AN ALGORITHM FOR SHADING OF REGIONS ON VECTOR DISPLAY DEVICES

Kurt E. Brassel, SUNY at Buffalo*
and
Robin Fegeas, U.S. Geological Survey**

ABSTRACT

The display of shaded polygons by line, cross-hatch, and dot patterns on vector devices is a task frequently used in computer graphics and computer cartography. In applications such as the production of shaded maps polygon shading turns out to be critical with respect to time requirements, and the development of efficient algorithms is of importance.

Given an arbitrary polygon in the plane without self-crossing edges (simply-connected polygon), the task at hand is to shade this polygon with one or two sets of parallel lines where for each set a shading angle and a line distance are given. The basic concept of this new algorithm is to decompose the polygon into a set of mutually exclusive trapezoids (in special cases triangles) where the parallel edges of the trapezoids are parallel to the desired shading lines. These trapezoids and triangles are then shaded in a fast procedure. In its present form the algorithm handles regions with up to 300 islands. Possible extensions include the construction of dash and cross patterns.

Key words and phrases: Software, computer graphics, polygons, shading, cartography, line-drawing processing, spatial information.

CR categories: 3.14, 3.24, 3.30, 3.79, 8.2

*Department of Geography, State University of New York at Buffalo, 415 Fronczak Hall, Amherst, NY 14260.

**Geography Program, U.S. Geological Survey, M.S. 710, Reston, VA 22092

1. INTRODUCTION

The shading of regions by a regular line pattern is a common task in various applications of computer graphics. Regions (polygons) as defined by a sequence of outline coordinates (x1,y1) ... (xn,yn) are to be shaded by sets of parallel lines, where the line width, the line distance, and the angle of shading orientation are commonly used as shading parameters. Such a routine allows for the construction of shaded line patterns of any shading density. Tobler [21] has proposed a continuous shading scheme for mapping which shades geographic areas by grey tones which are visually proportional to a geographic variable to be displayed. In practice, however, such a scheme encounters some technical and perceptual restrictions for the construction of very light or very dark tones: The human eye is not able to accurately perceive grey tones when shading-lines are closely spaced, and in the reproduction process such sets of lines may be modified in an undesirable way; also, parallel line shading is not very effective for light grey tones, where the line distance is large as compared to the line width or to the size of the region to be displayed. In these instances extended shading schemes such as cross-hatching (for dark grey tones) and dash, cross, or point patterns (for light tones) are preferably used. Brassel and Utano [3] have developed a scheme to create grey tones by continuously varying patterns from a coarse dot pattern to cross-hatch and solid-fill shading. Where this procedure fulfils the basic graphic needs, more efficient algorithms for general area shading would be desirable.

The basic module required is a routine which generates parallel sets of full and dashed lines. By appropriate combination of dash length, dash spacing, and rotation such a routine can be used for the generation of cross and point patterns as well. The use of various colors further extends the potential of such a scheme.

The literature on area shading in a raster environment is extensive [5,7,9,10,13,16], but is only of secondary importance to our problem. Various vector type area shading routines are in use in a broad range of applications [1,4,6,8,12,14,15,20,22,24,25]. In several cases, however, an explicit description of the underlying algorithm is missing. It is assumed that several of the above authors use the general polygon shading scheme which is discussed by [4] and [12].

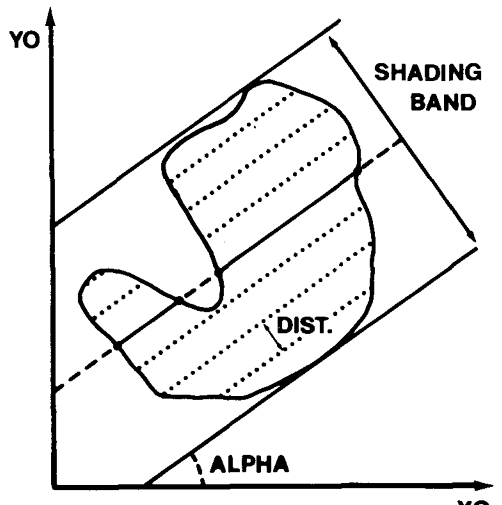


Figure 1

Given a polygon, a rotation angle and a line spacing, this algorithm first determines the bounding rectangle of the polygon parallel to the direction of the shading lines (Fig. 1). Within this rectangle shading lines are sequentially processed by computing their intersections with all segments of the polygon outline. Valid intersections of the outline segments and one shading line are sorted and pairwise connected. Coleman, et al.[4] call this algorithm the 'segment-polygon-algorithm'. Note that adjacencies between outline segments are of no consideration in this algorithm. For a polygon of N vertices which is to be shaded with M shading lines this algorithm considers N*M line intersections and uses M minor sorts.

