ISSN 0254–2757

QI QUÆSTIONES INFORMATICÆ

Volume 5 • Number 2 **October 1987**

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereeneging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

An official publication of the Computer Society of South Africa
and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa
en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

Annual subscription are as follows:

	SA	US	UK
Individuals	R10	$7	£5
Institutions	R15	$14	£10

Computer Society of South Africa
Box 1714 Halfway House

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.

POLYGON SHADING ON VECTOR TYPE DEVICES

C F Scheepers
Computer Science Division
National Research Institute for Mathematical Sciences
CSIR, P 0 BOX 395, Pretoria, 0001

ABSTRACT

A method is presented whereby the interior of boundary-defined regions on vector type devices may be shaded using regular line, cross-hatch and dot patterns. Different shades are realized by changing shading parameters such as line width, distance between consecutive lines or dots, and orientation of patterns.

The algorithm partitions any polygon (not necessarily convex), into mutually exclusive, pseudo-monotone polygons that can then be shaded independently using a fast procedure. The partitioning process is based on the topology of the polygon, and notions such as type-1 and type-2 critical points are introduced.

Shading is frequently used in computer graphics applications, such as cartography, engineering graphics, art, animation and hidden-line removal.

KEYWORDS AND PHRASES Computer graphics, shading, filling, monotone polygons, cartography, map-making.

1. INTRODUCTION

The shading of regions with a regular pattern of lines, dots or symbols is a task common to many applications in computer graphics. Examples of applications are choropleth maps in cartography [3], cross-sections in engineering drawings [11], scene enhancement in animation, and light intensity approximation after hidden-line removal in three-dimensional graphics.

The outer and possible inner boundaries of regions are often represented by one or more simply-connected polygons. These polygons have non-self-intersecting edges connecting a sequence of outline coordinates. The sequence of outline coordinates (or vertices) is ordered in such a way that the significant region always lies to the same side of the edges that connect consecutive vertices. Following this convention and for the sake of convenience, the vertices on the outer boundary of a region are henceforth ordered in a clockwise direction. If a region contains holes or islands, it has inner boundaries. The vertices on inner boundaries should then be ordered in a counterclockwise direction in accordance with the convention (figure 1).

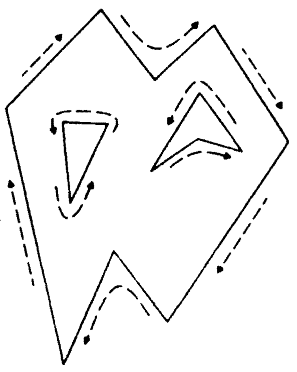

figure 1

A Region with Islands

The task at hand is to shade regions with sets of regular patterns, typically line, cross-hatch and dot patterns. The appearance of shading patterns is affected by changing input shading parameters such as line width and colour, distance between consecutive parallel lines, and angle of shading lines. Consequently, a dense pattern could be constructed that would appear darker than one with less density.

The literature on region shading (filling) in a raster type environment is extensive (see [1]). The well-known polygon filling technique of scan-converting boundary-defined polygons and extensions to this technique, are discussed in [4, 5, 7, 10]. Although adaptable to a vector type environment as in [2], these techniques are often not tailored to the whims of specific requirements in vector based systems. Brassel and Fegeas [1] note that cartographic applications such as map-making (typically requiring thousands of polygons with hundreds of vertices) can barely be realized successfully using this traditional approach. The reasons are twofold. Firstly, traditional algorithms are highly sensitive to the topology of regions and the density of shade lines, and secondly, these algorithms often use extensive extra storage and have a high order of complexity.

On vector type devices, a different approach to the shading problem is necessary to enable efficient utilization of available devices. Brassel and Fegeas present a very elegant algorithm that handles shading in a way similar to the filling of an arbitrarily shaped container with water, using an influx pipe at the lowest point of the container (figure 2). Water will fill the container in the order of the labelled volumes. This analogy illustrates an approach to partition a polygon into more manageable subparts prior to shading. Lee [6] employs a similar technique, except that partitioning is more efficient even though a greater number of subparts result. Cromley [3] also refers to the Brassel-Fegeas algorithm and improves on the speed of the partitioning algorithm. For producing sparse shading patterns, Cromley's technique is more efficient than the previously mentioned algorithms.

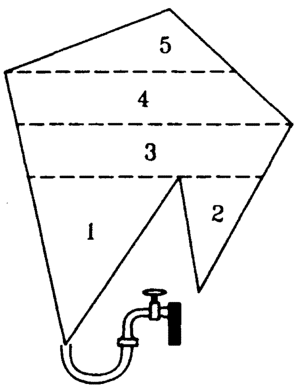

figure 2

Order of Shading in the Brassel-Fegeas Algorithm

The shading technique presented here is similar to the approach taken in these papers. A new partitioning algorithm extending on previous ideas and introducing some new concepts is presented. This algorithm yields less subparts after the partitioning process than the previous algorithms, resulting in faster shading of regions.

