ISSN 0254-2757

# QI QUÆSTIONES INFORMATICÆ

Volume 6 • Number 3 November 1988



**The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists**

**Die amptelike vaktydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes**

# QUÆSTIONES INFORMATICÆ

**The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists**

**Die amptelike vaktydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes**



## Production

Mr Q H Gee
Department of Computer Science
University of the Witwatersrand
Johannesburg
Wits
2050

## Subscriptions

The annual subscription is

| | SA | US | UK |
|---|---|---|---|
| Individuals | R20 | $7 | £5 |
| Institutions | R30 | $14 | £10 |

to be sent to:
*Computer Society of South Africa*
*Box 1714 Halfway House 1685*

**Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.**

# Frame Clipping of Polygons

E C Anderssen and S H von Solms
*Department of Computer Science, Rand Afrikaans University, P O Box 524, Johannesburg, 2000.*

## Abstract

*A polygon clipping algorithm is presented for the clipping of closed geographical areas such as dams or forests, against a surrounding rectangular window-frame. The algorithm employs a line clipping procedure which divides the vertices of the clipped polygon into three groups: interior vertices, border vertices and exterior vertices. The vertices on the borders of the window-frame are sorted and stored in a control vector. The control vector determines the sequence in which the clipped vertices are loaded for plotting purposes.*
***Keywords**: Computer graphics, Line clipping, Polygon clipping.*

Received September 1988, Accepted October 1988

## 1. Introduction

There are a number of possible ways in which a realistic representation of a geographical area can be done. A model of the area under consideration can be built; a scale map of the area can be constructed, or a sketch map of the area can be drawn. Using the sketch map approach, objects on a map are not necessarily drawn to scale. Those objects which are to be emphasized, can be represented as enlarged figures or can even be brightly coloured. Typically, sketch maps are used as tourist guides or as a means of representing historical events, such as field battles. Closed geographical regions, for instance dams, forests, cities or even larger areas such as countries, are frequently represented on sketch maps. An object belonging to this type can always be represented as an area with a closed boundary. In order to emphasize the cohesiveness of a closed geographical region, hatching techniques are usually employed in which the enclosed region is filled or painted with some characteristic pattern.

It often happens that a closed geographical region contains one or more islands. The existence of an island is indicated by either leaving the island unhatched, or hatching it with a different pattern to distinguish it from the surrounding enclosing area.

When the map of a geographical area is to be represented as a graphical image on the screen of a computer, it is not always possible to display the entire map of the area. Any part of the map can be selected for display by enclosing the required area within a rectangular window. The parts of the map falling outside the window boundary are discarded or clipped. Only those parts of the area inside the window boundary are retained for display purposes.

In the case of non-convex closed areas such as dams, this clipping procedure can give rise to rather complex figures. A number of factors contributing to this complexity can be identified:

* The inclusion of the borders and corners of the window-frame as part of the clipped area;
* The presence of islands inside the clipped area;
* The forming of disjoint clipped areas as a result of concave inflection points in the original area.An example of such a complex region is illustrated in figure 1.