Considering applications in cartography with thousands of polygons and typical values for M and N in the hundreds, it is evident that this traditional algorithm (its complexity is of order O(N*M)) is not efficient enough. Alternatives are presented by [4], [20], and [26]. Coleman et al. [4] achieve major improvements by first stripping the polygon into bands (a window parallel to the shading lines), while applying the segment-polygon-algorithm to the intersection boundary of the window and the polygon outline. Efficiency is improved due to a reduction of the quantity of points per intersection boundary. Implicitly this algorithm is equivalent to a 2-step recursive use of the segment-polygon-algorithm. Wood [26] sorts the polygon outline points into a linked list. He then sequentially computes the shading lines by keeping an updated list of active boundary segments. His algorithm minimizes core memory. Tobler [20] keeps track of all ups and downs of the polygon outlines, i.e. he makes use of information pertinent to outline segment adjacencies. Efficiency is thus improved, but the fact that he stores all segment-shading line intersections makes this program excessive for polygons of large sets of points and/or shading lines. All the algorithms discussed so far have in common that they use two rotation steps: rotation of the polygon vertices into a coordinate system parallel to the shading lines, and a rotation of the shading line endpoints back to the original system.

This paper presents a new and fast algorithm: it uses adjacency information of the outline segments and breaks the polygon down into trapezoids and triangles prior to shading; it uses only one rotation process.

2. BASIC CONCEPTS OF THE NEW ALGORITHM

a) Notations and Data Arrays

Given is a polygon in the plane defined by a string of outline vertices P(xo,yo) as measured in a coordinate system SO (compare Fig. 2); given is also a desired line spacing DIST and an orientation angle ALPHA. Elements of the outline string defined by two adjacent vertices are called outline segments. Define a coordinate system S in the plane where the origin of S coincides with the origin of SO, where DIST is used as unit length, and where the coordinate axes are rotated by ALPHA. Compute the coordinates P(x,y) of all polygon outline points with respect to S (rotation by ALPHA, scaling by DIST). Sort the array of P's in x and y directions (independently) and assign a rank to each vertex with respect to both the x- and y- axes (linked lists of x-ranks, y-ranks). Also assign pointers to the next vertices in x and y directions relative to each outline vertex. Table 1 illustrates the various arrays stored for the polygon in Figs. 2 and 3. Based on this data structure the polygon is now subdivided into trapezoids (in special cases triangles), and the trapezoids are independently shaded.

Table 1

OUTLINE SEQUENCE	A	B	C	D	E	F	G
X-RANK (XR)	1	3	4	7	6	5	2
Y-RANK (YR)	5	7	6	3	1	4	2
XO/YO COORDINATES	.	.	.	.	.	.	.
X/Y COORDINATES	.	.	.	.	.	.	.
NEXT X	G	C	F	.	D	E	B
NEXT Y	C	.	B	F	G	A	D

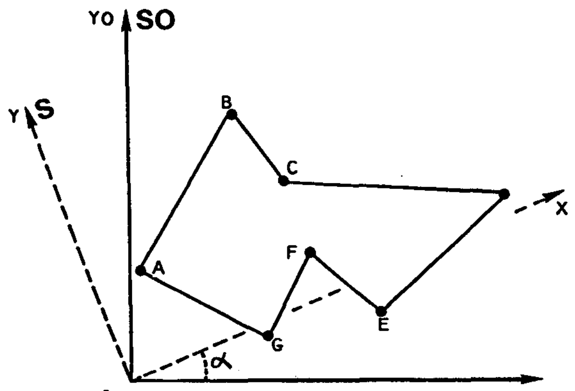


Figure 2

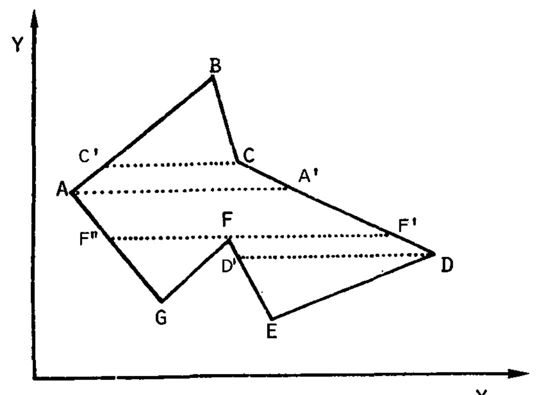


Figure 3

b) Trapezoid Extraction

The process of subdividing the polygon into trapezoids and triangles is initiated at the point of lowest rank in a particular direction: Assume we want to shade the polygon in Fig. 3 with lines parallel to the x-axis. As our starting point we then select the vertex with the minimum y-coordinate, i.e. point E (compare also with Table 1). We further define the two outline edges EF and ED as left (segment clockwise of E) and right (segment counterclockwise of E) shading limits. Shading limits will be constantly updated. In the starting phase these shading limits are

left	right
EF	ED

From the semi-bounded region DEF a triangle ED′D is extracted for shading. Since y-rank(D) < y-rank(F), D′ is defined such that y(D′)=y(D) and E,D′,F collinear. DD′ is called the upper limit of the triangle to be shaded. At this point, however, it is not known whether the triangle ED′D can be shaded entirely or whether there is some insinuation point P within the triangle, as this is indicated in Fig. 4. To find a potential insinuation point P a point-in-trapezoid (triangle) search is performed. This search makes use of the rank pointers defined above (x-rank: XR(i), y-rank: YR(i)).

For all points P:
YR(E) ---> YR(D) check for XR(F) < XR(P) < XR(D).