2. TERMINOLOGY

A simply-connected polygon P is represented by an ordered list of its n vertices (V_0, V_1,.., V_{n-1}) and consists of a circular sequence of edges

(V_i, V_{i+1}), for i = 0,1,...,n-1 (index addition/subtraction modulo n),

each edge being a line segment connecting consecutive vertices. The polygon will divide the plane into two regions, the interior and exterior regions. According to the convention presented in the introduction, the order of occurrence of the vertices identifies whether the interior or exterior of the polygon forms part of the region to be shaded.

Consider a region R whose outer boundary is represented by polygon P_0. If this region contains x islands P_1, P_2,..., P_x to be excluded from shading, the vertices of island or inner boundaries are ordered in an opposite direction to the vertices of the outer boundary. Thus the region R is represented by a path-connected point set [6] composed of simply-connected polygons P_0, P_1,..., P_x, where P_0 is the outer boundary polygon and P_1,...,P_x are islands.

If a polygon P is considered topologically, three types of vertices can be identified with respect to the Y-axis. A vertex V_i of P is called a peak if both $y(V_{i-1})$ and $y(V_{i+1})$ are less than $y(V_i)$ and is called a pit if both $y(V_{i-1})$ and $y(V_{i+1})$ are greater than $y(V_i)$ [3, 6]. These vertices are jointly called critical vertices, whereas vertices that are neither peaks nor pits are referred to as non-critical vertices (see figure 3a).

A polygon containing only two critical vertices, the ones with minimum and maximum y-coordinates, is called monotone [6]. An example of a monotone polygon is illustrated in figure 3b. Note that for any closed polygon, there exist as many peaks as pits [3].

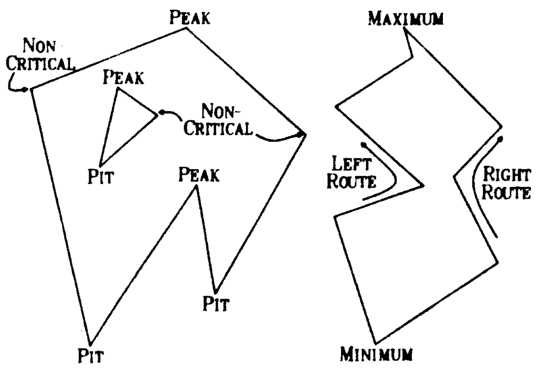

figure 3

(a) Peak, Pit and Non-Critical Vertices

(b) A Monotone Polygon

Following this terminology, a peak of type-1 is a peak of the inner region of R and a peak of type-2 is a peak of outer region of R. Similarly, a pit of the inner region of R is called a type-1 pit and a pit of the outer region of R is a type-2 pit (figure 4).

Furthermore, a polygon is called pseudo-monotone if it contains exactly two non-crossing, non-descending routes from its minimum vertex to its maximum vertex [9]. This notion is illustrated in figure 5, where a hypothetical region has been partitioned into mutually exclusive, pseudo-monotone polygons. A pseudo-montone polygon can be shaded in linear time.

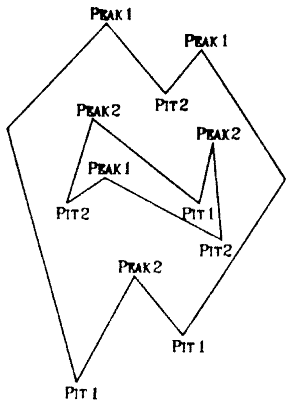

figure 4

Distinction between Type-1 and Type-2 Critical Vertices

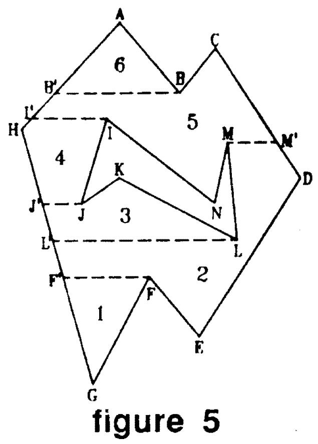

figure 5

Partitioning into Pseudo-Monotone Polygons

3. POLYGON PARTITIONING

Assume that shading lines are parallel to the x-axis, since if they are not, the polygons $P_0,...,P_x$ representing region R can always be rotated accordingly. Assume furthermore that none of the vertices have the same y-coordinates. Given these assumptions, the partitioning algorithm may now be presented.

In an initialization phase, the N vertices of all polygons $P_0...P_x$ are ranked according to their y-coordinates, the vertex with minimum y-coordinate having a rank of 1. Then, all vertices of R are classified as being peaks, pits or non-critical vertices. Furthermore, during this process a distinction is made between type-l and type-2 peaks and pits (Program A). Note that during the ranking process, vertices that have the same y-coordinates are naturally separated by assigning consecutive ranks to the vertices. Therefore, the second assumption above may be relaxed.

