Comput. & Graphics, Vol. 3, pp. 23–28
© Pergamon Press Ltd., 1978. Printed in Great Britain

0097–8493/78/0101–0023/$02.00/0

GENERALIZED TWO- AND THREE-DIMENSIONAL CLIPPING

MIKE CYRUS and JAY BECK

U.S. Air Force Human Resources Laboratory, Williams AFB, AZ 85224 and Tektronix Inc, P.O. Box 500, Beaverton, OR 97077, U.S.A.

(*Received* 29 *April* 1977; *received for publication* 4 *August* 1977)

Abstract—This paper derives a clipping algorithm and discusses both two- and three-dimensional implementations of the algorithm. The algorithm finds the proper intersection of a line with any convex planar, polygon or spacial polyhedron. Interpretation of the computed clipping coefficients can produce both interior and exterior clipping to a convex region. Finally, the implementation lends itself well to highly parallel excution which reduces execution time.

NOMENCLATURE

$\mathbf{D}$	directorix of the vector $P(\tau)$
$\mathbf{f}$	a boundary point for S
$\mathbf{n}$	inward normal
$\mathbf{n}(b)$	set of positive normals drawn from b which is a member of a set of all planes bounding S
S	convex clipping region
$\mathbf{P}$	parametrisation of a given line segment
T_i	defines a set of feasible τs used for line segment acceptance or rejection
T^*	parameterization of the line segment that lies within S, or a set of all τs such that $P(\tau)$ is contained within S
$\mathbf{w}_i$	ith plane weight vector
τ	parameter $0 < \tau < 1$ used in specifying a line segment;
ϵ	denotes membership in a set
W_B	window bottom
W_H	window hither
W_L	window left
W_R	window right
W_T	window top
W_Y	window yon
$\cap$	intersection of set
$\bigcap_{i=i}^{K}$	intersection over a set of K sets
$\times$	vector cross-product
$\otimes$	Cartesion set product.

INTRODUCTION

Clipping is a procedure used to determine the proper line segment(s) of a given line, such that the resulting segment(s) contain no points exterior to a plane that defines a boundary. Clipping, by drawing only the visible parts of a line, limits the amount of work a display device must perform. This is done by using a clipping algorithm to compute only the line segments interior (or exterior) to a defined region that has been selected for display. The resulting list of lines represents a substantially smaller amount of work for the display hardware to process.

A good clipping algorithm quickly rejects lines that lie outside the viewing area. Since speed is always desirable, algorithms that lend themselves to parallel or pipeline implementation in either hardware or firmware are sought.

At first, we will discuss only two-dimensional clipping. Later we will explore three-dimensional clipping.

N-DIMENSIONAL CLIPPING

Display regions

Typically, display regions are rectangular (see Fig. 1). The number of sides is not important to the development of this clipping algorithm, although it does affect performance. Viewing windows are always convex sets in a Hilbert space.

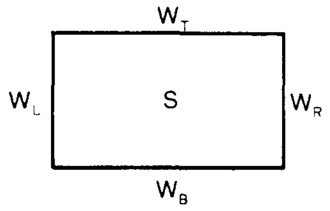


Fig. 1. Typical clipping boundaries.

The standard properties of Hilbert spaces, as well as the set concepts referred to in this paper, are found in most tests on linear analysis.

Keeping those properties in mind, the following is clear:

A straight line, intersecting the interior of a convex set, can intersect the boundary of the set in, at most, two places. In the event the convex set is closed and bounded, the straight line will intersect it in exactly two places.

Inward normals. At this point, we need to introduce the idea of "inward" normals. We say $\mathbf{n}$ is an inward normal for any boundary point $\mathbf{x}$ of S if, for any other point $\mathbf{y}$ in S,

$$\mathbf{n} \cdot (\mathbf{y} - \mathbf{x}) \geq 0$$

where "·" denotes the inner production operation. With this represention, we have defined a natural geometry over the set S.

For the set S, only four unique inward normals exist. All other normals are mathematically equivalent to these four.