A point P which fulfils this condition is located within a window defined by E,D (y-direction) and

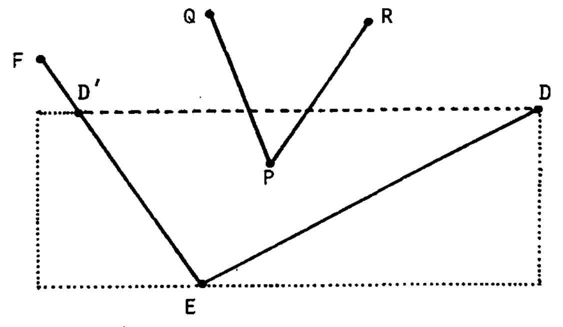


Figure 4

D,F (x-direction). Such a point P has to further be tested against the left and right limits of the active trapezoid (triangle). In the illustration example in Fig. 3 obviously no point P fulfils this condition. Therefore, the triangle ED′D can be subjected to the shading procedure as explained in section 2c.

In the next step the shading limits are updated as follows:

left	right	update	left	right
EF	ED	------>	D′F	exhausted

Since the right shading limit is exhausted, we retrieve a new segment from the polygon outline. In our example (Fig. 3) this is DC, i.e. the segment next to ED in the counterclockwise outline sequence. The shading limits are now defined as:

left	right
D′F	DC

Since YR(F) < YR(C) define the trapezoid D′FF′D with F′ such that y(F′)=y(F) and D,F′,C collinear. Since no further point is within D′FF′D this trapezoid is being shaded. Updating of the shading limits results in:

left	right
exhausted	F′C

To replace the exhausted left shading limit retrieve the next clockwise outline segment FG. Since YR(G) < YR(F), however, the segment FG cannot be a left shading limit, but rather a right shading limit of some other trapezoid. Therefore, search along the clockwise outline sequence for a local minimum; use its two adjacent segments as the two shading limits; and push the old shading limit(s) down into a stack. Since G is a local minimum the updated shading limits are as follows:

left	right
GA	GF
—	F′C (stack).

In the subsequent step F" will be defined and the triangle GF"F will be shaded. Since GF is exhausted segment F′C is brought back from the stack:

left	right
F"A	F′C

It can now easily be seen that the shading sequence for the remainder of the polygon is:

F"AA′F′ AC′CA′ and C′BC.

So far we have not discussed the case where an insinuation point P is located within a preliminary trapezoid (figure 5). The shading limits in this situation are:

left	right
ED′	ED
D′F	—

and a point P is found to be within ED′D. Eliminate point D′, shade the triangle EP′P" and update the shading limits as follows:

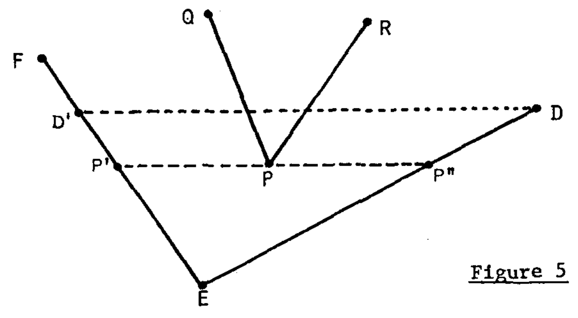


Figure 5

left	right
PR	P"D
P'F(stack)	PQ (stack).

The polygon is thus broken down into a left and right branch, where the right branch is pursued first (PR,P"D), while the left branch (P'F,PQ) is kept in the stack until the right branch is exhausted (Fig. 6a). The stack may contain two types of records: (1) single left or right shading limits for which opposite shading limits have not yet been detected, and (2) pairs of opposite shading limits which initiate a new polygon branch. It is reasonable to keep these two types separate in a primary (1) and a secondary (2) stack.

If point P is the bottom point of an island, the right branch shading is performed until the top of the island (T) is reached (Fig. 6b). At this point the search for a valid left shading limit follows the left island boundary to find a local minimum. Along this search path P is recognized as the bottom of an island; this is the signal to recall the left branch from the secondary stack for further processing.

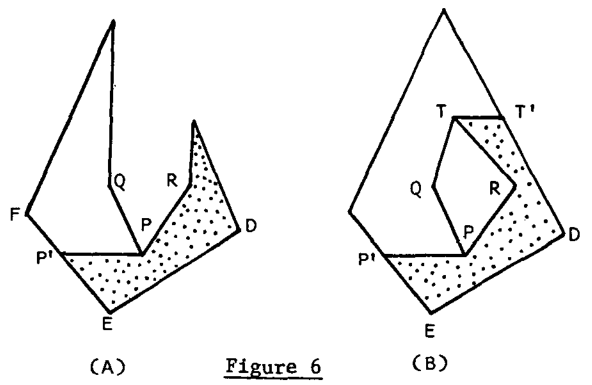


Figure 6

c) Trapezoid Shading

Given a trapezoid ABB'A'(Fig. 7) which is to be shaded by lines parallel to AA' and BB' (AA' and BB' are parallel). The coordinates of these points are known in both coordinate systems SO (P(xo,yo)) and S (P(x,y)), where the shading lines are to be drawn parallel to the x-axis of the S system. We assume the measurement units in the S system to be equal to the line spacing DIST; then the y-coordinates rounded to integer values equal to

