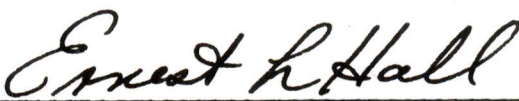
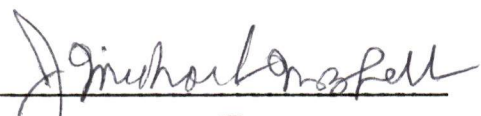
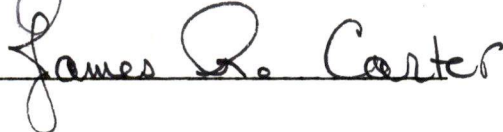


To the Graduate Council:

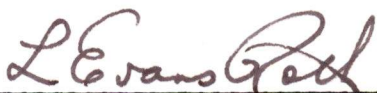
I am submitting herewith a thesis written by Edwin F. Abercrombie entitled "A System for the Construction of a Three-Dimensional Block Diagram Representation from Two-Dimensional Data." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.


Ernest L. Hall, Major Professor

We have read this thesis and
recommend its acceptance:

Accepted for the Council:


Vice Chancellor
Graduate Studies and Research

Thesis
78
.A247
Copy 2

A SYSTEM FOR THE CONSTRUCTION OF A THREE-DIMENSIONAL
BLOCK DIAGRAM REPRESENTATION FROM
TWO-DIMENSIONAL DATA

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Edwin F. Abercrombie

June 1978

1358397

Copyright by Edwin F. Abercrombie 1978
All Rights Reserved

ACKNOWLEDGMENTS

To Dr. Ernest L. Hall and Dr. James R. Carter, I would like to extend my thanks for their guidance and assistance in the writing of this thesis. Gratitude is further expressed to Dr. J. Michael Moshell for his helpful suggestions in preparing the final manuscript. Further gratitude is expressed to Mr. E. Gray Simons for his assistance and continual support throughout my graduate career. To my family, especially my parents, and friends, I am forever indebted for their continual support and encouragement throughout my education and particularly during the years spent on this study.

ABSTRACT

The purpose of this study was to develop an interactive computer program to draw three-dimensional block diagrams of discontinuous surfaces from two-dimensional data using parallel lines to represent the surfaces. Unlike all parallel-line or fishnet programs, this program is unique in that the surface area at the top of these polygons was the same as the base surface area of each respective polygon. This is different from the existing discontinuous surface plotting programs, where the top is smaller due to incremental plotting techniques of sloping lines. The construction of these polygon-base columns was accomplished by having two Z-values for each XY point. Hidden lines were removed by keeping track of the starting and ending points of all new visible vectors and comparing them with the visible portions of all vectors which had been plotted. By starting in the lower front corner, plotting from left to right, and keeping track of the top line, the program needed only to know when the current vector being plotted changed states, that is, whether it was visible or invisible.

DISCON, the FORTRAN IV program created for this study, was used to plot a coordinate set of the ninety-five counties for the State of Tennessee as an example. The height of the polygons, corresponding to the third dimension of the polygons or, in this case the counties, were 1970 population data. However, any two-dimensional surfaces (with any third dimensional values assigned) could be used. With the data set defined, evenly spaced parallel lines can be overlaid at either 0° , 45° ,

90°, or 135°. DISCON finds the intersections between each parallel line and each line segment of each polygon. The intersection of these diagonals and polygon line segments are saved in a file for further data manipulation. These data manipulations consisted of rotation of the data base file of intersection in a Y and Z rotation. DISCON next executes the hidden-line elimination portion of the algorithm, computes the viewing angle, and displays the results on the TEKTRONIX terminal and/or plotter. At this point, the user has the option to return to the original base file, enter new data, change the input parameter for the rotation in a X,Y,Z direction, and view the surface in a different representation.

The results presented in this thesis show that DISCON is a practical program to construct a three-dimensional block diagram of discontinuous surfaces from two-dimensional data using parallel lines to represent the surfaces.

TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION	1
Outline of Thesis	5
Discontinuous Three-Dimensional Surface Plotting	9
II. HIDDEN-LINE ELIMINATION	15
Object Space Algorithms	15
Image Space Algorithms	22
List-Priority Algorithms	27
III. DISCON: A SYSTEM FOR CONSTRUCTION OF A THREE-DIMENSIONAL BLOCK DIAGRAM REPRESENTATION FROM TWO-DIMENSIONAL DATA . .	30
User's Parameter Inputs	33
Coordinate Data Input	37
Compute the End Points of the Diagonals	39
Polygons and Diagonal Intersections	41
Sorting and Deletion of Duplicate Entries on Diagonals .	48
X - Y - Z Rotation	50
Hidden-Line Elimination Algorithm	56
Hardware and Software	59
IV. CONCLUSIONS	61
LIST OF REFERENCES	72
APPENDICES	75
A. GLOSSARY	76
B. USER'S GUIDE AND PROGRAM LISTING	81
VITA	113

LIST OF FIGURES

FIGURE	PAGE
1. Representation of Continuous Surface by Contours	2
2. Representation of Continuous Surface by Fishnet Plot	3
3. Representation of Continuous Surface by Transect Plot	4
4. Population of Tennessee Represented by Prisms	6
5. Population of Tennessee Represented by an Early Version of SYMVU Using a Matrix Generated by SYMAP	7
6. Rectangular Columns Created by Raising Sets of Points to Various Heights	8
7. Matrix Where Only One Value May Exist for Any X Position	10
8. Matrix Where Two End Points Can Have Same X Position	10
9. Cylinder Represented by ASPEX Using a Matrix Generated by SYMAP	12
10. Cylinder Represented by DISCON	14
11. Line 1 Is Tested against All the Surfaces A, B, C, D, E, and F	17
12. Quantitative Invisibility of Point P = 2	20
13. Computation of the Quantitative Invisibility	20
14. View Plane of an Object Which Has Been Subdivided by the Warnock Algorithm	25
15. Choropleth Map of Tennessee Population	31
16. Flow Chart for DISCON	32
17. Data Structure for Input and Output Files for Case 1	38
18. Polygons with Diagonal Overlays	40
19. Data Storage of the End Points of the Diagonals	42
20. Diagonal Incremental Corrections for Corners	43
21. Data Storage of the Intersections Along Each Diagonal	45

FIGURE	PAGE
22. Special Cases for Line Segment Intersections	47
23. Transformation Matrix	51
24. Rotation of Position Vector about the Point M,N through an Arbitrary Angle	53
25. Transformation Matrix Produced by Carrying Out the Two Interior Matrix Products of Figure 24	53
26. Diagonal Plotting and Change Chart	54
27. Example: Diagonals at 45 Degrees and Rotation of 100 Degrees to Show the Effect of SUBROUTINE CHANGE	55
28. Plotting Sequence	58
29. Population of Tennessee by Counties Represented by ASPEX Using a Matrix Generated by SYMAP	62
30. Population of Tennessee by Counties Represented by DISCON .	63
31. Examples of Rotation about X-Axis with Diagonals of 45 Degrees	64
32. Examples of Rotation about Y-Axis with Diagonals of 45 Degrees	65
33. Examples of Rotation about Y-Axis with Diagonals of 135 Degrees	66
34. Examples of Rotation about Z-Axis or Tilt	67
35. Data Structures for Input and Output Case 1 Example	83

CHAPTER I

INTRODUCTION

The purpose of this thesis is to develop an interactive computer program to represent with parallel lines on a CRT discontinuous surfaces made up of irregular polygons since a survey of the literature reveals that most currently used computer programs do not accurately represent these discontinuous surfaces. This program will produce three-dimensional representations from two-dimensional data and permits an interactive display. This display process can be applied to any two-dimensional data as long as closure exists for each polygon that makes up the coordinate data set. Closure exists when the starting point of the first line segment of a polygon and the ending point of the last line segment have the same coordinates. The third dimension is computed from the height of each polygon which may represent any user defined two-dimensional function.

Surfaces can be classified as either continuous or discontinuous. There are several ways to represent a continuous surface. Some of the conventional ways of expressing continuous surfaces are by contours (Figure 1), fishnet (Figure 2), or by transects (Figure 3). However, not all surfaces are continuous in nature, but rather are naturally discontinuous. One such example is county or state information data. In these cases, the county or state lines are the continuous boundaries but the information is of a discontinuous nature. One of the existing ways to represent this data is by creating prisms by raising the

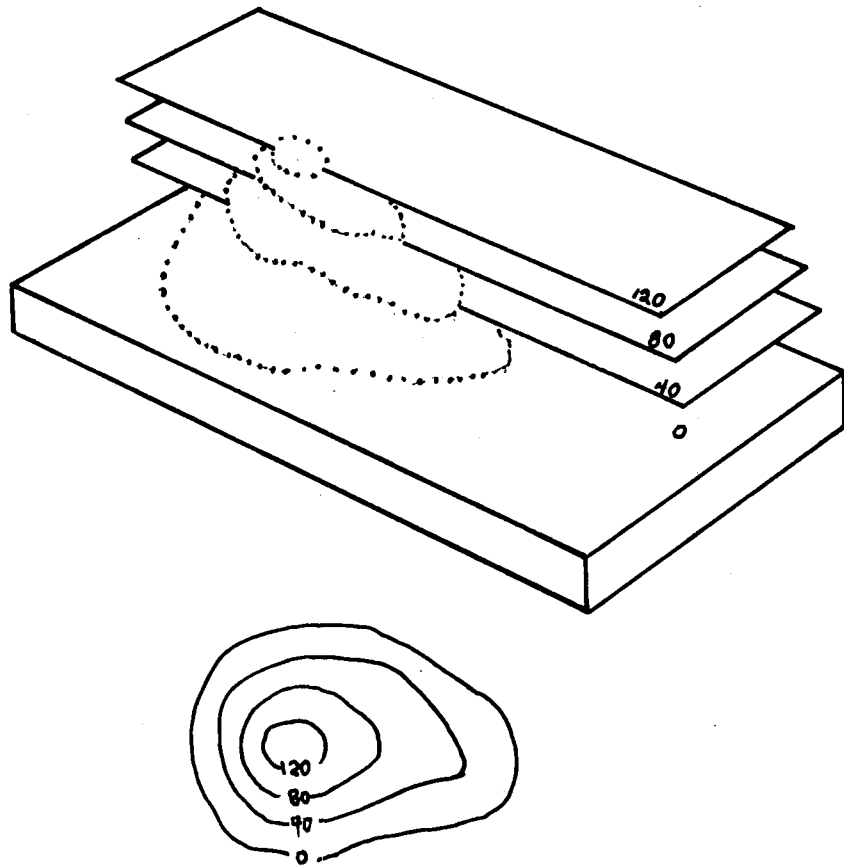


Figure 1. Representation of continuous surface by contours.