The problem. Our clipping problem is to determine the points of intersection between a particular line segment and the set S. We can then partially resolve the problem by examining the parameterized behavior of the line segment with the four inward normals to S. Let's say

$\mathbf{f}$ is a boundary point for S,
$\mathbf{n}$ is an inward vector for S,
$\mathbf{P}(\tau)$ is a parameterization of a given line segment,

then, for particular values of τ,

$$\mathbf{n}\cdot[P(\tau)-\mathbf{f}]>0$$

implies $\mathbf{P}(\tau)$ is "pointed" toward the interior of S with respect to at least one of the four normals, and

$$\mathbf{n}\cdot[P(\tau)-\mathbf{f}]=0$$

implies $\mathbf{P}(\tau)$ is "pointed" parallel to the plane containing $\mathbf{f}$ and perpendicular to $\mathbf{n}$ (that is, parallel to this particular boundary of S) and

$$\mathbf{n}\cdot[P(\tau)-\mathbf{f}]<0$$

implies $\mathbf{P}(\tau)$ is "pointed" away from the set S.

So far, we have said only that any line piercing S does so at exactly two points. It also is clear that these two points of intersection do not lie on the same target planes to the boundary (a plane not completely containing a line may be intersected by it no more than once). These facts give rise to our first important result:

if $\mathbf{f}$ is a point in a particular plane bounding S for which $\mathbf{n}$ is an inward normal, then the scalar equation

$$\mathbf{n}\cdot[\mathbf{P}(\tau)-f]=0$$

has only one solution.

So there are two possibilities. First, $\mathbf{P}(\tau)$ is parallel to some plane lying along the boundary of S. Second, $\mathbf{P}(\tau)$ pierces that plane for some value of τ. Now, suppose we want to talk about drawing only a line segment:

$$\mathbf{P}(\tau)=\mathbf{P}_0+(\mathbf{P}_1-\mathbf{P}_0)\tau \qquad \tau\in[0,1].$$

It is the restriction of τ to $[0,1]$ that makes $\mathbf{P}(\tau)$ a segment.

With regard to the set S, either:

(1) the line segment is wholly contained in S, or

(2) the line segment is partly contained in S, or

(3) The line segment is no part of S at all.

Before introducing the primary result, we need the following definitions:

Define S_b as the set of all planes bounding S, that is, intersecting only the boundary of S.

For each $b\in S_b$, define $\mathbf{n}(b)$ as the set of positive normals drawn from b with respect to S.

The primary result. The set of all τ such that:

(1) $\tau\in[0,1]$,

(2) $\mathbf{n}\cdot[\mathbf{P}(\tau)-\mathbf{f}]>0$,

where for every $b\in S_b$, $\mathbf{n}\in\mathbf{n}(b)$, and $\mathbf{f}$ is an arbitrary vector in b, completely determines the line segment containment in S.

The flowchart (Fig. 5) shows the underlying geometry involved in this technique. The implementation lends itself to swift line-segment rejection.

Implementation considerations. The above result seems simple in theory, but the practice is more complex. To put the theory into practice, we need to know answers to these questions:

How many planes are involved in the convex object? If the number is large, our procedure would be tedious.

Are there any ways to "set up" the problem to facilitate the solution?

Are there any time-saving ways to accept or reject segments?

It should be clear that there is no one fastest way to examine any vector segment under all circumstances. For example, examine that rectangle again (see Fig. 2).

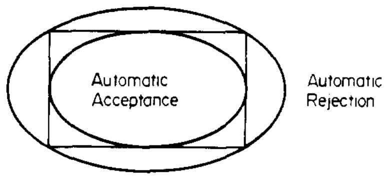


Fig. 2. Acceptance or rejection test.

We may inscribe an ellipse and circumscribe an ellipse and test $\mathbf{P}(\tau)$ for either immediate acceptance or rejection. In the event of test failure, the area we must further test is considerably smaller. This kind of test procedure is one type of conditional test, where we maximize the probability of an accurate selection. Now, let's consider the general formulation of the orientation vector technique.

Let

$$\mathbf{P}(\tau)=\mathbf{P}_0+(\mathbf{P}_1-\mathbf{P}_0)\tau \qquad \tau\in[0,1].$$

$\mathbf{P}(\tau)$ can be made a line by allowing $\tau\epsilon[-\infty,\infty]$.