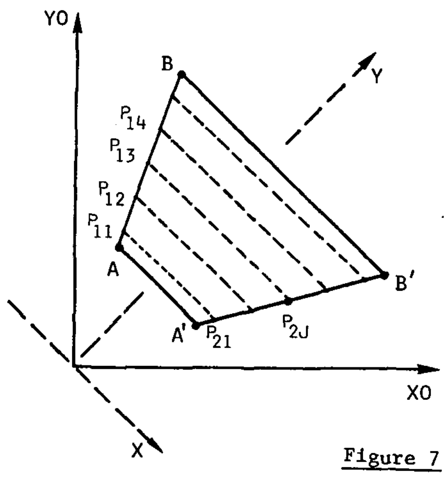


Figure 7

the ID-number of the shading lines where line #0 cuts through the system origin. The task at hand is to compute the endpoints P_{ij} of the set of shading lines within the trapezoid ABB'A'. Given the y-coordinates of points A and B the endpoints P_{1j} are located on AB at integer coordinate locations of the y-axis. The y-coordinate of P_{11}, for example, equals to

$$y(P_{11}) = \text{IFIX} (y(A) + .99999)$$

and the ratio R1 = AP_{11}/AB can be expressed as

$$R1 = (y(P_{11}) - y(A)) * YAB,$$

where YAB = 1./(y(B) - y(A)). Define the ratio R2 as 'one shading line increment as a ratio of the distance AB ':

$$R2 = (y(P_{12}) - y(P_{11})) * YAB = YAB$$

R1 is used to compute the coordinates of the first shading line endpoint P_{11}, where its computation is directly performed in the original coordinate system SO:

$$xo(P_{11}) = xo(A) + R1 * DX$$

$$yo(P_{11}) = yo(A) + R1 * DY,$$

where DX= xo(B) - xo(A) and DY = yo(B) - yo(A). Notice that the above coordinates are measured in the SO system, so that no rotation is required here. The increment between subsequent line endpoints in the SO system is

$$DXO = DX * R2$$
$$DYO = DY * R2.$$

The coordinates of the shading endpoints are thus

$$xo(P_{1j}) = xo(P_{1,j-1}) + DXO$$

$$yo(P_{1j}) = yo(P_{1,j-1}) + DYO,$$

where

$$j = 2 \ldots k$$
$$k = \text{IFIX} (y(B)) - y(P_{11}) + 1.$$

The shading line endpoints P_{2j} on A'B' are computed in analog fashion, the points P_{1j} and

P_{2i} are pairwise connected and the lines plotted. The computation of the end points of the first shading line requires thus 14 additions and 16 multiplications, where for each subsequent shading line 4 additions are needed. A necessary condition is that the resolution of the binary coordinate representation be higher than the distance between consecutive shading lines.

3. IMPLEMENTATION AND EVALUATION

The algorithm as described has been encoded in FORTRAN IV and implemented on an IBM 370/155. An overview of the procedure is given in Fig. 8. First, duplicate adjacent polygon outline points are eliminated. Further, polygons consisting of various islands (up to 299 islands are allowed) are preprocessed (outline strings are identified as islands). The outline points are then rotated and scaled (xo,yo ---> x,y). The transformed values are sorted in both x- and y-directions, and duplicate points which are non-adjacent in the string sequence are separated by a small distance. These preprocessing steps construct the data arrays as shown in table 1. The actual shading procedure, i.e. the decomposition into trapezoids and triangles is then performed in a single or double call (cross-hatching) to a procedure called TRAPEZ. Note that for orthogonal cross-hatching the preprocessing steps have to be performed only once (re-use of the values in Table 1, with x and y reversed). After completion of the shading, control is returned to the calling procedure for handling of a next polygon.

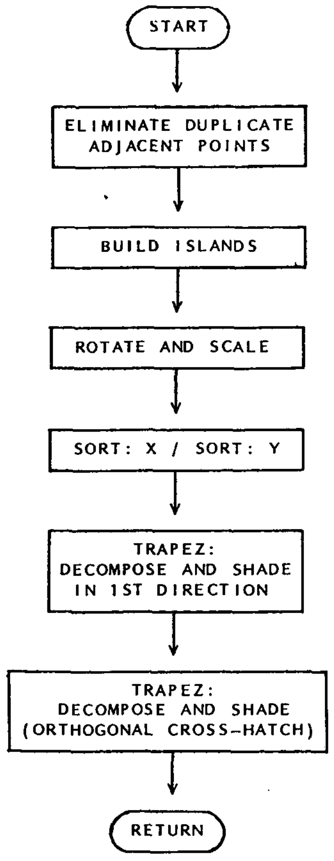

START
ELIMINATE DUPLICATE ADJACENT POINTS
BUILD ISLANDS
ROTATE AND SCALE
SORT: X / SORT: Y
TRAPEZ: DECOMPOSE AND SHADE IN 1ST DIRECTION
TRAPEZ: DECOMPOSE AND SHADE (ORTHOGONAL CROSS-HATCH)
RETURN