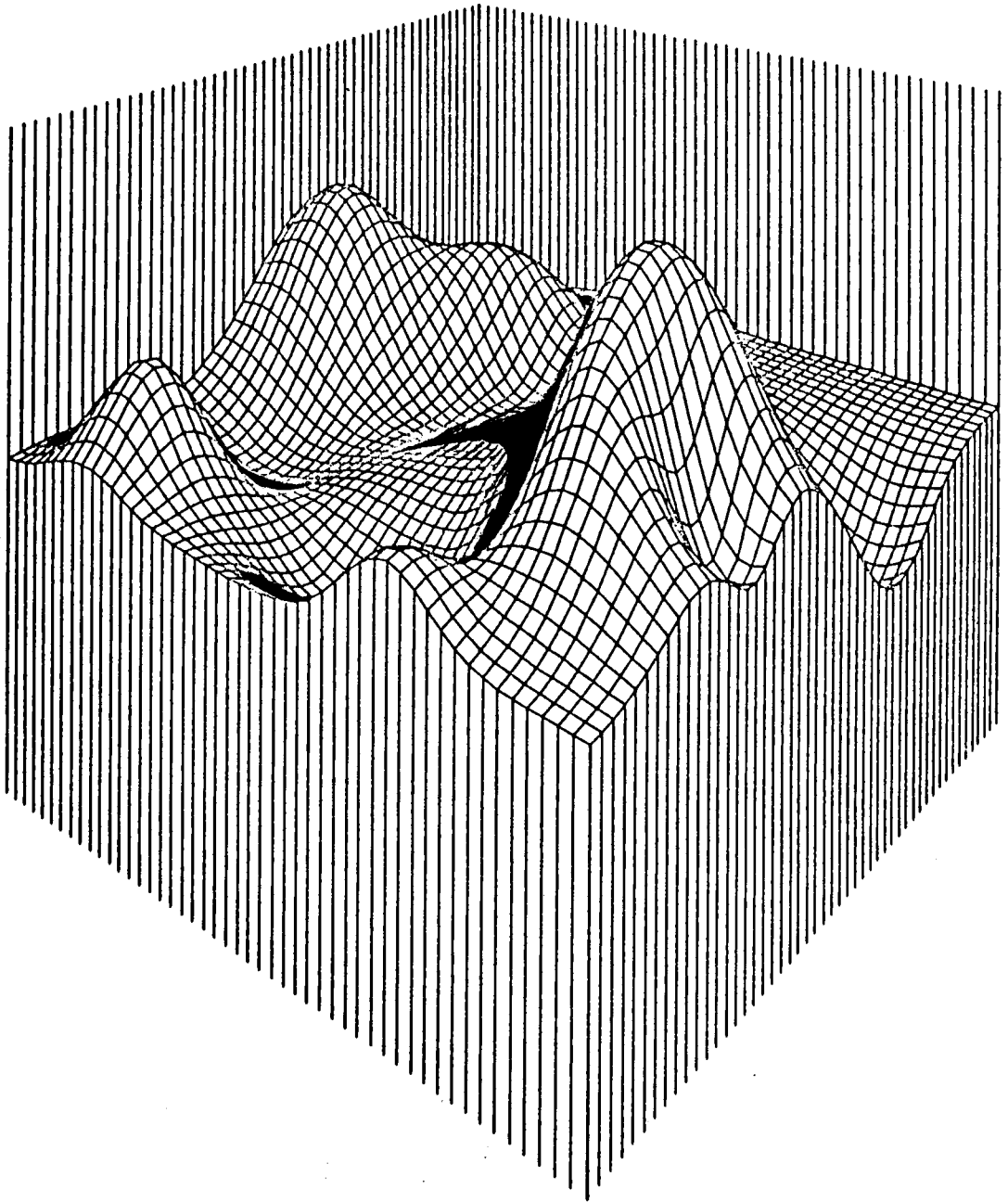


Figure 2. Representation of continuous surface by fishnet plot [11, p. 27].

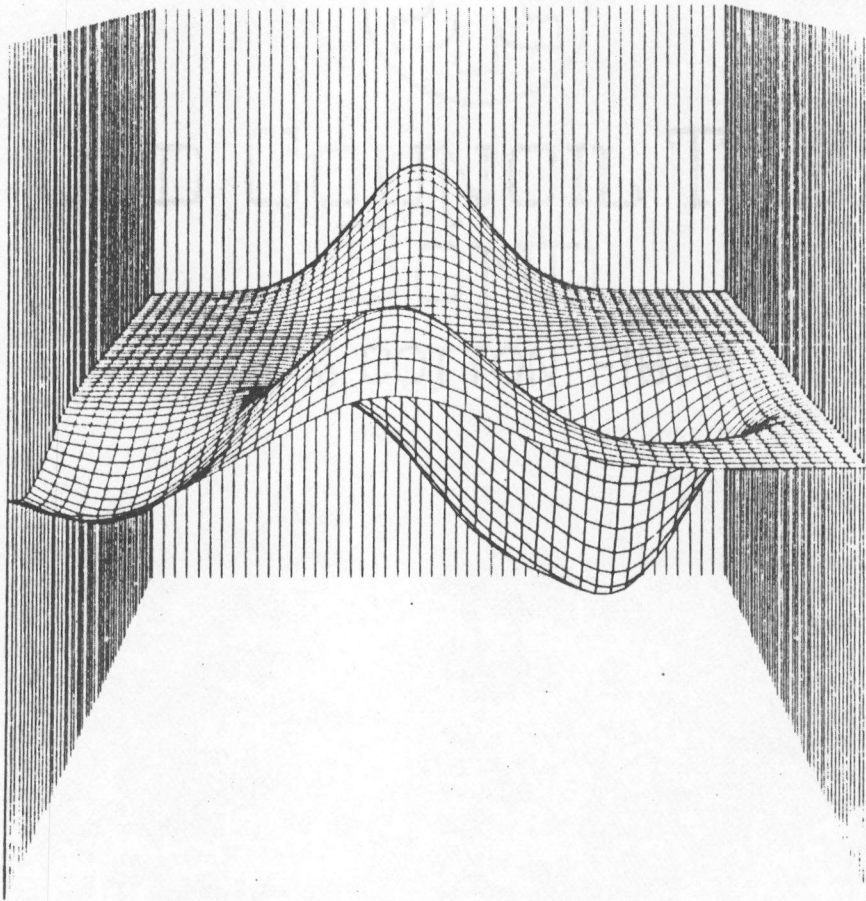


Figure 3. Representation of continuous surface by transect plot [11, p. 24].

irregular polygons to a certain height corresponding to the information content of that polygon (Figure 4). Tobler [15] has developed a program to generate such prisms on a CRT or plotter. Discontinuous data is frequently represented by using programs such as ASPEX or SYMVU (Figure 5) which are really not appropriate for data of this type because the discontinuous boundaries are shown as diagonal lines. The nature of these diagonal lines is well illustrated in Figure 6 where a fishnet plot is used to represent rectangular columns but the columns appear as spikes.

Existing fishnet and transect programs currently used in computer graphics as ASPEX and SYMVU do not accurately represent discontinuous surfaces because plotting is accomplished by connecting points of a three-dimensional matrix. Since two values cannot exist at one point in such a matrix, boundaries between prisms on a discontinuous surface are shown by sloping lines which is inappropriate to these surfaces. The program developed for this study can represent these discontinuous surfaces since it does not incorporate a matrix.

A. OUTLINE OF THESIS

This idea of discontinuous surface plotting will be described in detail in Section B of this chapter. The next step in the program concerned the development of a way to eliminate hidden-lines, a problem common in almost all applications of computer graphics. Chapter II will discuss existing hidden-line algorithms and the characteristic features of each one.

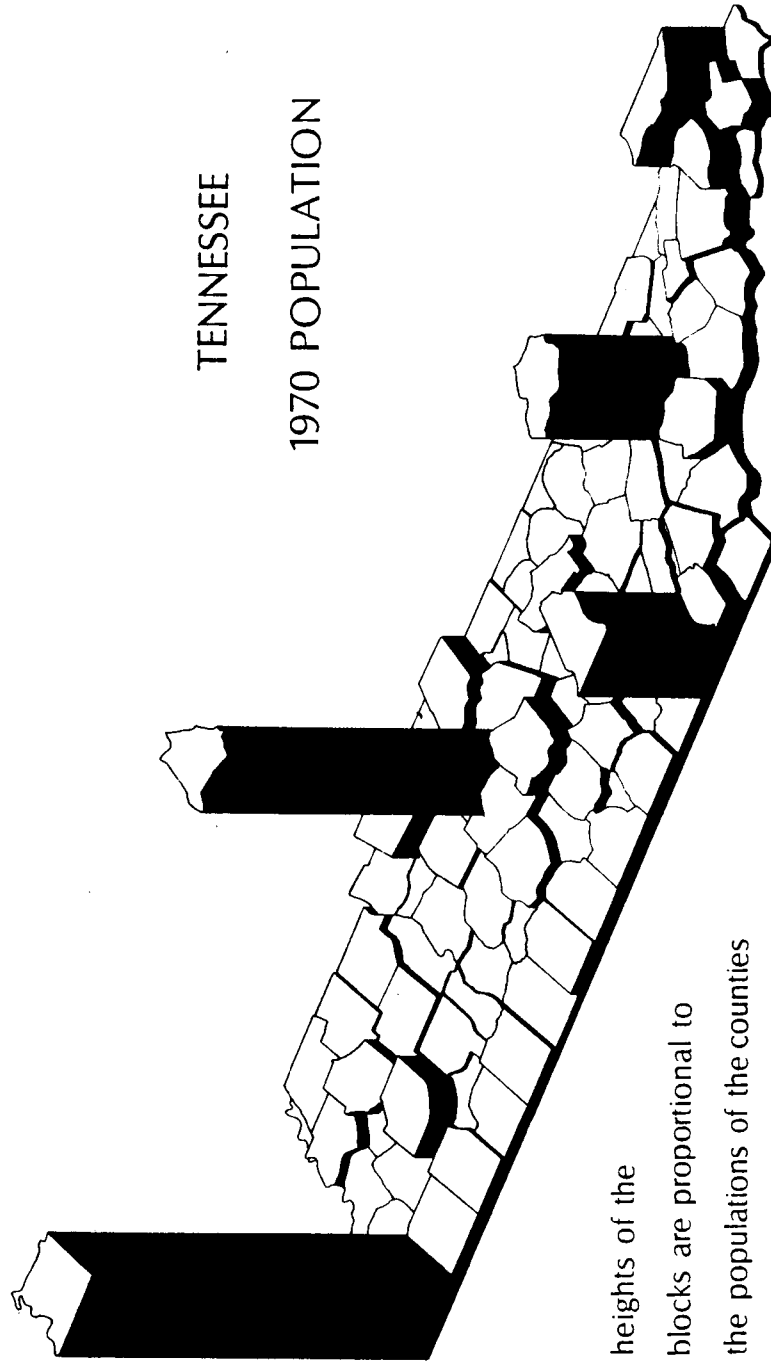


Figure 4. Population of Tennessee represented by prisms [courtesy Dr. James R. Carter, Geography Department, University of Tennessee].