The five conditions become

$$[\mathbf{P}(\tau)-\mathbf{f}_i]\cdot\mathbf{n}_i>0 \qquad i=1,2,3,4$$

or

$$\mathbf{P}_0\cdot\mathbf{n}_i+\tau(\mathbf{P}_1-\mathbf{P}_0)\cdot\mathbf{n}_i>0 \qquad i=1,2,3,4$$

or

$$\tau(\mathbf{P}_1-\mathbf{P}_0)\cdot\mathbf{n}_i>(\mathbf{f}_i-\mathbf{P}_0)\cdot\mathbf{n}_i \qquad i=1,2,3,4$$

and

$$\tau\in(0,1),$$

letting

$$\mathbf{P}_1-\mathbf{P}_0=\mathbf{D}$$

[$\mathbf{D}$ is called the directorix of the vector $P(\tau)$] and

$$\mathbf{P}_0-\mathbf{f}_i=w \qquad i=1,2,3,4,$$

[w_i is the ith plane "weight" vector].

Then we can write:

$$\tau\mathbf{D}_i\cdot\mathbf{n}_i+w_i\cdot\mathbf{n}_i>0 \qquad i=1,2,3,4.$$

The algorithm. Now we have our algorithm: for a

K-sided convex object S with inside normals:

$$\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_K.$$

Points on each plane:

$$\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K.$$

Let $\mathbf{P}(\tau)$ be the parameterization of the line running from $\mathbf{P}_0$ to $\mathbf{P}_1$.
Define:

$$\mathbf{w}_i = \mathbf{P}_0 - \mathbf{f}_i$$

$$\mathbf{D} = \mathbf{P}_1 - \mathbf{P}_0.$$

Then the $K+1$ conditions for any portion of the line segment being in S are:

$$\tau\mathbf{D}\cdot\mathbf{n}_i + \mathbf{w}_i\cdot\mathbf{n}_i > 0 \qquad i = 1, 2, \ldots, K$$
$$\tau \in [0, 1].$$

Define the set T_i such that each ith case above is satisfied:

$$T_i = (\tau | \tau\mathbf{D}\cdot\mathbf{n}_i + \mathbf{w}_i\cdot\mathbf{n}_i > 0).$$

Then for each T_i, $T_i \in [-\infty, \infty]$,
let

$$T = \bigcap_{i=1}^{k} T_i.$$

Then

$$T = [\tau | \mathbf{P}(\tau) \in S].$$

Finally, $T^* = T \cap [0, 1]$.

TWO-DIMENSIONAL CLIPPING CASE

Figure 3 defines the clipping parameters for the two-dimensional case. Using the Tektronix 4051 Graphic Computing System[1], we programmed an implementation of this algorithm. The program and sample results follow. In this program,

$$P_{0_x} = X1 \qquad P_{1_x} = X2$$

$$P_{0_y} = Y1 \qquad P_{1_y} = Y2$$

$$\mathbf{f}_1 = \begin{pmatrix} W_L \\ W_B \end{pmatrix} = \mathbf{f}_2 \qquad \mathbf{n}_1 = \begin{pmatrix} W_R - W_L \\ 0 \end{pmatrix} = -\mathbf{n}_3$$

$$\mathbf{f}_3 = \begin{pmatrix} W_R \\ W_T \end{pmatrix} = \mathbf{f}_4 \qquad \mathbf{n}_2 = \begin{pmatrix} 0 \\ W_T - W_B \end{pmatrix} = -\mathbf{n}_4.$$

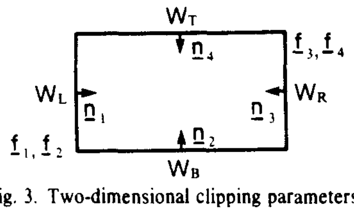


Fig. 3. Two-dimensional clipping parameters.

Figure 4 defines the major variables that are used in the program (shown in Fig. 6). We can introduce efficiency into the computation by first evaluating directly for the values of τ.

$$\mathbf{W}_1 = \begin{pmatrix} P_{0X} - W_L \\ P_{0Y} - W_B \end{pmatrix} = \mathbf{W}_2 \qquad \mathbf{W}_3 = \begin{pmatrix} P_{0X} - W_R \\ P_{0Y} - W_T \end{pmatrix} = \mathbf{W}_4$$

$$\mathbf{n}_4 = \begin{pmatrix} W_R - W_L \\ 0 \end{pmatrix} = -\mathbf{n}_3 \qquad \mathbf{n}_2 = \begin{pmatrix} 0 \\ W_T - W_B \end{pmatrix} = -\mathbf{n}_4$$

$$\mathbf{f}_1 = \begin{pmatrix} W_L \\ W_B \end{pmatrix} = \mathbf{f}_2 \qquad \mathbf{f}_3 = \begin{pmatrix} W_R \\ W_T \end{pmatrix} = \mathbf{f}_4$$

$$\mathbf{D} = \begin{pmatrix} P_{1X} - P_{0X} \\ P_{1Y} - P_{0Y} \end{pmatrix}.$$