If region R is now considered to be a piece of paper (possibly with holes cut into it), then using a pair of scissors, it would be easy to 'cut away' pseudo-monotone polygons (pmp's) from R in the following way: Start cutting at a critical vertex of type-2 in a horizontal direction until the end of the paper in that particular direction is reached. Repeat this process for all type-2 critical vertices. The result would yield a number of oddly shaped pieces of paper that are all pmp's (Program B). Note that by using this method, the number of pmp's extracted is :

$$n(pmp) = n(\text{type-2 critical points}) + 1 - n(\text{islands})$$

As an example, consider the region illustrated in figure 5. Vertices F, L, J, M, I and B are all type-2 critical vertices. The proposed cuts are also illustrated, yielding six pmp's. The actual cutting directions chosen here is of no interest as a change in direction would still result in six pmp's.

To present the partitioning process more precisely, consider the following programs:

```
PROGRAM A;
  Input
    A list of the N vertices of polygons $P_0...P_x$ representing region R (vlist).
  Output
    Lists of ranks, classifications and types of $V_i$, rank(.), class(.) and type(.)
    respectively.
  Functions
    PRED(.) and SUCC(.) denoting the predecessor and successor of vertices on the
    boundary sequence respectively.
  Procedures
   • SORT(.) that sorts a list in ascending order.
   • CNODE(from, to, use, new), a procedure that creates a node [new] on the line
     through points [from] and [to], [new] being the horizontal cut-off point of [use] on
     the line.
  Program block
    BEGIN
      (* determine ranks *)
         FOR i FROM 1 TO N DO
            rank($V_i$) := i;
         ENDFOR;
         SORT(rank); (* ranks list is sorted using $y(V_i)$ as keys *)
      (* determine classifications *)
      FOR i FROM 1 TO N DO
         IF $y(PRED(V_i)) < y(V_i)$ AND $y(SUCC(V_i)) < y(V_i)$
            THEN class($V_i$) := peak;
         ELSIF $y(PRED(V_i)) > y(V_i)$ AND $y(SUCC(V_i)) > y(V_i)$
            THEN class($V_i$) := pit;
         ELSE class($V_i$) := non-critical;
         ENDIF;
      ENDFOR;
```

```
(* determine types *)
    FOR i FROM 1 TO N DO
        IF class($V_i$) = non-critical THEN   type($V_i$) := nil;
        ELSE
            $V_i$ := $V_i$; Succ := SUCC( $V_i$ );
            Pred := PRED($V_i$);
            CNODE($V_i$, Succ, Pred, Test); (* See figure 6 *)
            IF class(i) = peak THEN
                IF x(Test) > x(PRED($V_i$)) THEN   type($V_i$) := type1;
                ELSE type($V_i$) := type2;
                ENDIF;
            ELSIF class(i) = pit THEN
                IF x(Test) < x(PRED($V_i$)) THEN   type($V_i$) := type1;
                ELSE type($V_i$) := type2;
                ENDIF;
            ENDIF;
        ENDIF;
    ENDFOR;
ENDPROGRAM.
```

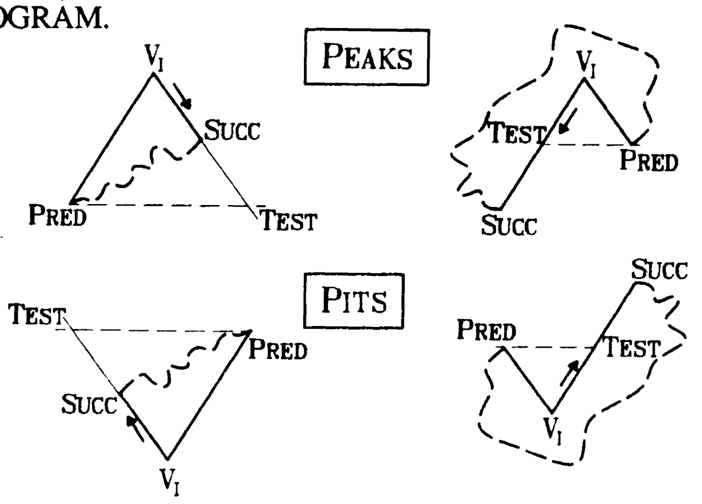

figure 6

Determining the Types of Peaks and Pits

PROGRAM B;

Input :
- A list of the N vertices of polygons $P_0 \dots P_x$ representing a region R (vlist).
- The lists of ranks, classes and types output by PROGRAM A.

Output
Two lists of vertices defining the left and right routes of a pmp (leftr and rightr respectively).