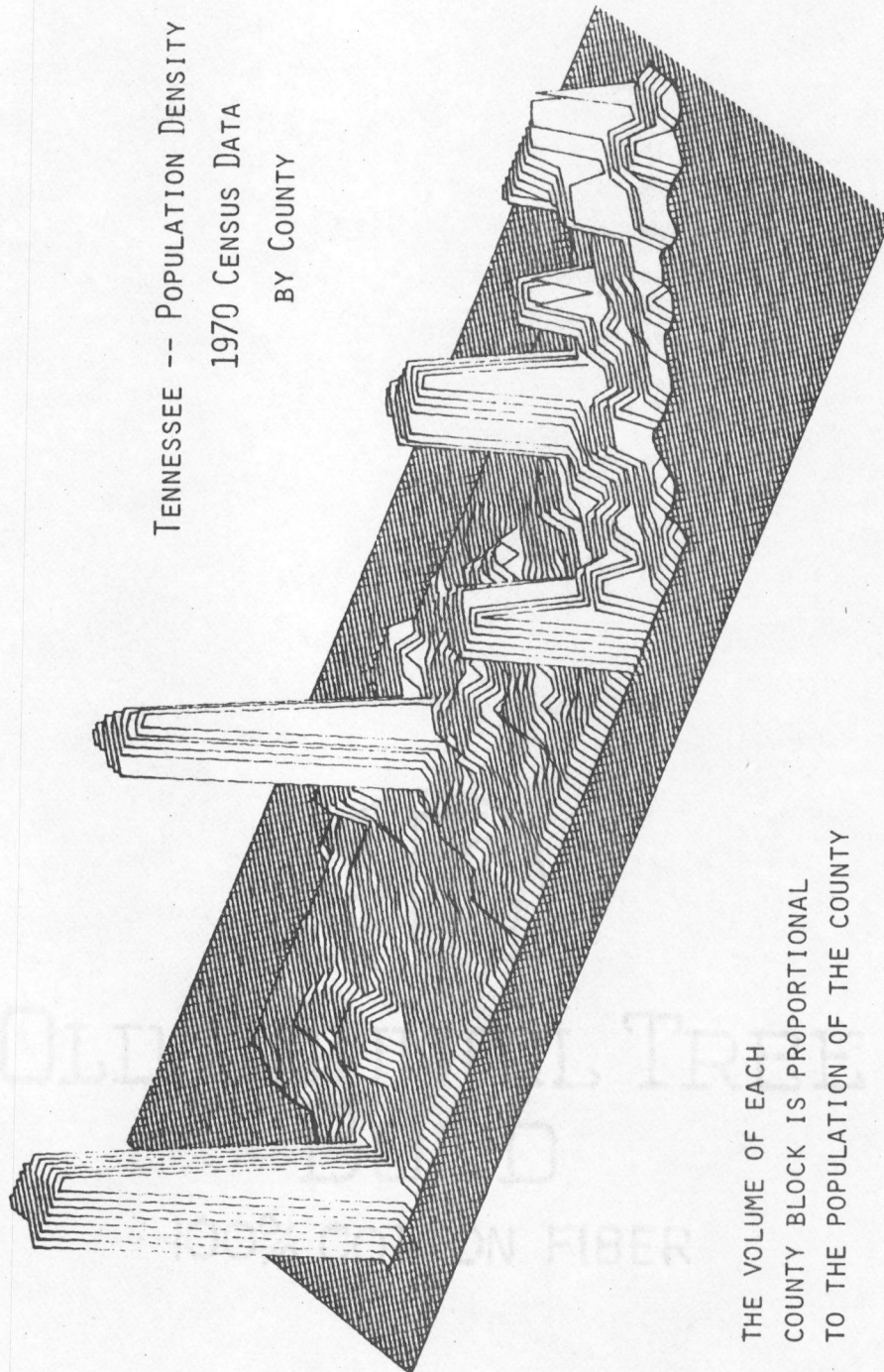


Figure 5. Population of Tennessee represented by an early version of SYMVU using a matrix generated by SYMAP [courtesy Dr. James R. Carter, Geography Department, University of Tennessee].

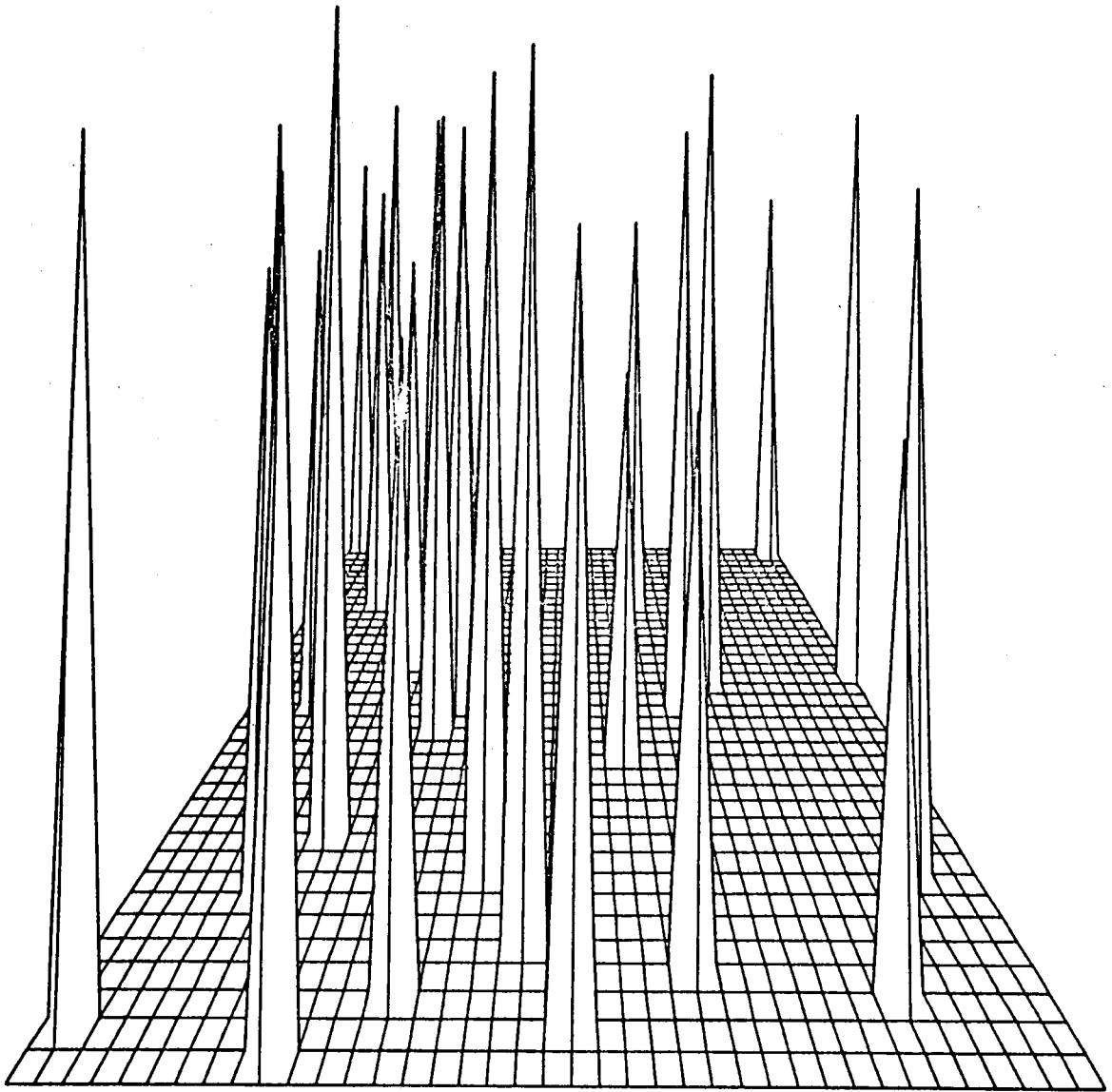


Figure 6. Rectangular columns created by raising sets of points to various heights [11, p. 36].

Chapter III will discuss DISCON—the interactive computer program in FORTRAN IV which has been developed to draw graphically a three-dimensional block diagram from the two-dimensional irregular polygonal data.

B. DISCONTINUOUS THREE-DIMENSIONAL SURFACE PLOTTING

Most programs that represent three-dimensional surfaces (X, Y and Z axes) with either parallel lines or a fishnet employ a matrix of Z values as the input and draw the surface by connecting diagonals between transformed XY points of the matrix. Because the plotting is done utilizing this matrix concept only one value may exist for a given XY position in that particular matrix (see Figure 7 for the two-dimensional case). This is the single value nature of a continuous function where for any X value there is only one Y value in the two-dimensional case. In the three-dimensional case, for any XY value there is only one Z value. Existing matrix-based programs that construct three-dimensional representations of surfaces are useful and applicable in most cases, but not all surfaces can be drawn appropriately from a matrix with nonvertical lines. To correct this misrepresentation of discontinuous surfaces, a vertical line must be perpendicular to the XY plane, and is determined by the two end points. Both of these end points must occur at the same XY position (see Figure 8 for the two-dimensional case). The central theme of this thesis was to develop such a program, hereafter referred to as DISCON, a FORTRAN IV program to be implemented on the DECsystem-10.

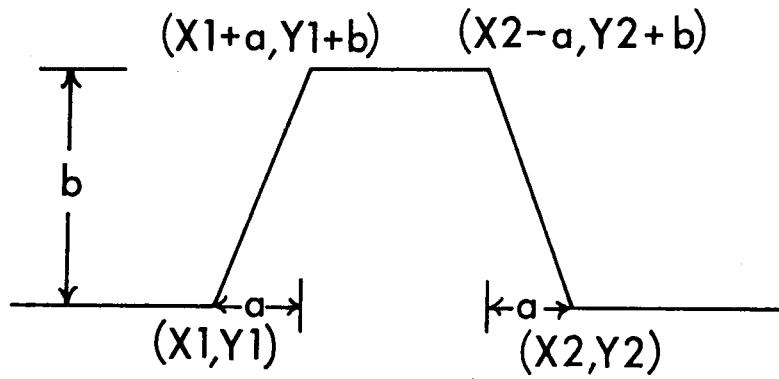


Figure 7. Matrix where only one value may exist for any X position.