FIGURE 8

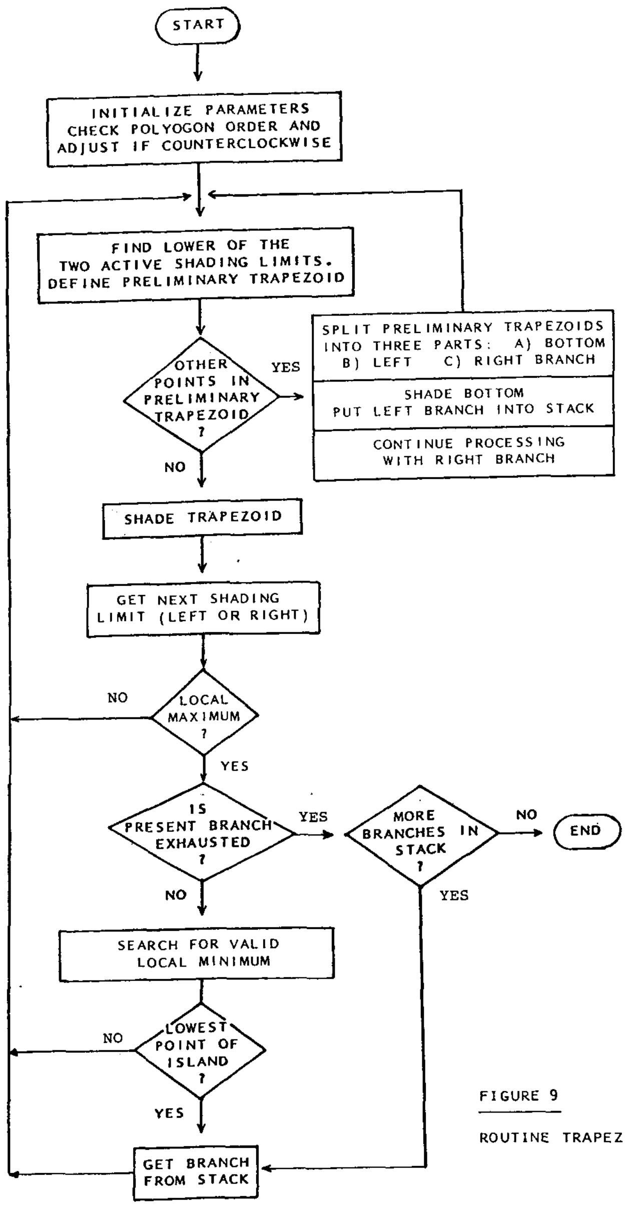

START
INITIALIZE PARAMETERS CHECK POLYOGON ORDER AND ADJUST IF COUNTERCLOCKWISE
FIND LOWER OF THE TWO ACTIVE SHADING LIMITS. DEFINE PRELIMINARY TRAPEZOID
OTHER POINTS IN PRELIMINARY TRAPEZOID ?
YES
NO
SPLIT PRELIMINARY TRAPEZOIDS INTO THREE PARTS: A) BOTTOM B) LEFT C) RIGHT BRANCH
SHADE BOTTOM PUT LEFT BRANCH INTO STACK
CONTINUE PROCESSING WITH RIGHT BRANCH
SHADE TRAPEZOID
GET NEXT SHADING LIMIT (LEFT OR RIGHT)
LOCAL MAXIMUM ?
NO
YES
IS PRESENT BRANCH EXHAUSTED ?
YES
NO
MORE BRANCHES IN STACK ?
NO
YES
END
SEARCH FOR VALID LOCAL MINIMUM
LOWEST POINT OF ISLAND ?
NO
YES
GET BRANCH FROM STACK

FIGURE 9

ROUTINE TRAPEZ

The procedure for decomposition and shading (TRAPEZ) is illustrated in Fig. 9. First, the order of the polygon (clockwise, counterclockwise) is determined and some related parameters are initialized. The low point and the first two shading limits are established, and a first preliminary triangle (trapezoid) is defined. This preliminary triangle is tested for inclusion of any other outline point. If no point is within the trapezoid, it is being shaded, and a new valid shading limit (left or right) is searched for. The new pair of active shading limits is used again to define a new preliminary trapezoid. If no valid new adjacent shading limit can be found -- be it that the shading branch is exhausted or that the low point of the newly found shading limit is a member of the secondary stack — the present branch is abandoned and a new branch is retrieved from the secondary stack. If the search for an outline point within a preliminary trapezoid is successful, the preliminary trapezoid is split into three parts: (a) the bottom part (below the newly found point) is shaded and eliminated from the list, (b) the left shading branch is entered into the secondary stack, and (c) the right branch is used for immediate processing, i.e. its shading limits define the next preliminary trapezoid. The procedure terminates when all shading branches are exhausted.

The basic procedure consists of iterations of (a) the definition of preliminary trapezoids, (b) the test for inclusion of outline points within the preliminary trapezoid, (c) trapezoid shading, and (d) the search for new valid shading limits, where for each outline point within a preliminary trapezoid a branch is put on the stack, and each exhausted branch is replaced from the stack.

Figure 10 illustrates the shading of a complex polygon, where triangles and trapezoids are labelled in their processing sequence. The shading procedure may be compared to the filling of an arbitrarily shaped container with water, where the influx pipe is at the lowest point of the container (MIN). The liquid first fills the volume labelled 1 (Fig. 10) and then flows over into area 2. Whenever the water level reaches some insinuation point propagating from the top (TP), then the filling process is artificially interrupted to the left of the propagating point. The blocking of the left branches is achieved in the program by using the secondary stacks. These blockages are re-opened when either the right branch is entirely filled, or if the water overflows the top of the island.

The procedure as implemented keeps all arrays in core memory and is thus programmed for time efficiency. The stoarge requirements for this version are approximately 15K (or 22.6K if not overlaid) + 16N + 8NI + 32NS bytes, where N is the maximum number of outline points in any single polygon, NI is the number of islands, and NS is the number of shading limit stacks.