(1) $\tau(P_{1X} - P_{0X})(W_R - W_L) > (W_L - P_{0X})(W_R - W_L)$,
(2) $\tau(P_{1Y} - P_{0Y})(W_T - W_B) > (W_B - P_{0Y})(W_T - W_B)$,
(3) $\tau(P_{1X} - P_{0X})(W_L - W_R) > (W_R - P_{0X})(W_L - W_R)$,
(4) $\tau(P_{1Y} - P_{0Y})(W_R - W_T) > (W_T - P_{0Y})(W_B - W_T)$,
(5) $\tau \in [0, 1]$.

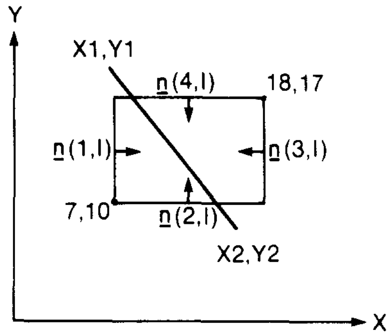


Fig. 4. Major variables used in program in Fig. 6.

Assuming that $\mathbf{P}(\tau)$ is not parallel to any side of the rectangle, then we can solve for the values of τ so that each equation (1)–(4) is satisfied identically.

$$(1) = \frac{W_L - P_{0X}}{P_{1X} - P_{0X}},$$

$$(2) = \frac{W_B - P_{0Y}}{P_{1Y} - P_{0Y}},$$

$$(3) = \frac{W_R - P_{0X}}{P_{1X} - P_{0Y}},$$

$$(4) = \frac{W_T - P_{0Y}}{P_{1Y} - P_{1X}}.$$

THREE-DIMENSIONAL CLIPPING

Since the algorithm was not restricted to two dimensions, the basic result applies in the three-dimensional case. We will examine a particular case. The set of points chosen for perspective purposes is centered with the axis center at the eye point, looking along the Z axis. The set of points is given by the Cartesian product set

$$(W_L, W_R) \otimes (W_B, W_T) \otimes (W_H, W_Y).$$

The region in space R, now being viewed as our convex

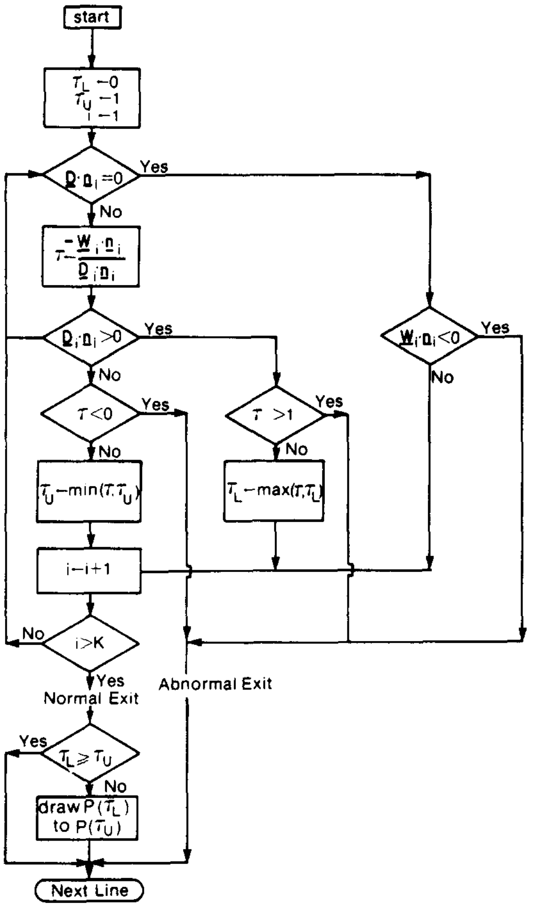


Fig. 5. Flowchart of the algorithm for the program in Fig. 6.

set, is given by the product

$$(W_L, W_R) \times (W_B, W_T) \times (W_H, W_Y).$$

In the 2-D case, there were four equations in τ and one additional constraint. In this 3-D case, there are six equations in τ with the same constraint.