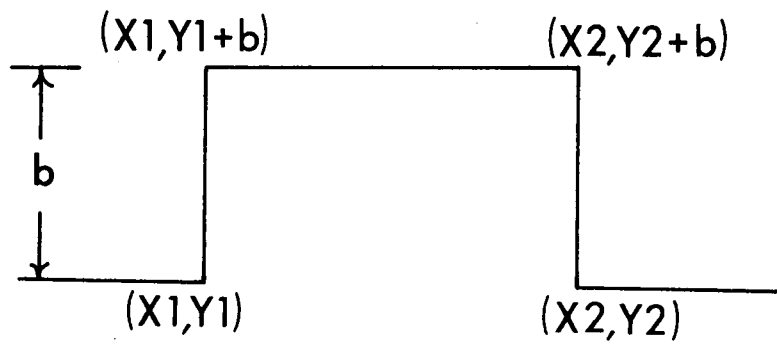


Figure 8. Matrix where two end points can have same X position.

Two examples of discontinuous surfaces are: (1) a cylinder rising out of a plane, or (2) a choropleth map showing the population densities of several counties. A block diagram can be used to illustrate the form of a cylinder rising out of a XY plane which might be applied as a low pass filter in the image processing terminology [7]. A drawing illustrating this same point is shown in Figure 9. This feature is not a true representation of a cylinder since the base of the cylinder has a different surface area than the top of the cylinder, in this case larger. The matrix for this cylinder was generated by SYMAP and drawn as a surface by ASPEX. SYMAP with its line-printer base, can produce a matrix of regularly spaced data values which can be used as input to a three-dimensional mapping program such as SYMVU or ASPEX. Such a matrix can be generated for contour (isopleth), conformant (choropleth), or proximal maps.

SYMVU generates a three-dimensional perspective surface using a pen or CRT plotter. SYMVU detects hidden-lines by keeping track of the highest visible point (greater Y value) in an array called HT(10,000) where each increment is 1/10,000 of the X axis. Incremental plotting is accomplished by comparing the value of the point being plotted at that X value with the value of the point in the HT(10,000) array which represents a Y value. If the point being plotted has a greater Y value, it is a visible point, and SYMVU plots it and updates HT(10,000) with that new Y value for that X point. If the Y value point is less than the HT(10,000) value, it is invisible and therefore not plotted or updated.

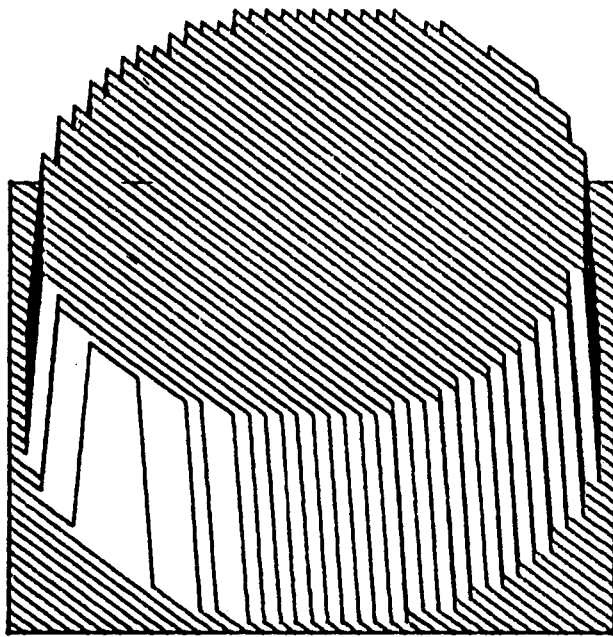


Figure 9. Cylinder represented by ASPEX using a matrix generated by SYMAP. The surface area at the top is less than that at the bottom which is characteristic of block diagrams drawn from matrices.

Figure 10 represents a similar type of situation as Figure 9, except that vertical lines are perpendicular to the XY plane and the base and top of the cylinder represent the same surface area. This surface was drawn by DISCON. The difference in the surface areas at the base and top of the polygons in these plotting routines illustrates the uniqueness of this problem.

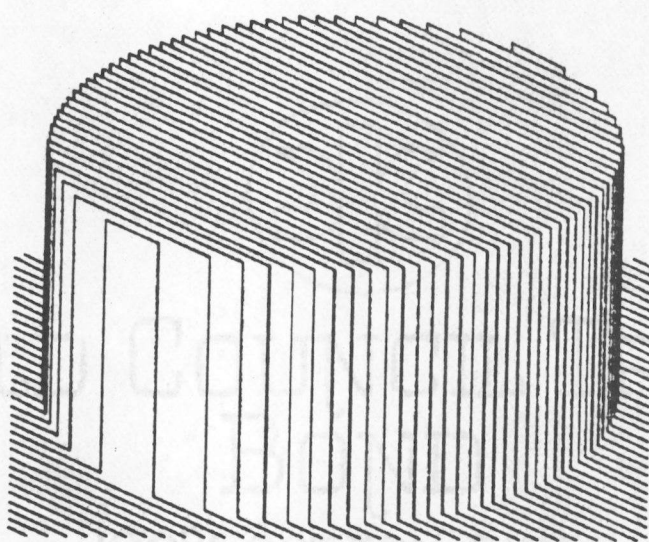


Figure 10. Cylinder represented by DISCON. Surface area at the top is the same as that at the bottom.

CHAPTER II

HIDDEN-LINE ELIMINATION

Hidden-line elimination almost always occurs in any true representation of a two- or three-dimensional rendering of an environment on a two-dimensional surface. There are many different types of algorithms to remove the hidden-lines of a scene. The algorithms can be divided into three main categories through the use of: (1) object space, or (2) image space, or (3) the combination of these two types known as list-priority computations.

A. OBJECT SPACE ALGORITHMS

Object-space algorithms are those algorithms "where computations are performed to an arbitrary precision, usually the precision available in the computer executing the algorithm" [14]. From this, the cost of the algorithm grows as the complexity of the scene. Watkins [17] refers to these types of algorithms as path of edges algorithms. Two types of object space algorithms occur by either types of the following testing: (1) edge against volumes, or (2) edge against edges.

The Roberts algorithm is the only one of the object-space type 1 edge against volumes testing algorithm discussed. This algorithm was implemented in 1963 and was the first solution to hidden-line elimination. Each line is tested against every opaque surface in the scene to determine which portions of the line are visible from the viewpoint, with resulting plots of the visible portion. For example, line 1 is

tested against all the surfaces A, B, C, D, E, and F, with the visible portion being plotted (Figure 11). Roberts developed an algorithm that performed hidden-line eliminations for convex planar polyhedra. Since plane-surfaced solid objects can be defined by a unique number of such polyhedra [9], the algorithm is very general. The test is performed on each plane P_0 and every point S_0 . Plane P_0 has the equation $aX + bY + cZ + d = 0$ where X, Y, and Z are measured in the object coordinate system. P_0 is stored in a normal vector $[a \ b \ c \ d]$. Point S_0 in homogeneous coordinates is represented as $S_0 = [WX \ WY \ WZ \ W]$. From the inner product $P_0 \cdot S_0$, point P_0 can be determined whether it is visible or not. If $S_0 \cdot P_0 = 0$, then S_0 lies on P_0 . If $S_0 \cdot P_0 \neq 0$, the sign of the product determines which side of the plane the point is on. Roberts uses a positive product to represent a point inside a polyhedron [9].

Roberts eliminates lines in three stages which are as follows [9]: (1) clipping against screen boundary, (2) rejection of back lines, and (3) testing the line against the volumes.

Clipping is performed in three-dimensions to eliminate those segments which are invisible because they are either off one of the four screen sides (top, bottom, left or right), or behind the viewpoint. This technique is accomplished by checking points on each line against a special volume matrix which represents the edges of the viewbox in screen coordinates [9].

Rejection of back lines, that is, those edges of a polyhedron which are hidden by that same polyhedron, is accomplished through the inner

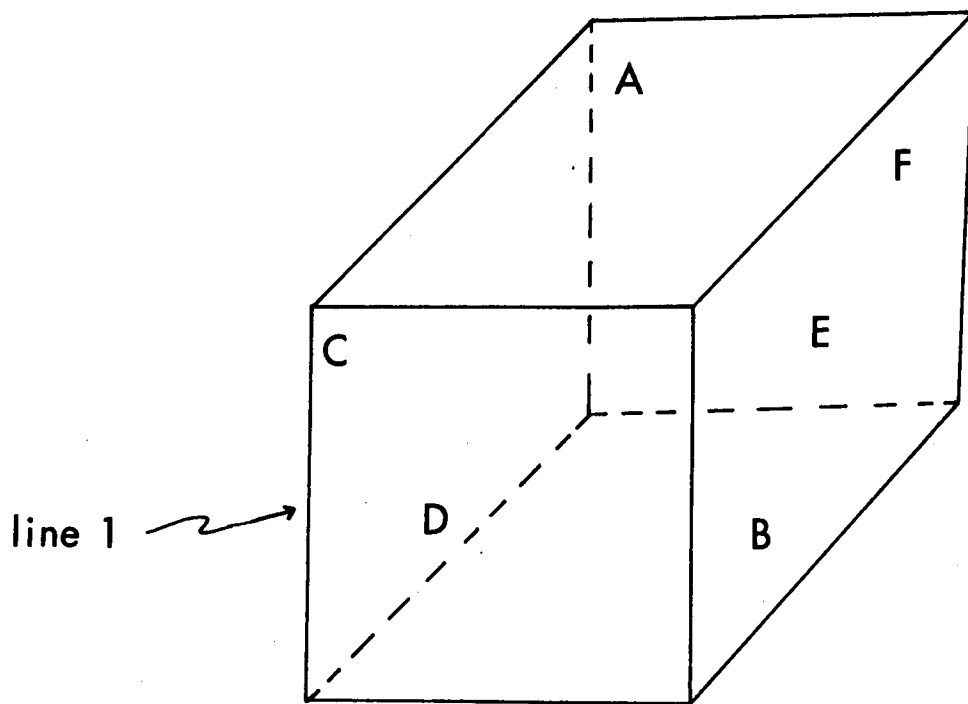


Figure 11. Line 1 is tested against all the surfaces A, B, C, D, E, and F.

produce concept mentioned earlier. Testing each line against all other volumes is the next step. First, a line is drawn from a point on the line being tested to the viewpoint in a negative Z direction. If this line reaches $-\infty$ without passing through any polyhedra, then the point from which it was drawn is visible. If the line toward $-\infty$ passes through a polyhedra, it is invisible [9]. These three stages of elimination indicate four possible outcomes: (1) the line is entirely hidden or invisible, (2) the line is entirely visible, (3) one segment of the line is visible and one invisible, or (4) two portions of the line are visible. Any visible portion must be retested [17].