Storage
- Two lists, one containing all type-1 pits in ascending order of ranks (plist) and the other containing all type-2 pits in ascending order of ranks (qlist).
- A list of all type-2 peaks (rlist), as well as a list of markers associated with every type-2 peak (mark(.)).
- A stack with potential maximum size s equal to the total number of type-2 pits in R, where the stack is empty initially (vstack).
- Two temporary variables (Curl and Curr) representing vertices currently under consideration.

Functions
- MINI(a,b), returns the lower of two vertices [a] and [b].
- PRED(.), SUCC(.) and CNODE(...) as in PROGRAM A.
- TYPE2PEAK(x), true if vertex [x] is a type-2 peak.
- INLIST(list,point), true if vertex [point] is in [list].

Procedures

```
• PROCEDURE ADDLIST(list, point);
  BEGIN
      \code to add [point] at the end of [list]\
  ENDPROCEDURE;
• PROCEDURE CUTTOLEFT;
  BEGIN
      (* Cut to the left edge to extract a pmp *)
      CNODE(PRED(Curl), Curl, Min, New);
      ADDLIST(leftr, New);
      mark(Min) := New; (* Associate Min with New *)
  ENDPROCEDURE;
• PROCEDURE CUTTORIGHT; (* Cut to the right edge *)
  BEGIN
      CNODE(SUCC(Curr), Curr, Min, New);
      ADDLIST(rightr,  New); mark(Min) := New;
  ENDPROCEDURE;
• PROCEDURE INSINUATE (x);
  BEGIN
      \code to check if insinuation occurs (see figure 7)  All elements in qlist with
      ranks smaller than x is  possible insinuation points\
      IF\insinuation occurs\THEN
          \code to remove the insinuation point (Inpnt) from  qlist as a type-2 pit can
          'insinuate' only once\
          CNODE(PRED(Curl), Curl, Inpnt, New);
          ADDLIST(leftr,  New); ADDLIST(leftr, Inpnt);
          \code to push New and  Inpnt onto vstack\
          Curl := SUCC(Inpnt);
      ELSE
          IF flag THEN Curl := SUCC(Curl)
          ELSE Curr := PRED(Curr);
          ENDIF;
      ENDIF;
  ENDPROCEDURE;


• PROCEDURE XTRACT;
  BEGIN
      LOOP
          Min := MINI(Curl, Curr);
          IF Curl = Min THEN  flag := true; ADDLIST(leftr, Min);
          ELSE  flag := false; ADDLIST(rightr, Min);
          ENDIF;
          IF TYPE2PEAK(Min) THEN
              IF INLIST(rlist, Min) THEN
                  \code to remove Min from rlist\
                  IF flag THEN CUTTORIGHT;
                  ELSE CUTTOLEFT ENDIF;
                  EXITLOOP;
              ELSE (* Peak has already been used to cut from *),
                  New := mark(Min);
                  IF flag THEN  ADDLIST(leftr, New); Curl := SUCC(New);
                  ELSE  ADDLIST(rightr, New); Curr := PRED(New);
                  ENDIF;
              ENDIF;
          ELSE (* Min is not a type-2 peak *)
              IF Curl = Curr THEN EXITLOOP
              ELSE  INSINUATE(rank(Min)) ENDIF;
          ENDIF;
      ENDLOOP;
  ENDPROCEDURE;
```

```
Program block  (Execute for every pmp to be extracted)
    BEGIN
        IF NOT empty (plist) THEN
            leftr := rightr := empty;
            Curl := SUCC(plist(l)); Curr := PRED(PLIST(l));
            ADDLIST(leftr, plist(l)); ADDLIST(rightr, plist(l));
            \ code to remove the first element from plist \
            XTRACT;
        ELSIF NOT empty (vstack) THEN
            \ code to pop insinuation vertex Inpnt and point New \
            Curl := SUCC(New); Curr := PRED(Inpnt); XTRACT;
        ENDIF;
    ENDPROGRAM;
```

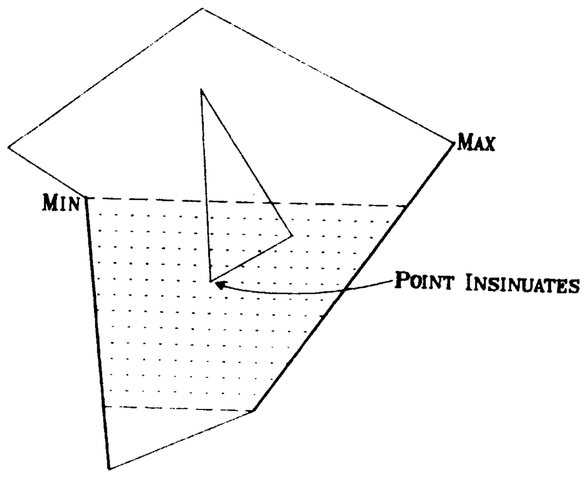

figure 7

An Insinuation of a Type-2 Pit Occurs if the Vertex Lies within the Coloured Area

4. POLYGON SHADING

The pseudo-monotone polygons are shaded in a way very similar to the technique used in [1,3] for shading trapezoids or triangles. The left and right routes that have been introduced in the discussion above are used as left and right shading limits. Instead of intersecting each shading line with these limits to determine the shading line end points, an incremental displacement along each shading edge is calculated.