There are six bounding planes, four of which contain the point [0, 0, 0]. The other two are the hither and yon planes, which are handled as special cases (see Fig. 7). Symmetry conditions imply the following:

$$W_L = -W_R \qquad W_B = -W_T.$$

These four vectors are the vectors along the intersection of the four bounding planes to the first perspective plane.

$$\mathbf{n}_1 = \mathbf{V}_1 \times \mathbf{V}_2 = (0, -2W_R W_H, 2W_R W_T)$$

$$\mathbf{n}_2 = \mathbf{V}_2 \times \mathbf{V}_3 = (2W_T W_H, 0, 2W_R W_T)$$

$$\mathbf{n}_3 = \mathbf{V}_3 \times \mathbf{V}_4 = (0, 2W_R W_H, 2W_R W_T)$$

$$\mathbf{n}_4 = \mathbf{V}_4 \times \mathbf{V}_1 = (-2W_T W_H, 0, 2W_R W_T)$$

$$\mathbf{n}_5 = (\text{Hither}) = (0, 0, 1)$$

$$\mathbf{n}_6 = (\text{Yon}) = (0, 0, -1).$$

```
100 INIT
110 PAGE
120 WINDOW 0,22,0,22
130 VIEWPORT 65,130,0,65
140 DIM F(4,2),N(4,2)
150 REM..........Define window'
160 W1=7
170 W2=18
180 W3=10
190 W4=17
200 REM..........Draw axis
210 MOVE 0,0
220 DRAW 22,0
230 MOVE 0,0
240 DRAW 0,22
250 REM..........Draw Window boundary
260 MOVE W1,W3
270 DRAW W2,W3
280 DRAW W2,W4
290 DRAW W1,W4
300 DRAW W1,W3
310 REM..........Enter line definition
320 PRINT @32,21:0,90
330 PRINT "Enter x1,y1, x2,y2 :";
340 INPUT X1,Y1,X2,Y2
350 REM..........Normals defined
360 N(1,1)=W2-W1
370 N(1,2)=0
380 N(3,1)=-N(1,1)
390 N(3,2)=-N(1,2)
400 N(2,1)=0
410 N(2,2)=W4-W3
420 N(4,1)=-N(2,1)
430 N(4,2)=-N(2,2)
440 REM..........Window boundary
450 F(1,1)=W1
460 F(1,2)=W3
470 F(2,1)=F(1,1)
480 F(2,2)=F(1,2)
490 F(3,1)=W2
500 F(3,2)=W4
510 F(4,1)=F(3,1)
520 F(4,2)=F(3,2)
530 REM..........Algorithm
540 D1=X2-X1
550 D2=Y2-Y1
560 L=0
570 U=1
580 FOR I=1 TO 4
590 W1=X1-F(I,1)
600 W2=Y1-F(I,2)
610 D=D1*N(I,1)+D2*N(I,2)
620 W=W1*N(I,1)+W2*N(I,2)
630 IF D=0 THEN 690
640 T=-W/D
650 IF D>0 THEN 710
660 IF T<0 THEN 860
670 U=T MIN U
680 GO TO 730
690 IF W<0 THEN 860
700 GO TO 730
710 IF T>1 THEN 860
720 L=T MAX L
730 NEXT I
740 REM..........Segment drawn
750 IF L=>U THEN 86
760 X4=X1+(X2-X1)*L
770 Y4=Y1+(Y2-Y1)*L
780 X5=X1+(X2-X1)*U
790 Y5=Y1+(Y2-Y1)*U
800 MOVE X4,Y4
810 DRAW X5,Y5
820 PRINT @32,21:0,30
830 PRINT "SEGMENT DRAWN"
840 END
850 REM..........Segment rejected
860 PRINT @32,21:0,30
870 PRINT "SEGMENT REJECTED"
880 END
```

Fig. 6.