Roberts uses the minimax test very effectively. This test is a crude comparison of one object to another and eliminates the second object if it does not overlap. This is accomplished by the following: if maximum X(Y) coordinate for the first surface is less than the minimum X(Y) for another surface, the surfaces cannot overlap. By eliminating the object before the matrix multiplication step, a great deal of computation time is saved by utilizing the concept of object coherence or that the objects cluster together.

The Roberts algorithm has two major drawbacks. The first drawback is that the objects must be convex. The second being that the computation time for this algorithm of edge-volume comparison grows as the square of the number of objects, because every edge must be compared and tested to every object in the scene. For example, computation times of one second for objects up to 30, 90 seconds for 40 objects, and 22 minutes for 200 objects were reported [14].

The second type of object-space algorithms is tested by edge against edges. There are three algorithms of this kind which will be discussed. These are: (1) A. Appel, (2) P. P. Loutrel, and (3) Galimberti and Montanari.

The first of these, Appel's algorithm developed in 1967, produced a display by means of scan lines. The question of which parts of which surfaces are visible is solved by a method called "quantitative invisibility." The structures are composed of planar polygonal surfaces [1, 6]. "Quantitative invisibility" of a point is the number of relevant faces that lie between the point and the viewpoint. Figure 12 illustrates the two-dimensional case. Appel computes the quantitative invisibility of every point on each relevant edge but uses edge coherence to limit the computing requirements [14]. Edge coherence are lines that are close to each other and thereby have similar intersections. "If the quantitative invisibility of the initial vertex of the edge is known [Figure 13, point IV], the visibility of each edge segment can be calculated by summing invisibility changes" [14]. These invisibilities change (Figure 13, points IC1 and IC2) only where images of contour edges intersect the image of the relevant edge and the contour edges are closer to the viewpoint than the relevant edge (Figure 13, lines RE and CE) [14]. From these summings of initial and incremental invisibilities, the final vertex (Figure 13, point FV) can be computed which is the initial vertex of other edges leaving that point. "Invisibility corrections" must be accounted for when faces that interact a certain vertex hide a relevant edge starting from that vertex [14].

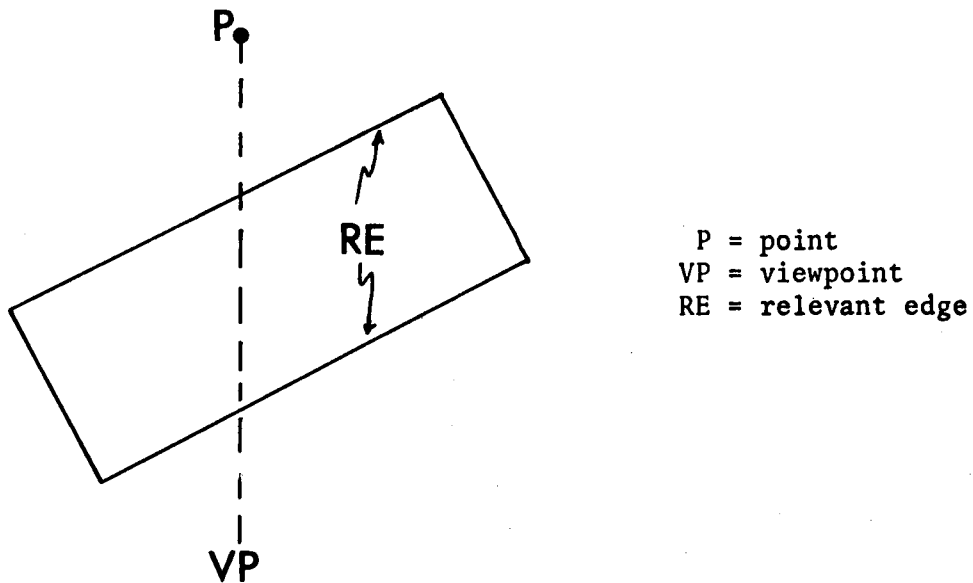


Figure 12. Quantitative invisibility of point $P = 2$.

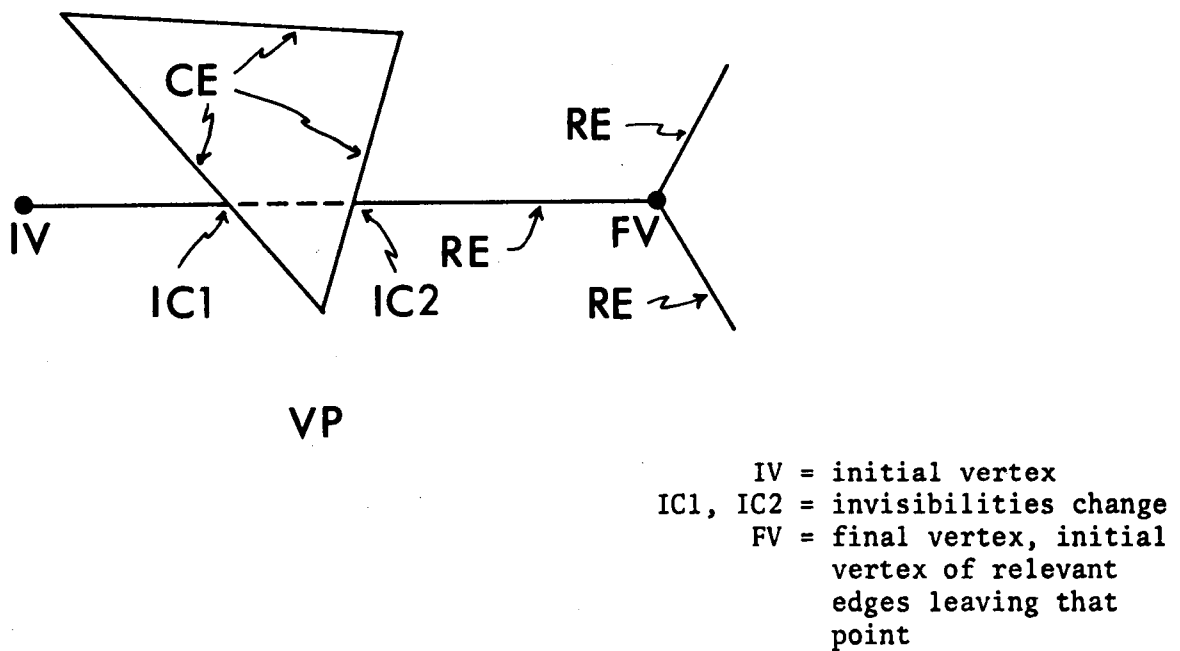


Figure 13. Computation of the quantitative invisibility [14, p. 21].

The Loutrel algorithm, also developed in 1967, removes hidden-lines in the same manner as the Appel algorithm but Loutrel uses his own terms. "Order of invisibility" has the same meaning as Appel's "quantitative invisibility." "Boundary edge" has the same meaning as contour edge. Loutrel intersectional computations are done in a two-dimensional plane after his algorithm has transformed the edge onto the picture plane. When he finds an intersection, he decides which edge hides the other by depth computations.

The Galimberti and Montanari algorithm, developed in 1969, is similar to the previous edge against edges algorithm of Appel and Loutrel. There are three basic steps which are involved in the Galimberti and Montanari algorithm [6].

Step 1. Every edge is recognized as corresponding to a convex or concave dihedral.

Step 2. All lines hidden by their own volumes are eliminated; here it is necessary to know whether an edge is concave or not. At this point, if it is found that a convex object has to be represented, the program stops automatically.

Step 3. All the edges not hidden by their own volumes are sequentially examined. For each edge, one of two extreme points is considered. The task is to determine the set of faces that hide this point, that is, the "nature" of the point. Then, starting from the chosen point, the segment is examined, and the set is modified by making use of the intersections of the projected segments on the picture plane.

This concept of the "nature" of the point is different from Appel and Loutrel algorithm. Where Appel and Loutrel compute its number of faces hiding a point, Galimberti and Montanari compute a set of faces. When dealing with initial vertexes, the invisibility correction, and the incremental invisibility, all three algorithms are basically the same. All three algorithms do some kind of sorting on the relevant edges which are fairly simple, and take into account edge coherence. Ad hoc methods for solving particular cases due to the incremental nature of the calculation are mentioned in connection with Galimberti and Montanari and Loutrel algorithm. All three algorithms of edge against edges comparison require a great deal of computation in three areas [13]. The first area is of initial visibility. Many computations are due to the required comparison of all faces to each object. The next area of many computations is in connection of invisibility correction which must be performed twice for every relevant edge and for about three faces each time. The third area is the edge interaction test where each relevant edge against each contour edge must be compared [14].

B. IMAGE SPACE ALGORITHMS

The resolution of the screen is the limiting factor of image-space algorithms. Each dot on the screen is assigned a desired intensity depending on what is visible for that dot. From this, the cost of image-space algorithms will reach an upper limit because there is a unique number of dots on the display screen. Watkins [17] refers to these types of algorithms as sample space algorithms. Two types of image-space

algorithms exist. The first kind deals with scan lines and the others deal with area subdivision.

Several scan line algorithms will be discussed including:

(1) C. Wylie, G. W. Romney, D. C. Evans, A. C. Erdahl, (2) W. J. Bouknight, and (3) G. S. Watkins. All three of these scan line algorithms have the same four basic steps which are [14]: (1) Y sort—edges are sorted by Y, from top to bottom on the screen, so that only those edges intersecting the current scan line need be examined; (2) X sort on each scan line, from left to right on the screen, appropriate sample spans are determined (this usually involves sorting edges on the scan line by X coordinates); (3) span cull—within each sample span, the algorithm must cull out the segments which fall in the span and therefore must be examined; and (4) Z depth search—the segments that fall within a span are searched to find which one is visible.

The Wylie, Romney, Evans, and Erdahl algorithm was the first to use these four basic steps in 1967. The Y-sort was a bucket sort or bubble sort on the upper most vertex of the nonintersecting triangles which made up the objects. The X bucket sort was used on faces intersecting the scan line, depth comparisons were made only on triangles crossing the scan line thereby not having depth computations on each raster point [17]. In this algorithm the X sort and Y sort are very efficient. Even though the Z depth search need not be computed for every raster point, the search takes up a considerable amount of computation time because of the planar equation involved. One drawback of this algorithm is that it only handles nonpenetrating triangular faces [14].