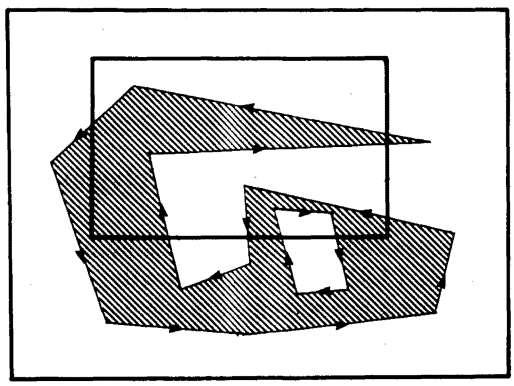


**Figure 1 (a): Closed geographical area with island showing the window-frame before clipping.**

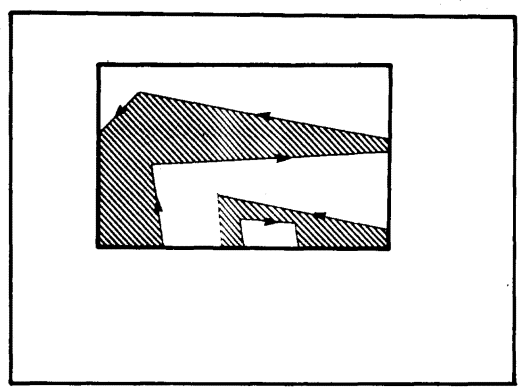


**Figure 1 (b): The clipped portion of the geographical area showing two disjoint areas of the same region.**

In computer graphics, a closed geographical area such as a dam, can be represented as a polygon. The polygon clipping problem was solved a number of years ago. In their 1974 article, Sutherland and Hodgman [7] mention an existing solution due to Archuleta. They then present a very general re-entrant polygon clipping algorithm. Other solutions and extensions were later suggested [4,6,8]. A major reason for the development of some of these clipping algorithms was for solving the hidden surface problem. These solutions are therefore very comprehensive and provide for clipping in three dimensions as well as for clipping against another polygon.

The solution presented in this article is not as comprehensive as the more general solutions mentioned above. This approach was taken because the solution was developed for a rather restricted application area, namely the clipping of closed geographical regions against a rectangular border in two dimensions. In geographical applications, islands frequently occur inside closed regions. For this reason the proposed solution also provides for the clipping of islands. An advantage of limiting the algorithm to the solution of the closed geographical area problem, is that the solution is presented in terms of rather simple and straight forward algorithms which can easily be implemented on a micro computer with graphical capabilities.

## 2. The Representation of a Polygon

A polygon can be represented as an ordered set of vertices [7,4]. When graphically displaying a polygon on the screen of a computer, an edge is drawn between each adjacent vertex pair. For the purpose of closing the polygon, the last and the first vertices are considered to be adjacent. A vertex is represented as a (X,Y) coordinate pair, indicating the cartesian coordinate of the vertex on the map of the geographical area.

It was already indicated that a polygon may contain one or more interior island polygons. It is therefore necessary to convey the position of the first coordinate of each of these island polygons to the clipping algorithm. This is done by inserting a liftpen control character preceding the first coordinate of each island polygon (see figure 2) in the vector containing the coordinates of the vertices.

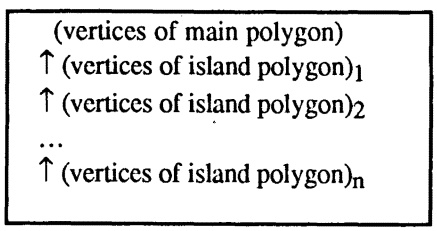



**Figure 2: Vector containing the coordinates of the vertices of the geographical area to be clipped.**

The ↑ indicates a liftpen control character, indicating the starting position of an island polygon.

The algorithm requires that the vertices of the polygons are loaded into the vertex vector in a specified order. The vertices of the main polygon are loaded in such a way that the interior of the polygon is always to the left when moving from vertex to vertex. This implies that movement around the main polygon is always in an anti-clockwise direction. The vertices of an island polygon are loaded in such a way that the interior of the island is always to the right when moving from vertex to vertex. This implies a clockwise movement around an island polygon. The direction of movement around the polygon is not important for the functioning of the algorithm. It can be changed if necessary, by applying a few minor modifications to the algorithm.

The window-frame against which the clipping is done, is defined by entering the coordinates of the bottom left and upper right corners of the frame.

## 3. The Clipping Algorithm

The algorithm used for clipping a polygon against a rectangular window, is divided into two separate algorithms. The first algorithm performs the line clipping of the vertex vector against the window. The second algorithm assembles these clipped values into the output vector. Apart from these clipped vertices, the output vector contains the coordinates of included corners and also the control characters indicating the start coordinates of island polygons.

A more detailed description of the two algorithms is given in the following paragraphs.