For a polygon with N outline points which is to be shaded by M parallel lines, traditional segment-polygon-algorithms involve as their major operations M*N computations of line intersections, at least N + 2M rotations, and M minor sort steps. For cross-hatching these values are doubled. The order of operations performed in the various parts of the trapezoid algorithm is given in Table 2.

The present algorithm requires two sorts (O(N log N)), the creation of NT = N - 1 + NI trapezoids (NI = number of islands), and the shading of NT trapezoids. The sort steps do not have to be repeated for orthogonal cross-hatching. Non-linear behaviour of the present algorithm is observed with the elimination of duplicate points, the sorts, and the search for points within the preliminary trapezoids. This search for points within a preliminary trapezoid appears to be the most critical step. As a characteristic index for the polygon we use the "number of lobes", i.e. the ratio of the y-range of all trapezoids divided by the polygon height. A convex polygon would have a lobe number NL = 1 where for a polygon consisting of two external islands of the same height NL = 2; the polygon in Fig. 10 has a lobe number NL = 3.03. For maximally dissected polygons NL would approach N/2. In these extreme cases the number of points consulted in the insinuation test approaches $N^3/2$. However, this has no practical impact since most of the points can be immediately eliminated by a simple check against the two x-limits of the preliminary trapezoids; further, all points which are not local minima are eliminated as well, so that the overall number of point-in-trapezoid checks for the entire polygon is smaller than the number of local minima NLM in the polygon (NLM < N/2). This point-in-trapezoid test is thus linear.

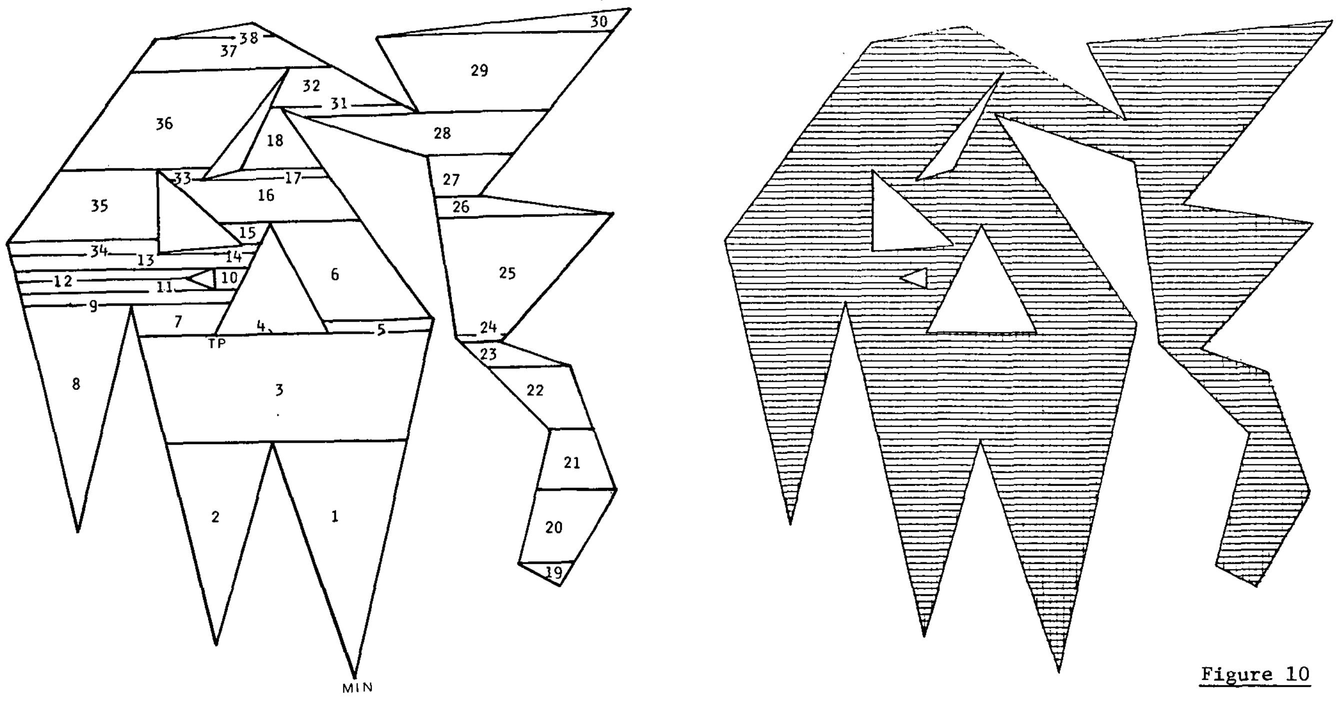


Figure 10

OPERATION	COMPLEXITY
Eliminate duplicate points	N * NDP
Build island	N
Rotate and scale	N
Sort and separate	N log N
Initialize trapez routine	C
Define preliminary trapezoids	NT
Search for points within prelim. trap.:	
a) Search for all points within y-range of all trapezoids	NT*[1+(NL-1)*NL]
b) Point-in-trapezoid search	< NLM
Split preliminary trapezoid	NIP
Polygon shading:	
a) Initialization + first line (all trap.)	NT
b) Additional lines (all trap.)	NL*Hp/SP-NT
Get next shading limit	NT
Search for valid local minima	<N
Retrieve branches from stack	<NLM

N	=	Number of outline points in polygon
NI	=	Number of internal islands in polygon
NE	=	Number of external islands in polygon
NDP	=	Number of duplicate adjacent points
C	=	Constant
NT	=	Number of trapezoids = N + NI - NE
NL	=	Number of lobes in polygon = $\sum_{i=1}^{NT} Hi/Hp$
NIP	=	Number of insinuation points (<N/2-1)
NLM	=	Number of local minima = number of local maxima (<N/2)
Hi	=	Height of the ith trapezoid (y-range)
Hp	=	Height of the polygon (y-range)
SP	=	Line spacing for polygon shading

Table 2. Number of Operations used in Trapezoid Shading.