Assume that apart from being parallel to the x-axis, all shading lines are also placed on integer y-axis values. (If this is not true, all vertices of polygons $P_0 \ldots P_x$ can always be scaled accordingly). Consider the left route of a hypothetical pmp as illustrated in figure 8a. Let L_i denote left end points of shading lines. Then by calculating $x(L_l)$ and $y(L_l)$ once for each shading limit, with repeated additions of d_x and d_y,

$$d_x = x(L_2) - x(L_1)$$
$$d_y = y(L_2) - y(L_1) = 1 \text{ (from the assumption),}$$

it is simple and fast to determine the other values of L_i : Let $DX=x(B)-x(A)$ and $DY=y(B)-y(A)$, then

$$y(L_1) = \text{CEILING}(y(A))$$
$$x(L_1) = x(A) + (y(L_1) - y(A))*DX/DY$$

and

$$d_x = DX/DY$$
$$d_y = 1.$$

Hence, the shading line end points are :

$$x(L_i) = x(L_{i-1}) + d_x$$

$y(L_i) = y(L_{i-1}) + 1,$
for $i = 2...COUNT$, where $COUNT = FLOOR(y(B)) - CEILING(y(A)) + 1$

A similar approach is taken to determine positions on a conceptual shading line for the placement of dots or symbols [8]. Using the parameters illustrated in figure8b (density, indentation and distance between consecutive shading lines), it is possible to construct many different dot and symbol patterns (see figures 9 and 10).

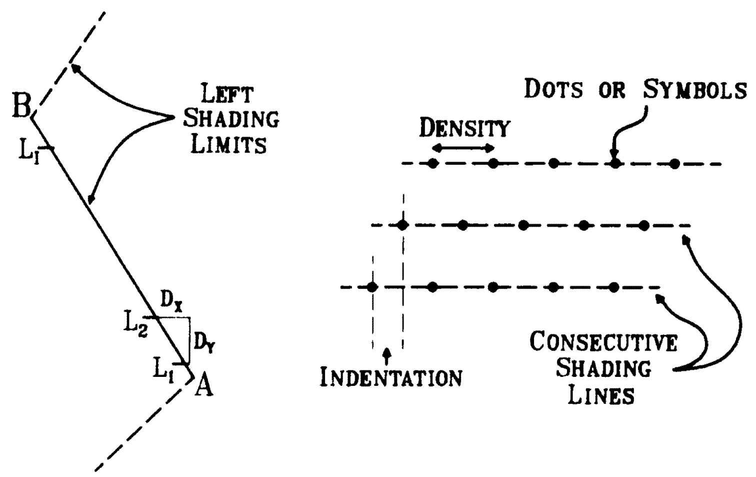

figure 8

(a) Determining Shading Line End Points

(b) Parameters to Control the Shading Using Dots or Symbols

5. IMPLEMENTATION

The shading technique described above has been implemented on a PERKIN-ELMER (3200-series) mini-computer in a locally developed procedural language called SCRAP. Using storage management facilities, it was possible to provide for regions with any number of islands and any number of vertices (only restricted by the storage available). This facility is required in a cartographic system currently being developed where geographical regions often contain a large number of islands, and the total number of vertices describing a region can be in the order of thousands.

The procedure first eliminates duplicate vertices on the boundaries of polygons representing the region, and then processes the island boundaries by reordering them in an opposite direction to the order of the outline boundary. Depending on the required parameters for shading, whether for shading lines or dot patterns, the boundary vertices are scaled and rotated so that conceptual shading line segments are always horizontal and fall on integer y-coordinates. Following this, the initialization phase requires a single O(n LOG n) sort on y-coordinates. These sorted coordinates are used to classify peak, pit and non-critical vertices of all polygons in an order O(n) process. The region to be shaded is then partitioned into pseudo-monotone polygons that can be shaded in linear time. Note that in the above discussion, no reference has been made to the fact that shading line segments should be rotated and scaled back to the original coordinate system. This is not necessary as the shading line end points can be calculated directly in the original coordinate system (see [8]).

Comparing this technique with the Brassel-Fegeas algorithm, it is obvious that the pmp partitioning algorithm is more effective. Apart from only requiring a single sort (as opposed to two sorts in the Brassel-Fegeas algorithm), this algorithm partitions a region into

N(pmp's) = N(type-2 pits and peaks) + 1 - N(islands).

The Brassel-Fegeas algorithm on the other hand, partitions the same region into N trapezoids, where

N(trapezoids) = N(vertices) - 1 + N(islands).