### 3.1 The line clipping algorithm

In the initial stage of the line clipping algorithm, each pair of vertices in the (X,Y)-vector is systematically clipped against the window using the Cohen-Sutherland line clipping algorithm as presented by Newman and Sproull [5:66-67].

The clipping process divides the vertices into three groups (see figure 3):

Interior vertices

A vertex is interior when its (X,Y)-coordinate falls within the borders of the window (see figure 3 points A and E).

Border vertices

A border vertex occurs when its (X,Y)-coordinate falls on a window-border (see figure 3 point B). In the case where the edge between two adjacent vertices and a border line intersect, one vertex lies inside and the other vertex outside the window (see figure 3 points E and D). Under these circumstances, a border

vertex is formed in place of the outside vertex, which is discarded (see figure 3 point C).

Exterior vertices

A vertex is exterior when its (X,Y)-coordinate falls outside the window-boundary. Two or more consecutive exterior vertices are reduced to one equivalent exterior vertex. This is done because the second algorithm responsible for assembling the clipped polygon, only uses an exterior vertex as a switch to control the selection of other vertices (see figure 3 point D).

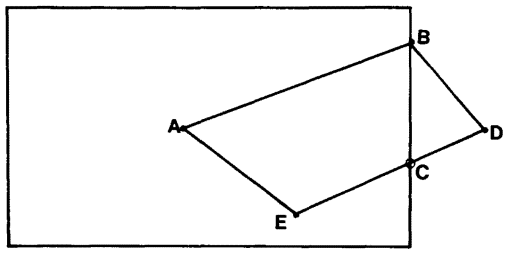



**Figure 3: The vertex classes after application of the line clipping procedure:**

**(a) Interior vertex (point A)**

**(b) Border vertex:**

**(i) Vertex on a window-border (point B)**

**(ii) Calculated vertex on a window-border (point C)**

**(c) Exterior vertex (point D)**

The four borders of the rectangular window are numbered according to the following scheme:

* Border 1 : Bottom border
* Border 2 : Right border
* Border 3 : Top border
* Border 4 : Left border

During the clipping process, each vertex is tagged, indicating the group to which it belongs. Vertices are tagged in the following way:

(1) Interior vertex: Coordinate is tagged with a null.

(2) Border vertex: Coordinate is tagged with border number on which it occurs.

(3) Exterior vertex: Coordinate is discarded and replaced with a tag value of 5.

The tags are stored in a tag vector, T. For explanatory purposes, the tags are represented in the following way:

(1) Interior vertex : $(i)^{+}$ : Where the tag '+' indicates the interior vertex i.

(2) Border vertex : $(i)^{N}$ : Where the tag 'N' indicates the border number of vertex i.

(3) Exterior vertex : $(0)^{-}$ : Where the tag '–' indicates an exterior vertex.

An example of a polygon and the window against which it is clipped, is shown in figure 4.

The explanatory representation of the (X,Y)-vector, obtained after clipping the polygon in figure 4, is given in figure 5:

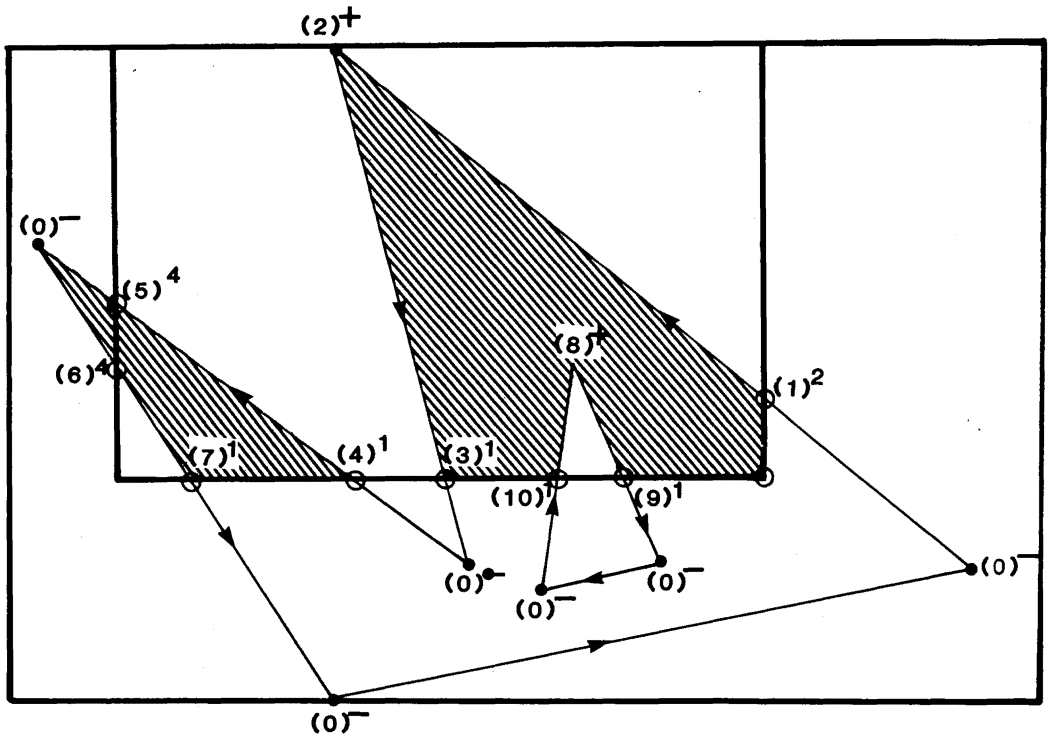



**Figure 4: A polygon clipped against a window. The interpolated border vertices are clearly marked with small circles on the different borders.**

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tagged (XY) vector | $(1)^2$ | $(2)^+$ | $(3)^1$ | $(0)^-$ | $(4)^1$ | $(5)^4$ | $(0)^-$ | $(6)^4$ | $(7)^1$ | $(0)^-$ | ↑ | $(8)^+$ | $(9)^1$ | $(0)^-$ | $(10)^1$ | ⊥ |

**Figure 5: The (X,Y)-vector obtained after the line clipping step. The liftpen and end-of-vector control characters are indicated with ↑ and ⊥ respectively.**

Although the vertex represented as $(2)^+$ in figure 5 falls on a border, it is not marked as such, but is classified as an inflection point. An inflection point is formed when the vertex following a border vertex is not an exterior vertex. Inflection points are classified as interior vertices.

In the final stage of the line clipping algorithm, the border vertices of the (X,Y)-vector are sorted. The (X,Y)-indexes of the sorted vertices are loaded in the R-vector. The border vertices are sorted in the following sequence:

* Vertices on border 1: In ascending X coordinate sequence.
* Vertices on border 2: In ascending Y coordinate sequence.
* Vertices on border 3: In descending X coordinate sequence.
* Vertices on border 4: In descending Y coordinate sequence.

The indexes of the sorted (X,Y)-values in the above example are loaded into the R-vector (figure 6).

### 3.2 The polygon assemble algorithm

3.2.1 Introduction

The input to the polygon assemble algorithm are:

* The (X,Y)-vector, containing the coordinates of the Clipped polygon;
* The T-vector, containing the tag values of the vertices;
* The R-vector, containing the (X,Y)-indexes of the sorted border vertices.

The algorithm systematically traverses the (X,Y)-vector, selecting internal and border vertices forming a closed polygon. The selected vertex coordinates are marked as done, and are then transferred to the (U,V)-output vector. This process is repeated until all the internal and border vertices have been processed.

The algorithm selects an interior or border vertex as starting point. The vertices following this starting point are transferred to the (U,V)-output vector until an exterior vertex is located. The border vertex just prior to this exterior vertex is an exit vertex, and is used to determine the starting border vertex of the next sequence of vertices to be transferred to the output vector. The starting vertex of the next sequence is the border vertex immediately following the exit vertex in the sorted R-vector.

The necessary control characters, indicating the starting coordinates of island polygons, are also loaded into the (U,V)- output vector. After all the relevant coordinates and control characters have been loaded in this way, the output vector is ready to be processed by a suitable plot and hatching algorithm.