Let

$\underline{V}_1 = [W_R, W_T, W_H]$
$\underline{V}_2 = [\ W_R, W_T, W_H]$
$\underline{V}_3 = [\ W_R, \ W_T, W_H]$
$\underline{V}_4 = [W_R, -W_T, W_H]$

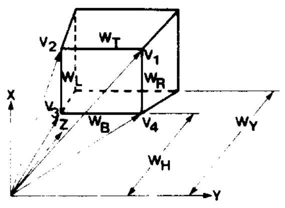


Fig. 7. 3-D Clipping parameters.

$$\mathbf{f}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{f}_2 = \mathbf{f}_3 = \mathbf{f}_4$$

$$\mathbf{f}_5 = \begin{pmatrix} 0 \\ 0 \\ W_H \end{pmatrix}$$

$$\mathbf{f}_6 = \begin{pmatrix} 0 \\ 0 \\ W_Y \end{pmatrix}.$$

$$\mathbf{D} = \mathbf{P}_1 - \mathbf{P}_0$$
$$\mathbf{W}_i = \mathbf{P}_0 - \mathbf{f}_i$$
$$\mathbf{W}_1 = \mathbf{P}_0 \quad \mathbf{W}_2 = \mathbf{P}_0 \quad \mathbf{W}_3 = \mathbf{P}_0 \quad \mathbf{W}_4 = \mathbf{P}_0$$

$$\mathbf{W}_5 = \begin{bmatrix} +P_{0X} \\ +P_{0X} \\ -W_H + P_{0Z} \end{bmatrix} \qquad W_6 = \begin{bmatrix} +P_{0X} \\ +P_{0Y} \\ -W_Y + P_{0Z} \end{bmatrix}.$$

The same rules apply to $D \cdot n_i$ and $W_i \cdot n_i$ as before. The primary formula is

$$\tau = \frac{-w_i \cdot n_i}{D \cdot n_i}$$

for each i, giving

$$\tau_1 = \frac{-W_T P_{0Z} + W_H P_{0Y}}{(P_{1Z} - P_{0Z})W_T - (P_{1Y} - P_{0Y})W_H}$$

$$\tau_2 = -\frac{W_H P_{0X} + W_R P_{0Z}}{(P_{1X} - P_{0X})W_H + (P_{1Z} - P_{0Z})W_R}$$

$$\tau_3 = -\frac{W_H P_{0Y} + W_T P_{0Z}}{(P_{1Y} - P_{0Y})W_H + (P_{1Z} - P_{0Z})W_T}$$

$$\tau_4 = \frac{-W_R P_{0Z} + W_H P_{0X}}{(P_{1Z} - P_{0Z})W_R - (P_{1X} - P_{0X})W_H}$$

$$\tau_5 = \frac{-P_{0Z} + W_H}{P_{1Z} - P_{0Z}}$$

$$\tau_6 = \frac{-W_Y + P_{0Z}}{P_{0Z} - P_{1Z}}.$$

These values of τ, define the intersection points of $P(\tau)$ with each bounding plane.

It can be seen from this analysis that while the computation of τ for 3-D case may appear to be a lengthy computation, each value of τ can be computed in a separate parallel process.

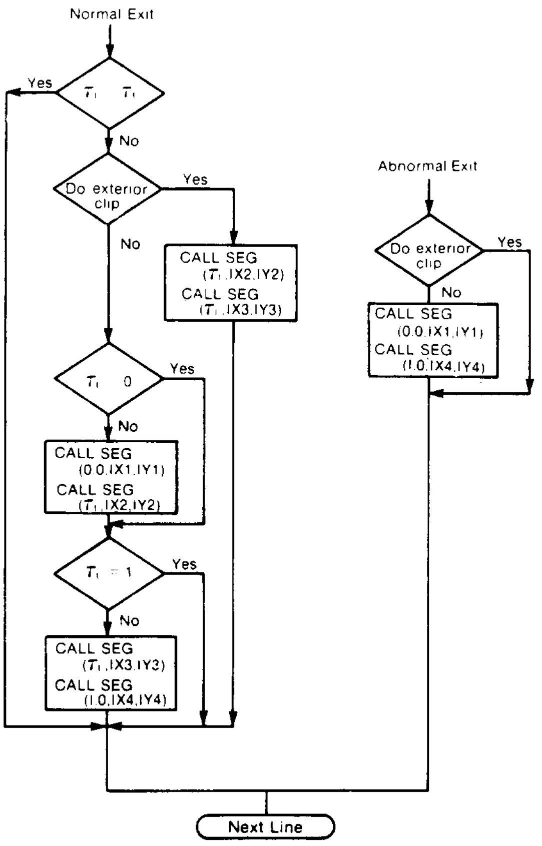


Fig. 9. Additional clipping algorithm logic.