The reduction in the number of subparts that a partitioning process requires forms the strength of the new algorithm as the number of subparts may be considered as an overhead of the algorithm (see [6]).

Furthermore, the Brassel-Fegeas algorithm uses all outline vertices to test for insinuation, making it a very time intensive process. In the present algorithm, which uses the classifications of vertices, the test for insinuation is highly optimized as the vertices to be considered each time are limited.

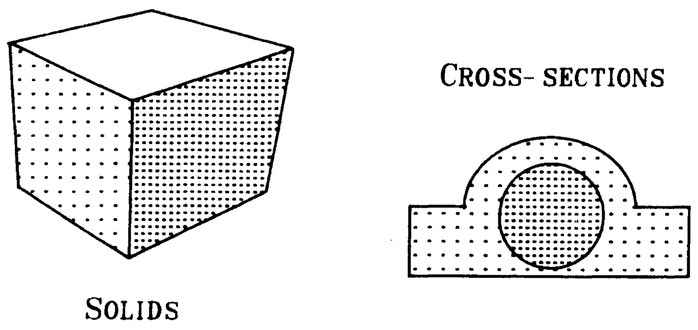

figure 9

Examples of Different Dot Pattern Shading

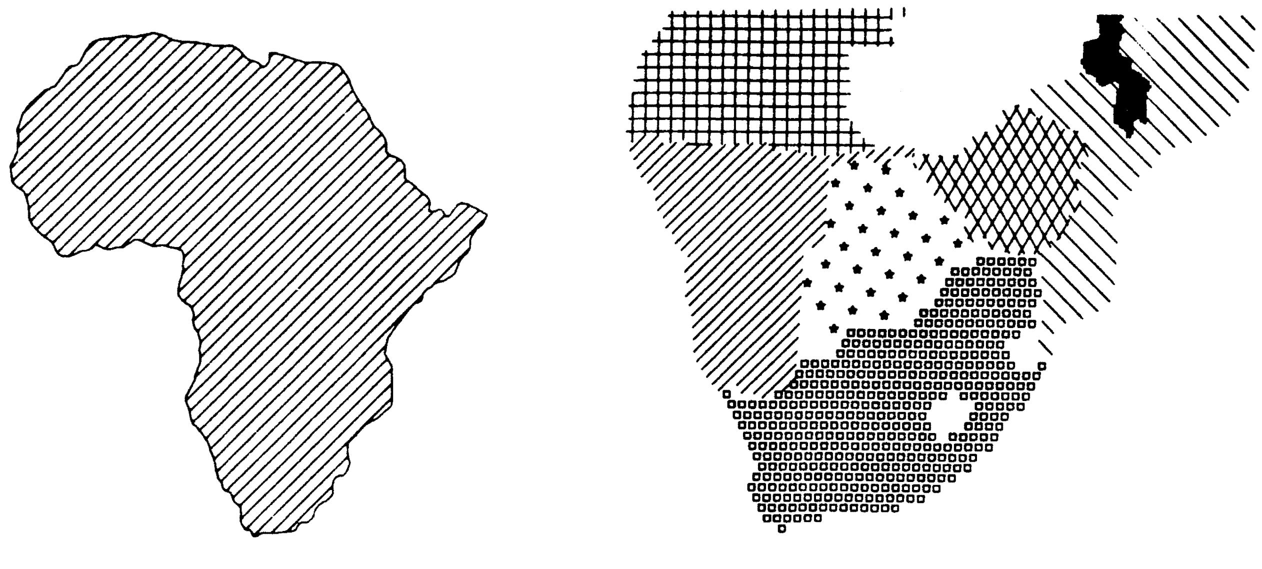

figure 10a

Examples of Results Obtained with the Pmp Partioning Algorithm

6. CONCLUDING REMARKS

An efficient technique for shading regions on vector type devices has been presented. This algorithm uses the topology of the region to partition it into less complex subparts that are then shaded by means of a fast procedure.

Using shading parameters, many different regular shading patterns may be constructed, making this technique very useful in a number of different computer graphics applications. Although the algorithm has been designed specifically for vector type devices, it can also be used for shading on raster type devices, where shading lines may be viewed as scan lines.