The second algorithm of this scan line type was developed by W. J. Bouknight in 1969. Both Bouknight and Watkins use the idea of keeping track of the X intercepts of edge crossings from one scan line to another through use of updating an X sort line, which Wylie, Romney, Evans and Erdahl did not [14]. This idea is another example of edge coherence.

The third scan line algorithm was developed by Watkins in 1970. From the points already mentioned, the Watkins algorithm differs in two ways. First the left end of the sample scan is fixed and the right end moves toward the left to find the visible portion or segment. The other difference is that the Z depth search is a logarithmic search which is very economical. In the Watkins algorithm the Y bucket sort, X merge, Y bubble sort on each scan line are all very efficient. The Z depth search does however take more computation time.

The next type of object space algorithm deals with the concept of area divisions. The Warnock algorithm is of this type and probably the one most single referenced hidden line algorithm. Figure 14 is adopted from Warnock's work to illustrate the concept of area division.

Either its contents are simple enough to process with some decision procedure or they are not sufficiently simple. If they are not, then we subdivide the square into four subsquare as was asked about the parent square. This process is repeated recursively until one of two conditions holds with respect to a subsquare. If the contents of the square are simple enough to process, then data is put into a display file. If the square is as small or smaller than the desired resolution of the picture, then put a reference to the surface nearest to the observer into the display file [16].

This algorithm has the characteristics of being "nondeterministic" [5] since it can announce failure if it cannot analyze the square or window

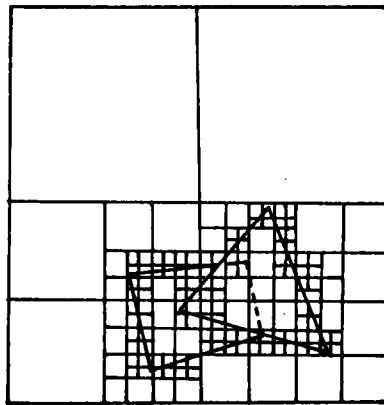


Figure 14. View plane of an object which has been subdivided by the Warnock algorithm [9, p. 300].

and therefore subdivide it. These subdivisions need not be square. The Warnock algorithm has two basic logic units, a "control unit" and a "looking unit" [4]. The control unit chooses a portion of the picture for the looking unit to do its work on. The "control unit" has two main functions: (1) make sure all windows have been examined and (2) subdivides windows where failure has occurred which the looking unit tells it. The "looking unit" work consists of three answers: (1) no features are visible, (2) a display is generated because the window is classified as simple, and (3) failure because the window is too complex. When the looking unit is of a very sophisticated model, not as many subdivisions need be done because it can handle unique cases.

The basic operation of the Warnock algorithm is to detect whether any edge of the polygon passes through a window. A special mode of the polygon passes through a window. A special mode of the clipping divider [13] was developed for this purpose. It merely tells whether or not a particular certain line passes through a window and does not compute the intersections.

When the "control unit" subdivides windows, that is, a window which is too complicated, sons are pushed into a stack. One way to scan the stack is by the "constant-depth walk" which uses a FIFO (first in, first out) stack of involved windows. The other way, which is preferred, is by prefix-walk which uses LIFO (last in, first out) stack which cannot exceed N , the maximum number of subdivisions. This preferred way saves storage.

Warnock's algorithm has two main disadvantages. The first being the computation of the relation between a face and an area is difficult. The second, is that no suitable output device is available to absorb the output information of an area at a time [14] and output is not suitable for raster-scan displays. Computation time is roughly proportional to the complexity of the final display and not proportional to the scene.

C. LIST-PRIORITY ALGORITHMS

There is another category which is part object space and part image space, that is the list-priority category. In this category, the algorithms are displayed on a fixed resolution display as in the image space algorithms. The list-priority algorithms are like object space algorithms in that the calculations of depth overlaps are performed in the same way. The list-priority algorithms, differ from image space in the way the final visible surfaces are computed. List-priority algorithms precompute in object space a visibility ordering of the surfaces. In image space the visibility test is last as in depth computation. The two list-priority algorithms that will be briefly discussed are by (1) R. A. Schumaker, B. Brand, M. Gilliland, and W. S. Sharp developed in 1969; and (2) M. E. Newell, R. G. Newell, T. L. Sancha developed in 1972.

The two algorithms differ in two ways, first their list-priority computation and second, clustering techniques. In the Schumaker algorithm, priority calculations are performed off-line by humans before the program ever starts, which does not depend on the viewpoint. The Newell algorithm computes a priority list from the environment before

processing each frame [14]. The second difference is clustering techniques. The Schumaker algorithm divides the scene into cluster of faces and only compares each face with the other faces in its own cluster group to produce bare priority. The Newell algorithm computes each face against all other faces by use of his priority computer to find the priority-ordering of faces of the scene [14].

The Schumaker algorithm is the only algorithm which makes use of frame-to-frame coherence where successive frames are likely to be closely related. This algorithm also makes use of cluster coherence which is similarity between clusters, implied-edge coherence, and does not consider viewpoint. Even though the sorts in the Schumaker algorithm are very efficient, the comparison of each face to all the other faces in that cluster require a great deal of computation time. Especially when there are a great number of clusters.

The Newell algorithm makes use of implied-edge coherence and the main contributions of the Newell algorithm are the priority computer and overwriting concept and depth coherence [14]. The drawback of the Newell algorithm is the great amount of computation time required to examine faces with all other faces in the Newell special sort algorithm.

In conjunction with other hidden-line algorithms DISCON is classified as a list-priority algorithm because it operates in both object space and image space. DISCON has the properties of the edge against edges type of the object space algorithm because of its test in two areas. The testing occurs first between diagonals and polygon segments to find the intersections. Next, in the hidden-line elimination section, the testing

of the segment being plotted against the visible top line is implemented. In both of these two areas, the testing is performed to an arbitrary precision which is characteristic of the object space algorithm. DISCON is similar to the image space algorithms because it takes into account that the algorithms are performed with only the resolution of the screen concerned. During the hidden-line elimination section of DISCON, an intersection is formed between the segment being plotted and the top line. An incremental amount is added to this intersection before continuing along the segment. The rationale for this incremental addition is avoid encountering the same particular intersection again. DISCON has the same concepts of priority calculations as to the list-priority algorithms. These concepts are displayed in the sorting of the intersection along the diagonal in a certain fashion so the calculations can be handled correctly by the plotting and hidden-line elimination sections of DISCON.

With this background information concerning hidden-line elimination. An algorithm was developed to draw a three-dimensional block diagram from two-dimensional data. This topic will be considered in detail in the next chapter.

CHAPTER III

DISCON: A SYSTEM FOR CONSTRUCTION OF A THREE-DIMENSIONAL
BLOCK DIAGRAM REPRESENTATION FROM
TWO-DIMENSIONAL DATA

A system called DISCON has been developed that permits a three-dimensional representation from two-dimensional data. DISCON has the capability to represent positive surfaces as long as the two-dimensional base file has been inputted so the sorting algorithm can accept it. This two-dimensional base file may consist of polygons or circles represented by unique values of discontinuous surfaces. The circular data computations can be rewritten by the user to meet his own specific needs as long as RPOLY(50,4,50) is filled so the sorting algorithm can accept it. One application of DISCON is its use in representing choropleth map surfaces. The choropleth, sometimes called conformat, map is based on a given set of contiguous areal units, as the counties in a state (Figure 15). The values that are assigned to these areal units are assumed to be applied uniformly throughout the unit and discrete breaks delineate the boundaries.

The flow chart illustrated in Figure 16 represents each major subdivision of the program (A through G) and corresponds to the sections of this chapter. Section F, the last section, describes the hardware and software used for this system.

POPULATION DISTRIBUTION IN TENNESSEE IN 1970

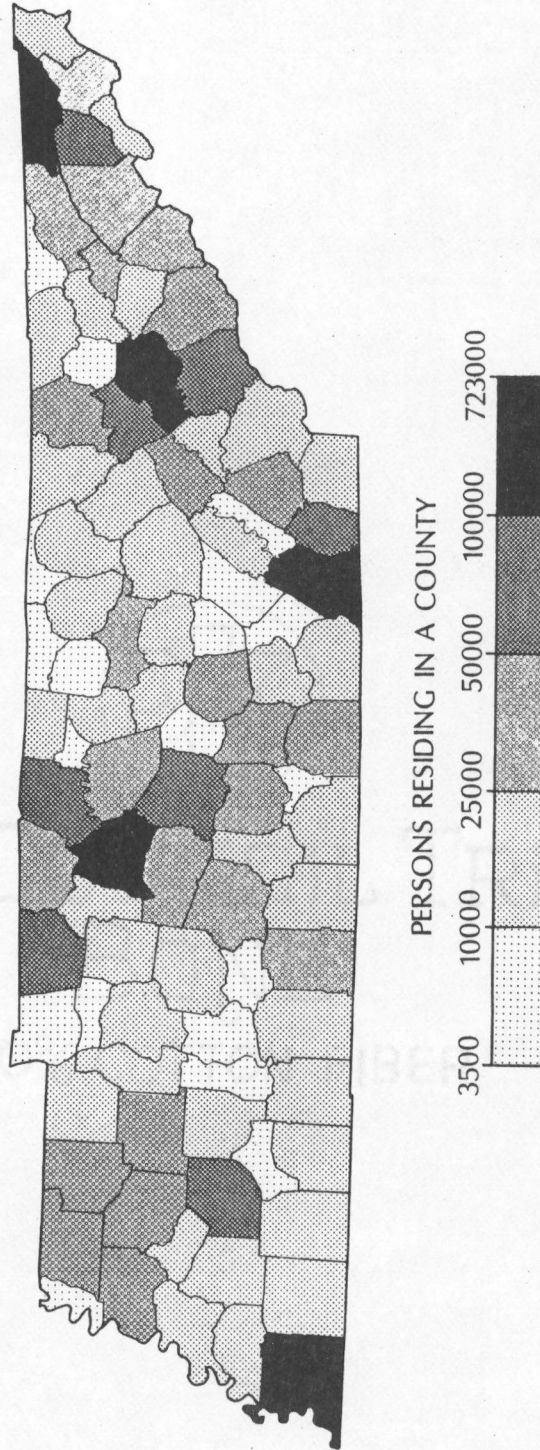


Figure 15. Choropleth map of Tennessee population [courtesy Dr. James R. Carter, Geography Department, University of Tennessee].