3.2.2 A description of the functioning of the algorithm

An exposition of the functioning of the algorithm in assembling the output vector using the clipped polygon (figure 4), is given in the following steps. The two vectors being used as input are given in figure 5 and figure 6.

Step 1

The starting vertex selected is $(1)^2$. This vertex is loaded into the (U,V)-vector and marked as done.
Contents of (U,V): $(1)^2$

Step 2

The vertex following $(1)^2$ is the interior vertex $(2)^+$. This value is loaded into (U,V). The vertex is marked as done.
Contents of (U,V): $(1)^2$ $(2)^+$.

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| XY-Index | 9 | 5 | 3 | 15 | 13 | 1 | 6 | 8 |
| Tagged (XY) vector | $(7)^1$ | $(4)^1$ | $(3)^1$ | $(10)^1$ | $(9)^1$ | $(1)^2$ | $(5)^4$ | $(6)^4$ |

**Figure 6: The R-vector contains the (X,Y)-indexes of the sorted (X,Y)-values.**

Step 3
The next vertex $(3)^1$ is on a border. It is loaded into the (U,V)-vector and marked as done. It is followed by an exterior vertex. The R-vector is therefore used in determining that $(10)^1$ is the successor of $(3)^1$.
Contents of (U,V): $(1)^2$ $(2)^+$ $(3)^1$.

Step 4
Before $(10)^1$ is loaded into (U,V), a corner test is performed. Because both $(3)^1$ and $(10)^1$ are border vertices on the same boundary, no corner is selected. The vertex $(10)^1$ is loaded into (U,V) and marked as done.
Contents of (U,V): $(1)^2$ $(2)^+$ $(3)^1$ $(10)^1$.

Step 5
As $(10)^1$ is the final vertex in the (X,Y) vector, the next vertex is determined by locating the nearest liftpen control character. The vertex selected in this way is $(8)^+$. This vertex is loaded and marked as done.
Contents of (U,V): $(1)^2$ $(2)^+$ $(3)^1$ $(10)^1$ $(8)^+$.

Step 6
The vertex $(9)^1$ follows $(8)^+$. It is loaded into (U,V) and marked as done. The vertex following $(9)^1$ is exterior. The R-vector is used to determine which vertex is to be loaded next. The node $(1)^2$ is selected in this way.
Contents of (U,V): $(1)^2$ $(2)^+$ $(3)^1$ $(10)^1$ $(8)^+$ $(9)^1$.

Step 7
A corner is inserted between the two adjacent nodes $(9)^1$ and $(1)^2$ because they lie on different boundaries.
Contents of (U,V): $(1)^2$ $(2)^+$ $(3)^1$ $(10)^1$ $(8)^+$ $(9)^1$ $\text{Corner}_{12}$.

Step 8
The node $(1)^2$ is the start node. The cycle is therefore completed. The liftpen control character is loaded into the (U,V)-vector to indicate the closure of the clipped polygon.
Contents of (U,V): $(1)^2$ $(2)^+$ $(3)^1$ $(10)^1$ $(8)^+$ $(9)^1$ $\text{Corner}_{12}$ $\uparrow$.

Step 9
The next cycle is completed in a similar way, starting with $(4)^1$. The eventual contents of the (U,V) vector is:
$(1)^2$ $(2)^+$ $(3)^1$ $(10)^1$ $(8)^+$ $(9)^1$ $\text{Corner}_{12}$ $\uparrow$
$(4)^1$ $(5)^4$ $(6)^4$ $(7)^1$ $\uparrow$.

## 4. Technical Considerations and Results

The clipping program was developed on a HP-150 micro computer, using Turbo Pascal. (See Appendix A for a pseudo Pascal version of the algorithm.)

The output of the clipping process, the (U,V)-vector, serves as input to the polygon filling procedure. The filling algorithm used in the procedure, is the standard HP-150 routine [3:5-74]. The format of the (U,V)-vector is such that any other suitable polygon filling algorithm [1,2] could be used instead.