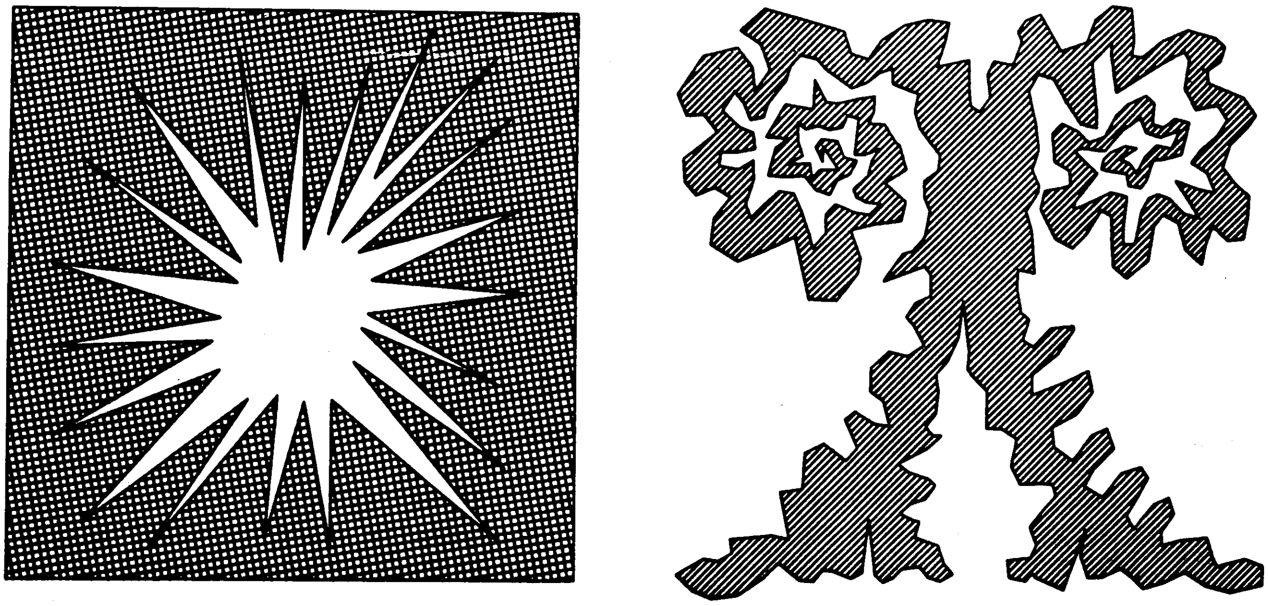

figure 10b

Examples of Results Obtained with the Pmp Partioning Algorithm

REFERENCES

1. BRASSEL,K.E. FEGEAS,R., [1979], An algorithm for shading of regions on vector display devices, *Computer Graphics*, **13**, 2, 126-133.
2. CARTLEDGE,C.J. WEEKS,G.H., [1985], Implementation of area fill for GKS, In : EARNSHAW,R.A. (ed.) *Fundamental algorithms for computer graphics*, NATO ASI Series F: Computer and Systems Sciences Vo1.17, Springer-Verlag, Berlin, 162-185.
3. CROMLEY,R.G., [1984], The peak-pit-pass polygon line-shading procedure, *The American Cartographer*, **11**, 1, 70-79.
4. FOLEY,J.D. VAN DAM,A., [1982], *Fundamentals of interactive computer graphics*, Addison-Wesley Publishing Company.
5. GAY.A.C., [1985], Experience in practical implementation of boundary-defined area fill, In : EARNSHAW,R.A. (ed.) *Fundamental algorithms for computer graphics*, NATO ASI, NATO ASI Series F: Series F: Computer and Systems Sciences Vo1.17, Springer-Verlag, Berlin, 153-160.
6. LEE,D.T., [1981], Shading of regions on vector display devices, *Computer Graphics*, **15**, 3, 33-44.
7. NEWMAN,W.M. SPROULL,R.F., [1979], *Principles of interactive computer graphics*, McGraw-Hill, Inc..
8. SCHEEPERS,C.F., [1986], *Veelhoekskakering in vektorgrafika*, NNWW Interne verslag, I679, Pretoria.
9. SCHEEPERS,C.F., [1986], Graphical communication and symbolism in a computerized cartography and map compilation system, In : BRADSHAW,J. WRENCH,K. (eds.) *First Computer Science research students conference*, Stutterheim, June 1986, (proceedings), Dept. of Computer Science, Rhodes University, August 1986, 133-154.
10. SHINDE,Y.N. MUDUR,S.P., [1986], Algorithms for handling the fill area primitive of GKS, *Computer Graphics Forum*, **5**, 105-117.
11. YING,D. FENG,X., [1985], Arbitrary area filling in a fast procedure, *Computer Graphics Forum*, **4**, 363-370.

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 artilces 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:

Prof. G. Wiechers
INFOPLAN
Private Bag 3002
Monument Park 0105
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. Manuscripts produced using the Apple Macintosh will be welcomed. Authors should write concisely.

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.

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 and the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Drawings etc., should be submitted and should include all relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

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. Ashcroft E. and Manna Z., The Translation of 'GOTO' Programs to 'WHILE' programs., *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255, 1972.
2. Bohm C. and Jacopini G., Flow Diagrams, Turing Machines and Languages with only Two Formation Rules., *Comm. ACM,* **9**, 366-371, 1966.
3. Ginsburg S., Mathematical Theory of Context-free Languages, McGraw Hill, NewYork, 1966.

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 minimize the risk of the author's contribution having to be held over to a later issue.