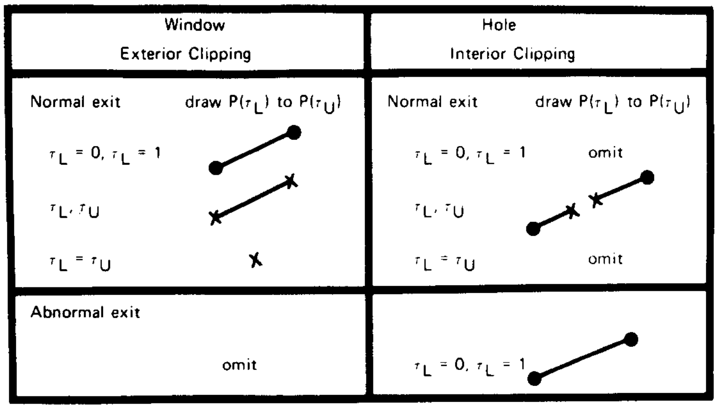
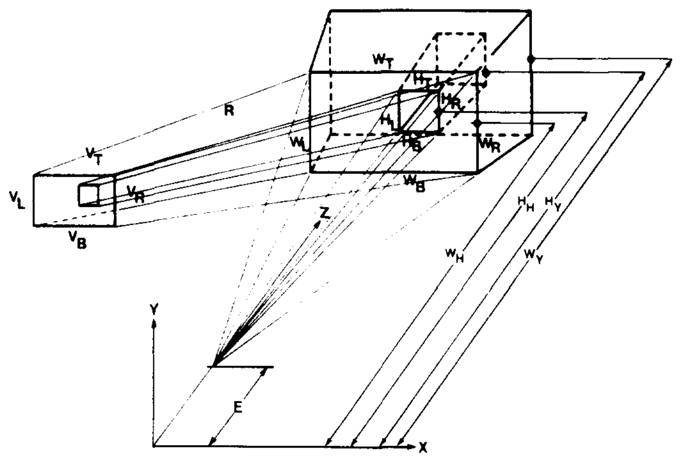


Fig. 8. Exterior and interior clipping possibilities.



WINDOW W_L W_R W_B W_T W_H W_Y E Where: $W_L \leqslant H_L$ $W_R \geqslant H_R$
HOLE H_L H_R H_B H_T H_H H_Y $W_B \leqslant H_B$ $W_T \geqslant H_T$
VIEWPORT V_L V_R V_B V_T $W_H \leqslant H_H$ $W_Y \geqslant H_Y$

Fig. 10. 3-D interior parameters and exterior clipping.

EXTENSION TO INTERIOR CLIPPING

Our discussion so far has been limited to determining a line segment that is either rejected or clipped to fit inside a convex set S. Our output (τ_L, τ_U) can also be used to draw the complement. The complement is the line segment(s) exterior to a given convex set S.

The possibilities are summarized in Fig. 8.

Interpreting values of (τ_L, τ_U) for exterior and interior cases (refer to Fig. 5), complete implementation of this algorithm might be used to apply exterior clipping to a "hole" contained within the convex window.

If we define a function which performs $P(\tau)$ as SEG, the flowchart in Fig. 5 can be extended as shown in Fig. 9.

We can combine the use of both interior and exterior clipping methods to aid us in examining highly complex three-dimensional figures. We define a frustrum of vision that serves to define an area-of-interest via exterior clipping. Within this frustrum a smaller frustrum is defined using interior clipping to look through selected parts of the object that falls within the area of interest. These overlapping frustrums are illustrated in Fig. 10.

CONCLUSION

Implementation of true three-dimension interior and exterior clipping gives a display processor added flexibility in studying complex shapes without using hidden-line elimination. Careful placement of the boundary planes enables the viewer to section shapes and remove obscuring surfaces.

REFERENCES

1. Douglas K. Brotz, Intersection polyhedra with successive planes. *Comput. Graphics* **2**, No. 1, 1–6 (1976).
2. William M. Newman & Robert F. Sproull, *Principles of Interactive Computer Graphics*, Chap. 7, p. 249. McGraw-Hill, New York (1973).
3. Michael B. Stephenson & Henry N. Christiansen, A polygon clipping and capping algorithm and a display system for three dimensional finite element models. *SIGGRAPH-ACM* **9**, No. 3, 1 (Fall 1975).
4. Ivan E. Sutherland and Gary W. Hodgman, Re-entrant polygon clipping. *CACM* **17**, No. 1, 32 (January 1974).
5. Tektronix Inc., 4051 *Graphic System Reference Manual*. Tektronix Inc. Beaverton, Oregon (1976).