The overall performance of the algorithm depends basically on the line spacing, the lobe index, the number of outline points and the number of islands. Empirical results comparing the trapezoid method with the segment-polygon algorithm are shown in Table 3. Test 1 and 2 measure the total execution time for polygon shading, whereas tests 3 through 8 record only the actual shading work — excluding input and plotting. Fig.11 graphically displays the major findings related to tests 3 - 8 above. The results suggest that the major strength of the trapezoid algorithm lies with high density shading, i.e. the shading of polygons for which the ratio

R = M / N (M = # of shading lines per polygon)

is high. Where both the sort and the trapezoid extraction steps are insensitive to M the shading of the trapezoid heavily depends on the line density. For polygons with an average of 12 outline points the present algorithm is better if R>.3 (i.e. if at least 3.6 shading lines are drawn), for polygons with more points the critical values of R is expected to be lower. Further, the efficiency of both algorithms depends on the number of outline points in a polygon. The trapezoid algorithm, however, is less sensitive to large numbers of outline points. In absolute terms ('CPU total') the trapezoid shading algorithm performs significantly better than the segment-polygon-algorithm, exept in the tests 3 and 4 where both the number of outline points and the number of shading lines are small (light shading of simply-shaped polygons).

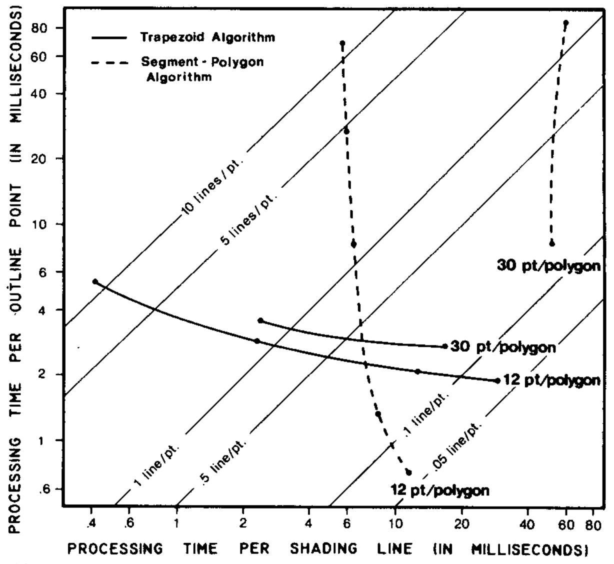


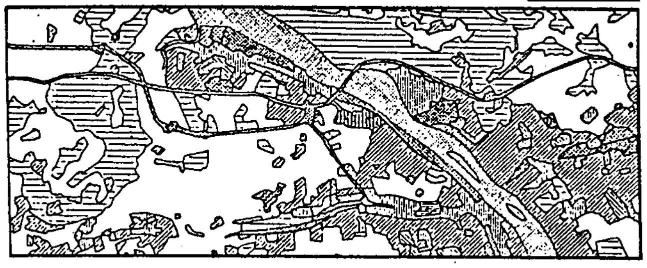
Figure 11

#	# OF POLY.	# OF PTS.	# OF LINES	PTS./ POLY.	LNS / PTS.	CPU tot.	CPU/ PT.	CPU/ LINE	ALG
1	1822	44540	6987	24.44	.16	143.	3.21	20.47	TRP
						176.	3.95	25.19	SEG
2	3886	169109	22916	43.51	.14	617.	3.64	26.92	TRP
						1284.	7.59	56.03	SEG
3	152	1786	109	11.75	.06	3.22	1.80	29.54	TRP
						1.27	0.71	11.65	SEG
4	152	1786	290	11.75	.16	3.73	2.09	12.86	TRP
						2.38	1.33	8.21	SEG
5	152	1786	2254	11.75	1.26	5.18	2.90	2.30	TRP
						14.36	8.04	6.37	SEG
6	152	1786	22129	11.75	12.4	9.38	5.25	.42	TRP
						128.6	72.0	5.81	SEG
7	633	18842	3005	29.77	.16	51.41	2.73	17.10	TRP
						157.0	8.35	52.27	SEG
8	633	18842	28526	29.77	1.51	68.26	3.62	2.39	TRP
						1655.	87.8	58.03	SEG

Table 3. Results of Test Runs. Tests 1 and 2 measure total execution time, test 3 through 8 do not include input and plotting operations. CPU tot. gives the execution time in seconds on an IBM 370/155. CPU/PT and CPU/LINE are given in milliseconds.

The shading test runs were performed with a land cover map file with a multitude of islands (Fig. 12).