Only orginal 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.

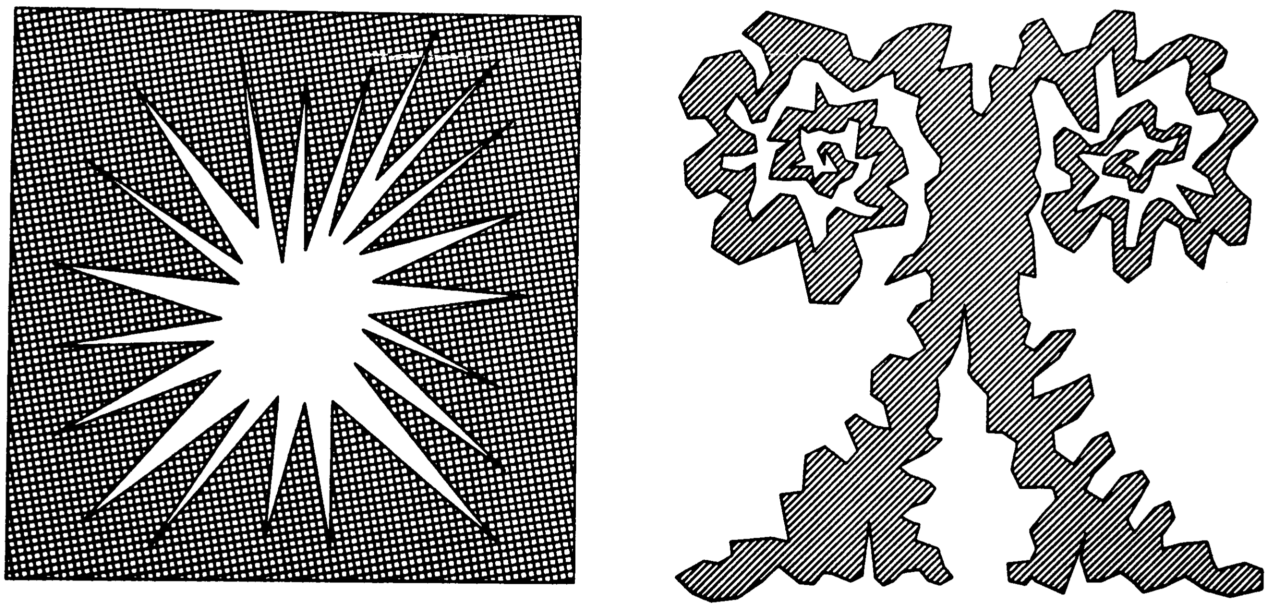

figure 10b

Examples of Results Obtained with the Pmp Partioning Algorithm

REFERENCES

1. BRASSEL,K.E. FEGEAS,R., [1979], An algorithm for shading of regions on vector display devices, *Computer Graphics,* **13**, 2, 126-133.
2. CARTLEDGE,C.J. WEEKS,G.H., [1985], Implementation of area fill for GKS, In : EARNSHAW,R.A. (ed.) *Fundamental algorithms for computer graphics*, NATO ASI Series F: Computer and Systems Sciences Vo1.17, Springer-Verlag, Berlin, 162-185.
3. CROMLEY,R.G., [1984], The peak-pit-pass polygon line-shading procedure, *The American Cartographer*, **11**, 1, 70-79.
4. FOLEY,J.D. VAN DAM,A., [1982], *Fundamentals of interactive computer graphics*, Addison-Wesley Publishing Company.
5. GAY.A.C., [1985], Experience in practical implementation of boundary-defined area fill, In : EARNSHAW,R.A. (ed.) *Fundamental algorithms for computer graphics*, NATO ASI, NATO ASI Series F: Series F: Computer and Systems Sciences Vo1.17, Springer-Verlag, Berlin, 153-160.
6. LEE,D.T., [1981], Shading of regions on vector display devices, *Computer Graphics*, **15**, 3, 33-44.
7. NEWMAN,W.M. SPROULL,R.F., [1979], *Principles of interactive computer graphics,* McGraw-Hill, Inc..
8. SCHEEPERS,C.F., [1986], *Veelhoekskakering in vektorgrafika*, NNWW Interne verslag, I679, Pretoria.
9. SCHEEPERS,C.F., [1986], Graphical communication and symbolism in a computerized cartography and map compilation system, In : BRADSHAW,J. WRENCH,K. (eds.) *First Computer Science research students conference,* Stutterheim, June 1986, (proceedings), Dept. of Computer Science, Rhodes University, August 1986, 133-154.
10. SHINDE,Y.N. MUDUR,S.P., [1986], Algorithms for handling the fill area primitive of GKS, *Computer Graphics Forum,* **5**, 105-117.
11. YING,D. FENG,X., [1985], Arbitrary area filling in a fast procedure, *Computer Graphics Forum*, **4**, 363-370.