The unclipped and clipped computer representations of the polygon in figure 4, are given in figure 7 and figure 8 respectively.

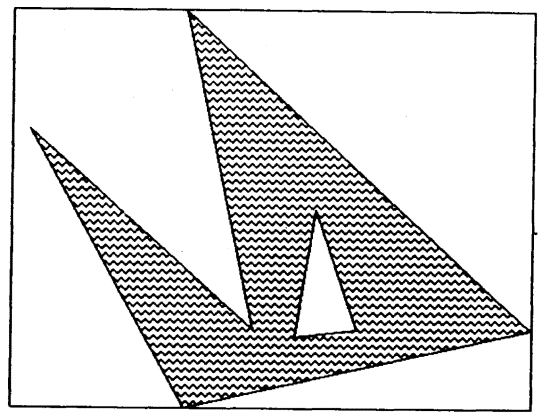
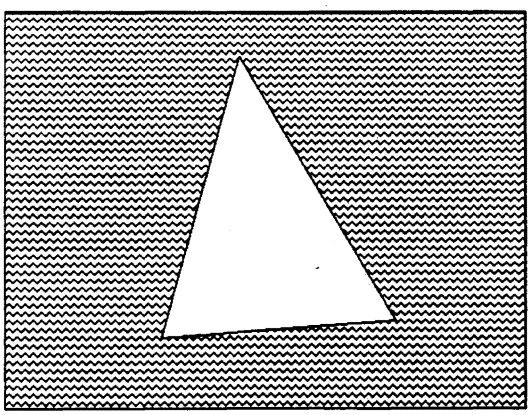
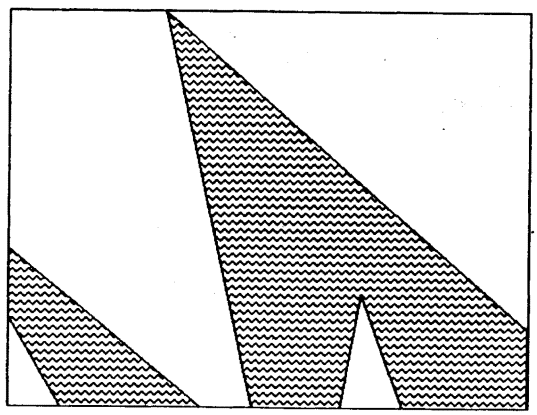