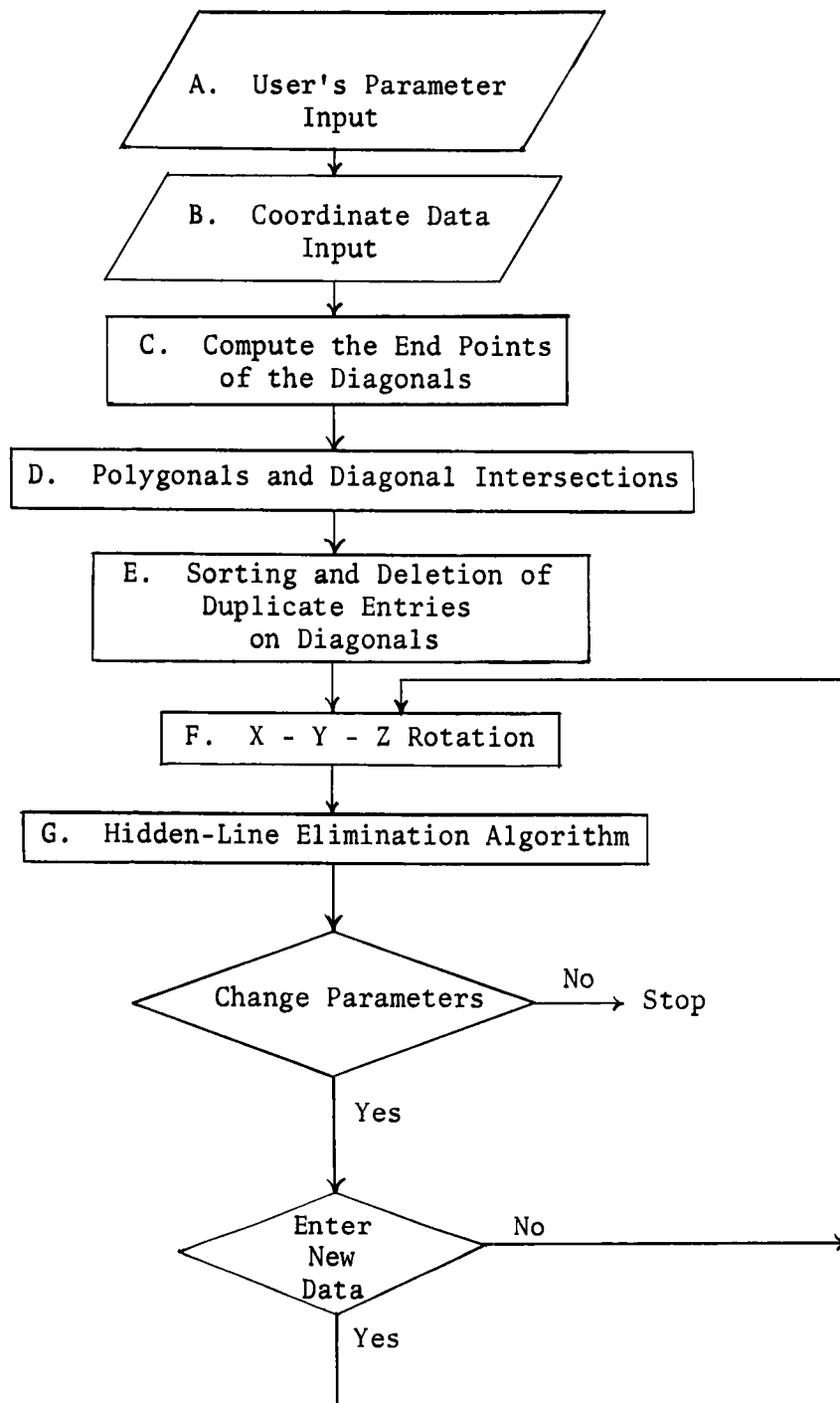


Figure 16. Flow chart for DISCON.

A. USER'S PARAMETER INPUTS

The parameter inputs are the controlling factors that cause this system to be an interactive computer graphic program. Their inputs are entered in response to certain questions at the terminal during execution. Branching and default options are directly related to these inputs. These inputs are straight forward answers to the following questions and must all be answered (braces, {}, are around system output to the terminal).

1. {WHAT BAUD RATE? 30 or 120}

User enters 30 or 120 depending on the baud rate of the terminal being used. {ENTER DATA FOLLOWED BY CARRIAGE RETURN}

2. {WHAT TYPE OF DATA? 1 IF POLYGONAL, 0 IF CIRCULAR}

User enters a 1, if using polygonal data such as the 16 counties for East Tennessee Development Districts or 95 counties for entire state. User enters a 0, if a circular data such as for the filter plot (Figure 5, page 7).

3. {HOW MANY DIAGONALS? 1 to 50}

User enters a number between 1 and 50. This is the number of diagonals in the plot.

If polygonal data are used branch to 6.

4. {ENTER DIAMETER OF CIRCLE < 140.0 UNITS}

User enters the diameter of the circle which must be less than 140.0. The rectangle to be fitted around this circle

has a length of 100.0 units (see Chapter III, Section C for description of rectangle).

5. {ENTER VALUE OUTSIDE CIRCLE, VALUE INSIDE}

User enters the desired heights for outside the circle and inside the circle (see Figure 10, page 14, for display).

GO TO 9 Flow of control branches around polygonal data.

6. {WHAT TYPE OF DATA INPUT? 1 for X, Y, DATA(I), ICOOR(I), 0 FOR X, ICOOR(I)}

User enters 1 when data inputs of X coordinates and Y coordinates of polygons, DATA(data value for polygons), and ICOOR(number of X, Y enters for each polygon) are in four separate files (See Chapter III, Section B).

User enters 0 when data X, Y coordinate inputs of the polygons and polygon data (see Chapter III, Section B).

At this point the correct files are opened.

7. {HOW MANY POLYGONS?}

User enters number of polygons in the data set.

8. {WHAT TYPE OF DIAGONALS? ENTER 1-LINE ALONG COLUMNS OF 90 DEGREES, 2-LINES ALONG ROWS OF 0 DEGREES, 3-LINES AT 45 DEGREES, 4-LINES AT 135 DEGREES}

User enters desired number to produce diagonals (see Chapter III, Section C).

9. {ENTER 1 TO SET LOCAL SCALING PARAMETER, 2 TO SET SHEARING, 3 TO SET VANISHING POINTS, 4 TO CONTINUE}

User enters 1, 2, 3, or 4 depending on desired branch. If 4 is used default options are set, 1. for scaling and shearing, 0. for vanishing points and branch to 13.

10. {ENTER LOCAL SCALING PARAMETERS OF INITIAL OBJECT IN AN X- AND Y-DIRECTION RESPECTIVELY}

User enters local scaling parameters greater than 0. If a 1. is entered, then scaling is unchanged. If the magnitudes are unequal, a distortion occurs. Numbers greater than 1. cause enlargement while numbers less than 1. cause reduction.

GO TO 9

11. {ENTER SHEARING PARAMETERS OF THE INITIAL OBJECT IN A Y- AND X-DIRECTION, RESPECTIVELY}

User enters shearing parameter greater than 0.. These parameters will pull the initial object in the given unit along the respective axis.

GO TO 9

12. {ENTER VALUE OF VANISHING POINT FOR X- AND Y-AXIS RESPECTIVELY, FOR TWO POINT PERSPECTIVE}

User enters the desired value of the vanishing point for the two-point perspective. If either one is 0., then a single-point perspective is created. If both are 0., then no perspective is created and the presentation is orthographic.

GO TO 9

13. If data is circular or input is of case 1 (see Chapter III, Section B), GO TO 15.

14. {ENTER DATA FORMAT}

User enters the format which he wishes the READ statements to use. With this option new data can be read in for each plot interactively.

15. {ENTER OVERALL SCALING OF FINAL BLOCK DIAGRAM $\leq 1.$ and $> 0.$ }

User enters scaling factor which reduces final output by that desired amount.

16. {ENTER VIEWING ANGLE IN DEGREES}

User enters viewing angle from 0.0 to 90.0. The degree angle of 0.0 is defined as looking straight at the object from ground level. The degree angle 90.0 is thus looking straight down on the object where 90.0 degrees would result in seeing just diagonals.

17. {DO YOU WANT ROTATION ABOUT THE Y-AXIS? TYPE 1 IF YES, 0 IF NO}

User enters 1 or 0 depending if rotation of the polygon and diagonal's intersection is desired (see Chapter III, Section F).

If rotation is not desired branch to 19.

18. {THE LIMIT OF THE X-AXIS ARE:} 2F7.2 {LIMITS OF THE Y-AXIS ARE}

2F7.2 {WHAT ROTATION ABOUT AN X - Y POINT (F7.2) WHAT X - Y POINT?}

User enters in desired rotation, -360.0 to +360.0, and what X - Y point rotation is to be performed on.

19. {DO YOU WANT ROTATION ABOUT THE Z-AXIS? TYPE 1 IF YES, 0 IF NO}

User enters 1 or 0 depending if rotation about the Z-axis is required (see Chapter III, Section F).

If rotation is not desired branch to 21.

20. {ENTER ANGLE OF ROTATION ABOUT Z-AXIS}

User enters degree of rotation, -90.0 to +90.0.

CONTINUE with computation of program. After these initial parameters are set up, execution of the program will continue until the final plot is finished. The following control statement then appears on the screen.

21. Enter a 1 to continue. {TYPE 1 TO EXIT, 0 TO CHANGE PARAMETERS AND REDRAW}

User enters desired course of action. If 1 is entered, the program terminates. If 0 is entered, control of program goes to number 13 and normal execution follows.

B. COORDINATE DATA INPUT

There are two cases for entering polygonal data. Each case deals with two-dimensional data in an XY plane with closure on the data sets.

Case 1 has four files in which input is loaded from the user's own area. Figure 17 is a simple example of these four input files which are:

(1) ICOOR.DAT—number of data points in each polygon, (2) XCOOR.DAT—the X coordinates of the polygons, (3) YCOOR.DAT—the Y coordinates of the polygons, (4) ZVAL1.DAT—the data values (third dimension) of the polygons. The various DO loops for reading in the data are set up by the number of polygons for the outside loop and the number of data points from ICOOR.DAT for the implied loop on X and Y coordinate data.