Figure 12



4. CONCLUSIONS

We have presented an algorithm for shading of polygons on vector display devices. This algorithm disaggregates the polygon into a set of triangles and trapezoids parallel to the direction of the shading lines. The basic structures used thus are similar to the 'slab' methods used in some point-in-polygon algorithms [17,18]. Similarities also exist with elements of scan type hidden line algorithms [2,11,19,23]. An implementation of the described algorithm on an IBM 370/155 has proven highly efficient. Where the present implementation is optimized with respect to execution time, future modifications will reduce storage requirements for the implementation on minicomputers. Planned extensions of the method include the shading by systematic dash or cross patterns. Further, future efforts will have to be directed towards a systematic comparative analysis of the present procedure and various other methods for vector type shading, including the potential of elements of scan type hidden line algorithms for area shading.

REFERENCES

1. Baxter, R.S., Choropleth Mapping Program By Computer, Manuscript, Building Research Establishment, Garstom, Watford, U.K., 26 pgs.
2. Bouknight, W.J., 1970, "A Procedure for Generation of Three-dimensional Half-toned Computer Graphics Representations", Communications ACM, Vol. 13, No. 9, pp. 527 - 536.
3. Brassel, K.E. and J.J. Utano, 1979, "Design Strategies for Continuous-tone Area Mapping", The American Cartographer, Vol.6, No.1 (forthcoming).
4. Coleman, P.R., R.C. Durfee, and R.G. Edwards, "Application of a Hierarchical Polygon Structure in Spatial Analysis and Cartographic Display", Harvard Papers on Geographical Information Systems, Vol. 3, 20 pgs.
5. Jarvis, J.F., C.N. Judice, and W.H. Ninke, 1976, "A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays", Computer Graphics and Image Processing, Vol. 5, pp.13-40.
6. Kern, H., 1978, "Neuere Techniken der Flaechenschraffur und der Herstellung von Farbauszuegen in der automatisierten thematischen Kartographie" (Recent Techniques of Area Shading and Color Separation in Automated Thematic Cartography), Kartographische Nachrichten, Vol. 28, No. 1, pp. 1-11.
7. Knowlton, K. and L. Harmon, 1972, "Computer-Produced Grey Scales", Computer Graphics and Image Processing, Vol. 1, pp. 1-20.
8. Laboratory for Computer Graphics and Spatial Analysis, 1976, CALFORM User's Manual. Cambridge: Harvard University.
9. Lieberman, H., 1978, "How to Color In A Coloring Book", Computer Graphics, Vol. 12, No. 3, pp. 111-116.
10. Negroponte, N., 1977, "Raster Scan Approaches to Computer Graphics", Computers and Graphics, Vol. 2, pp. 179-193.
11. Newman, W.M. and R.F. Sproull, 1973, Principles of Interactive Computer Graphics. New York: McGraw-Hill.
12. Monmonier, M.S., and D.M. Kirchoff, 1977, "Choroplethic Plotter Mapping for a Small Minicomputer", Proceedings of the American Congress on Surveying and Mapping, 37th Annual Meeting, Wash., D.C., pp. 318-338.
13. Pavlidis, Th., 1978, "Filling Algorithms for Raster Graphics", Computer Graphics, Vol. 12, No. 3, pp. 161-164.
14. Rase, W.D., SRAFOF shading subroutine, private communication.
15. Rase, W.D., 1978, "Computer-assisted Thematic Mapping for Federal Planning", Nachrichten aus dem Karten- und Vermessungswesen, Series II: Translations, No. 35, pp. 77-83.
16. Rosenfeld, A. and V.C. Kak, 1976, Digital Image Processing. New York/London: Academic Press.
17. Salomon, K.B., 1978, "An Efficient Point-in-Polygon Algorithm", Computers and Geosciences, Vol. 4, pp.173-178.
18. Shamos, M.I., 1977, Computational Geometry. Berlin/New York: Springer-Verlag.
19. Southerland, I.E., R.F. Sproull and R.A. Schumacker, 1974, "A Characterization of Ten Hidden-Surface Algorithms", Computing Surveys, Vol. 6, No. 1, pp. 1-55.
20. Tobler, W.R., 1971, Choropleth Mapping Programs. Cartographic Laboratory Report No.6, Dept. of Geography, Univ. of Michigan, Ann Arbor.
21. Tobler, W.R., 1973, "Choropleth Maps without Class Intervals?", Geographical Analysis,Vol. 5, pp. 262-265.
22. U.S. Geological Survey, Geography Program: Routine SHADT, personal communication.
23. Watkins, G.S., 1970, A Real-Time Visible Surface Algorithm, Computer Science Department, University of Utah, UTECH-CSc-70-101.
24. Waugh, T.C., and D.R.F. Taylor, 1976, "GIMMS / An Example of an Operational System for Computer Cartography", The Canadian Cartographer, Vol. 13, No. 2, pp. 158-166.
25. Wood, P.M., and D.M. Austin, 1975, "CARTE: A Thematic Mapping Program", Computers and Graphics, Vol. 1, No. 2/3, pp. 239-250.
26. Wood, P.M., 1978, "Interactive Display of Polygonal Data", Harvard Papers on Geographic Information Systems, Vol. , 20 pgs.

ACKNOWLEDGEMENTS

This project has been supported by the Geography Program, U.S. Geological Survey. Mr. Mike Wasilenko and Ms. Karen Weiss have contributed to the various figures in this paper.