**Figure 7: The unclipped polygon**



**(a) The island and surrounding area**



**(b) The main polygon and island showing disjoint areas**

**Figure 8 Clipped representations of the polygon**

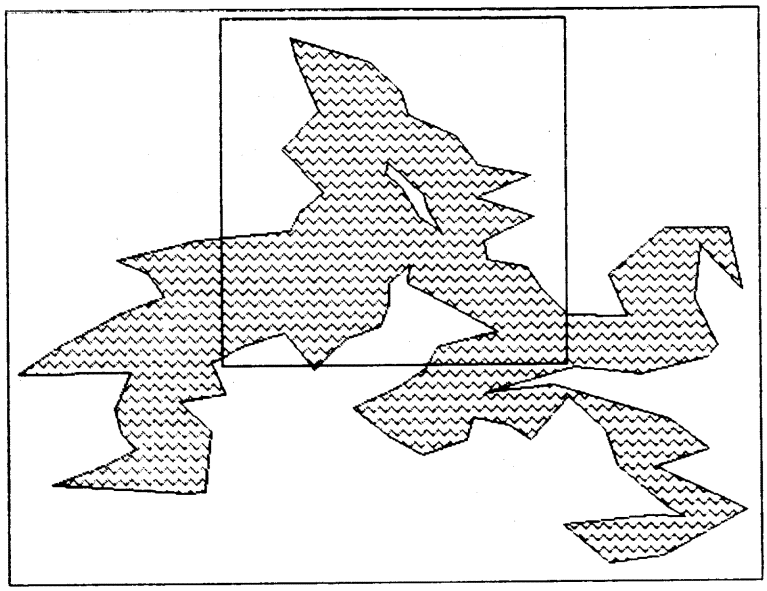


**Figure 9: The Vaaldam represented as a polygon**

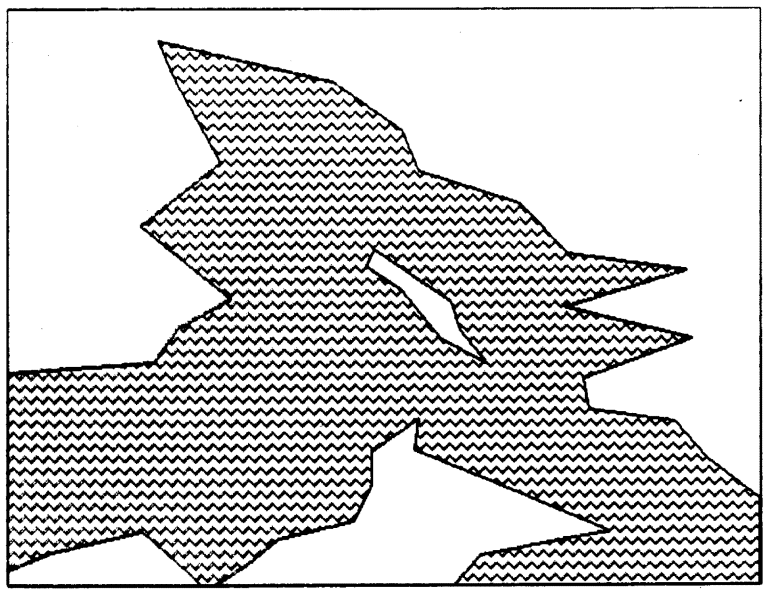


**Figure 10: The clipped central portion of the Vaaldam**

As a practical application of the technique, the perimeter of the Vaaldam was digitised. The resultant polygon, representing the dam and its island, is given in figure 9. The central portion of the dam, as demarcated by the window in figure 9, is represented as the clipped polygon in figure 10.

## 5. Summary

A polygon clipping algorithm for the clipping of a closed geographical area against a surrounding rectangular border was presented.

The algorithm uses a line clipping procedure to divide the vertices of the polygon describing the area into three groups. The first group comprises vertices exterior to the frame-area, the second group vertices on the borders of the frame-area, and the third group vertices interior to the frame-area. The vertices on the borders are sorted and loaded into a control vector. The interior and border vertices are transferred to the output vector in such a way that the region to be filled is always located to the left while traversing the main polygon and to the right in the case of an island. If a border is crossed when moving from inside the frame-area to outside, the next node is selected from the control vector in such a way that the direction of movement through the polygon is unaffected.

## References


[1] M Berger, [1986], *Computer Graphics with Pascal*, Benjamin/Cummings Publ. Company Inc. Menlo Park, Calif.

[2] J D Foley and A Van Dam, [1984], *Fundamentals of Interactive Computer Graphics*, reprinted with corrections, Addison-Wesley, July 1984.

[3] HP 150 Programmer's Reference Manual, [1984], Product 45435A, Part Number 45435/90002, Printed in the USA.

[4] Y-D Laing and B A Barsky, [1983], An Analysis and Algorithm for Polygon Clipping, *Comm. ACM*, **26** (11), 868-877.

[5] W M Newman and R F Sproull, [1981], *Principles of Interactive Computer Graphics*, Second Edition, McGraw-Hill.

[6] D F Rogers and L M Rybak, [1985], On an Efficient General Line-Clipping Algorithm, *IEEE CG&A*, 82-86.

[7]I E Sutherland and G W Hodgman, [1974], Reentrant Polygon Clipping, *Comm. ACM*, **17** (1), 32-42.

[8] C J Van Wyk, [1984], Clipping to the Boundary of a Circular-Arc Polygon, *Computer Vision, Graphics, and Image Processing*, 25, 383-392.


