J., van Dam,A., Feiner,S.K., Hughes,J,F.: Computer Graphics -Principles and Practice, Addison Wesley, 2nd ed., 1990.

[5] Hartley,R., Zisserman,A.: MultiView Geometry in Computer Vision, Cambridge University Press, 2000.

[6] Johnson,M.: Proof by Duality: or the Discovery of “New” Theorems, Mathematics Today, December 1996.

[7] Liang,Y., Barsky,B.: A New concept and Method for Line Clipping, ACM Trans.on Graphics, Vol.3., No.1., pp.1-22, 1984

[8] Nicholl,T.M., Lee D.T., Nicholl R.A.: An Efficient New Algorithm for 2-D Line Clipping: Its Development and Analysis, SIGGRAPH Proceedings, Vol.21, No.4, pp.253-262, 1987.

[9] Rappaport,A.: An Efficient Algorithm for Line and Polygon Clipping, The Visual Computer, Vol.7., No.1, pp.19-28, 1991.

[10] Skala,V., Bui,D.H.: Extension of the Nicholls - Lee - Nichols Algorithm to Three Dimensions, The Visual Computer, Springer Verlag, Vol. 17, pp.236 - 242, 2001

[11] Bui,D.H., Skala,V.: Fast Algorithms for Line Segment and Line Clipping in E2, The Visual Computer, No.1, Vol.14, Springer Verlag, pp. 31-37, 1998.

[12] Skala,,V.: Algorithms for 2D Line Clipping, Eurographics 89 conference proceedings, pp.355-366, 1989.

[13] Stolfi,J,: Oriented Projective Geometry, Academic Press, 2001