## Appendix A: Pseudo Pascal Version of the Algorithm

```
Procedure BuildDisplayPolygon(X,Y,T,R,Irmax,CoX,CoY,U,V,Iuv);

{The parameters have the following meaning:
      X:     The input vector containing clipped X-coordinates
      Y:     The input vector containing clipped Y-coordinates
      T:     The tag vector containing the tags of the input vector
             (i)   Interior vertices tagged with 0
             (ii)  Border vertices tagged with 1,2,3,4 depending on the border of the frame
             (iii) Exterior vertices tagged with 5
      R:     The vector containing the indexes of the sorted X,Y vector
      Irmax: The maximum number of indexes in the R vector
      CoX:   The X-coordinates of the four corners
      CoY:   The Y-coordinates of the four corners
      U:     The output vector containing the X-coordinates
      V:     The output vector containing the Y-coordinates
      Iuv:   The maximum number of entries in the output vector }

begin
   StartIndex in XY := 1;
   Iw {workindex in XY} := StartIndex;
   while (there are unmarked vertices in the XY-vector) do
   begin
      Store XY[Iw] at the next position of the output vector UV;
      Iw := Iw + 1;
      if XY[Iw]=liftpen then Iw:=index of previous liftpen + 1;

      while ( Iw <> StartIndex ) do {cycle not completed}
      begin
         if XY[Iw] is tagged as interior then PointCategory := internal;
         if XY[Iw] is tagged as border vertex
         then begin
            CurrentBorder := Tag of the border vertex;
            IndexOfCurrentBorderVertex {in XY} := Iw;
            PointCategory := internal;
         end;

         if XY[Iw] is tagged as exterior then PointCategory := external;
```

```
        Case PointCategory of

        internal:
        begin
            Store XY[Iw] at the next position of output vector UV;
            Mark tag of XY[Iw] as done;
            Iw := Iw + 1;
            if XY[Iw] = liftpen then Iw := index of previous liftpen + 1;
        end; {internal}

        external:
        begin
            Find IndexOfCurrentBorderVertex in the sorted R-vector at Ir;
            IndexOfNextBorderVertex {in XY} := R[Ir+1];
            NextBorder := Tag of XY[IndexOfNextBorderVertex];

            if any intervening corners between CurrentBorder and NextBorder
            then Load the coordinates of the relevant corners into the next available positions in the output
                 vector UV;

            if IndexOfNextBorderVertex = StartIndex
            then Iw := StartIndex; {cycle completed}
            else begin
                Store XY[IndexOfNextBorderVertex] at the next position of the output vector UV;
                Iw := IndexOfNextBorderVertex;
                IndexOfCurrentBorderVertex:=IndexOfNextBorderVertex;
                Mark tag of XY[Iw] as done;
                Iw := Iw + 1;
                if XY[Iw] = liftpen then Iw:=index of previous liftpen + 1;
            end;
        end; {external} end; {case}
    end; {while}

    Store XY[StartIndex] at the next position of vector UV;
    Insert liftpen at the following position of UV;
    Scan the XY vector for any unmarked vertices;

    if an unmarked vertex in XY exists
    then begin
        StartIndex := index of unmarked vertex;
        Iw := StartIndex;
    end;
  end; {while}
end. {routine}
```

# NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Professor ~~J M Bishop~~ D.C. Kaire
Department of Computer Science
University of the Witwatersrand
Johannesburg
Wits
2050

## Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review Categories.

Manuscripts may be provided on disc ~~using any Apple Macintosh package or~~ in ASCII format, once that a submitted paper has been accepted.

For authors wishing to provide camera-ready copy, a page specification is freely available on request from the Editor.

## Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Charges

## Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

## References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:


[1] E. Ashcroft and Z. Manna, [1972], The Translation of 'GOTO' Programs to 'WHILE' programs, *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.

[2] C. Bohm and G. Jacopini, [1966], Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9**, 366-371.

[3] S. Ginsburg, [1966], *Mathematical Theory of Context-free Languages,* McGraw Hill, New York.


## Proofs

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimise the risk of the author's contribution having to be held over to a later issue.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

## Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion ~~of recent problems~~.