Case 2 reads the XY data from a file named XY.DAT. User must enter the format of the data for the READ statements. As in Case 1, the

INPUT FILES

ICOOR.DAT	Number of vertex points in each polygon
-----------	---

Polygon #	
1	5
2	9
3	5
4	8

XCOOR.DAT	X-coordinates of polygons
1	10. 8.5 7.5 9.0 10.
2	9.0 7.5 8.5 7.5 6.5 5.5 5.0 7.0 9.0
3	7.0 5.0 3.0 3.5 7.0
4	7.5 5.5 3.5 3.0 5.0 5.5 6.5 7.5

YCOOR.DAT	Y-coordinates of polygons
1	5.0 3.0 5.0 7.0 5.0
2	7.0 5.0 3.0 1.0 3.0 4.0 6.0 8.0 7.0
3	8.0 6.0 4.0 8.0 8.0
4	1.0 1.0 2.0 4.0 6.0 4.0 3.0 1.0

ZVAL1.DAT	Z-data values of polygons
1	25.0
2	27.0
3	31.0
4	15.0

OUTPUT FILE

FOR20.DAT	User's Output
	Parameters which were specified
	Any errors, copy of screen output

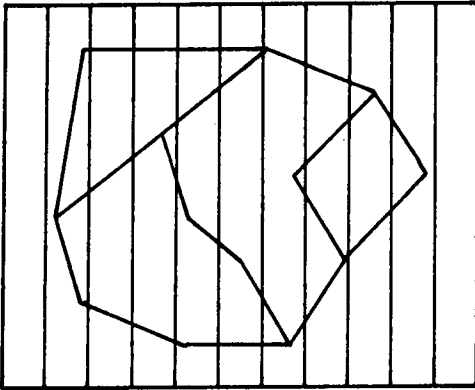
Figure 17. Data structure for input and output files for Case 1.

outside loop is determined by the input parameter of the number of polygons. In this file, the number of points per polygon is determined by the program counting each XY entry, and looking for a flag to determine the start of the next polygon. This number is then entered into ICOOR(I) where I is the polygon number. For this reason there must be a dummy flag for the last polygon. The data value are entered from a file named ZVAL2.DAT. Any such data file stored on the disk could be used. Again, the control of this loop is from the input parameter, number of polygons.

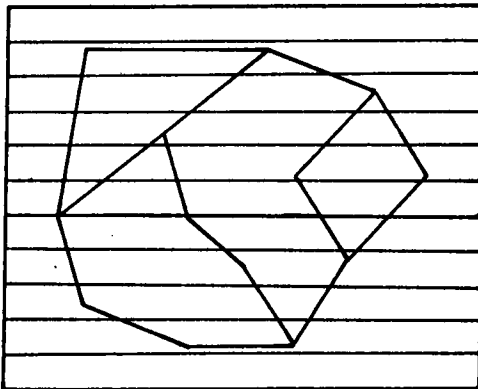
Circular data input is entered interactively from the terminal as was discussed in Section A of this chapter. For both polygonal and circular data, parameters that were specified and error listing will be entered into a file in the user's area named FOR20.DAT.

C. COMPUTE THE END POINTS OF THE DIAGONALS

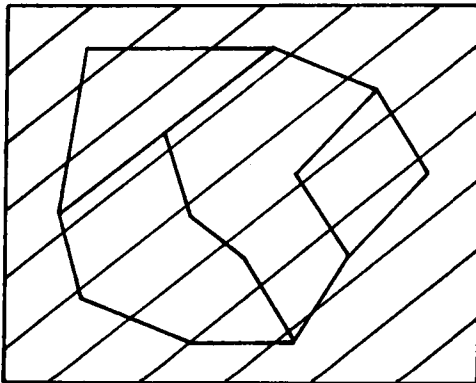
Once the polygonal data have been entered correctly, the next step is to overlay these polygonal with a rectangle, larger than the figure, with even spaced diagonals at 0° , 45° , 90° , or 135° (Figure 18). These lines are called diagonals, although it is acknowledged that in some cases they may be parallel to one side of the rectangle and are not really diagonals. Fitting a rectangle around the polygons is accomplished by finding the maximum and minimum values of the XY values of all the polygons. From this information the four corners of the rectangle can be determined.



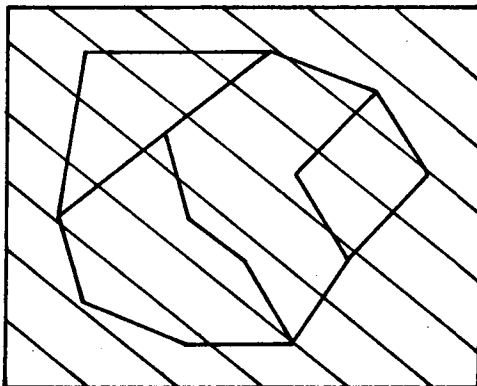
Case 1—diagonals at 90 degrees



Case 2—diagonals at 0 degrees



Case 3—diagonals at 45 degrees



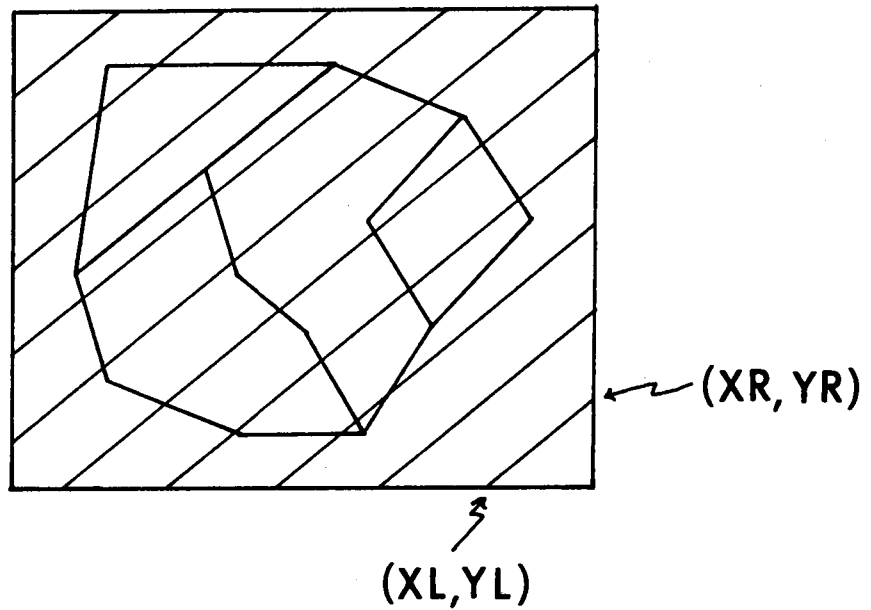
Case 4—diagonals at 135 degrees

Figure 18. Polygons with diagonal overlays.

The four cases of diagonals have to be solved separately. In Case 1 of 90° or vertical diagonals, the program evenly divides the X axis into the number of diagonals plus one increment. The program starts at the left hand side or the minimum X axis and adds this increment in a loop which is controlled by the user's parameter input of the number of diagonals. The values are stored in an array called DIAGLR(50,4) where the maximum number of diagonals is 50. The XY pairs for the two end points are in columns 1, 2 and 3, 4 respectively (Figure 19). Case 2 of 0° or row diagonals follows this same logic. Case 3 of diagonals at 45° starts at the lower right hand corner. The increment XDIV for spacing along the X-axis is equal to the width of the rectangle plus the length of the rectangle divided by the number of diagonals. With this increment, the Y increment is the same for 45° . Flags are used to determine when the increments go past the corners of the rectangle, lower left and upper right, and where adjustments must be made. The scheme behind all these adjustments for both Case 3 and Case 4 follow the same logic. Figure 20 illustrates this point. The value $SX = SDIV - (DIAGLR(J,I) - XMIN)$ and $SY = \tan(45^\circ) SX$. With this measurement added to YMIN, the next diagonals end point has been computed correctly. Case 4, diagonals at 135° , is computed using this same logic. The only difference being that plotting starts in the lower left hand side. The corners where flags are used are the upper left and lower right (Figure 18).

D. POLYGONS AND DIAGONAL INTERSECTIONS

To convert the two-dimensional data into a three-dimensional plot, the point where the diagonals and the segment of each ploygon intersect



DIAGLR(1)		$\bar{X}_L, Y_L, X_R, Y_R$			
N # of diagonals	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	(N)	.	.	.	.

Figure 19. Data storage of the end points of the diagonals.

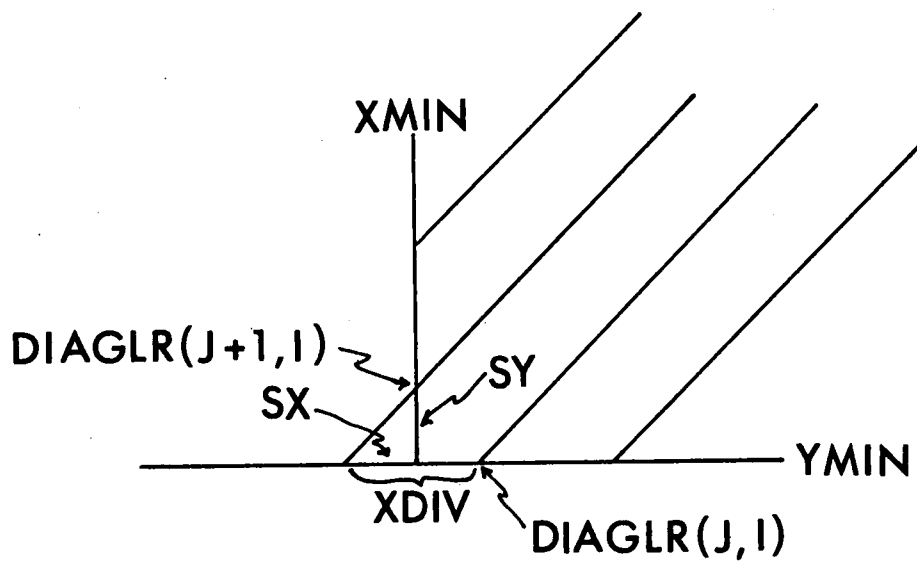


Figure 20. Diagonal incremental corrections for corners.

must be raised to the corresponding data value of that polygon. The subject of this section is to describe the method by which these intersections are found.

Each diagonal and each segment of every polygon is passed to a SUBROUTINE called INTER which returns information of the XY intersection, if it occurs. If there is an intersection, DISCON will enter that XY intersection into an array named RPOLY(I,J,L) where I is the row, J is the column, and L is the diagonal number. RPOLY(50,4,50) can handle 50 intersections per diagonal with a total of 50 diagonals. This dimension can be changed by increasing the array size. RPOLY(50,4,50) will be used as an example throughout this thesis. In the column entries, the X value is in column 1, Y value in column 2, polygon data value in column 3, and the polygon number in column 4 (Figure 21). After all the segments of each polygon have been tested, DISCON will enter the two end points with polygon number zero and then a flag of 9999. to show the finish of that diagonal. DO loops are controlled by the user's parameter inputs for number of polygons, number of diagonals, and the number of segments per polygon.

One restriction on the data is that once the diagonal enters the two-dimensional surface of polygons it cannot exit the surface and reenter. The way to get around this problem is to introduce dummy polygons with data values of zero. In this way the diagonal will find the correct number of intersections and when plotting takes place the correct height adjustment of zero will render a true representation.

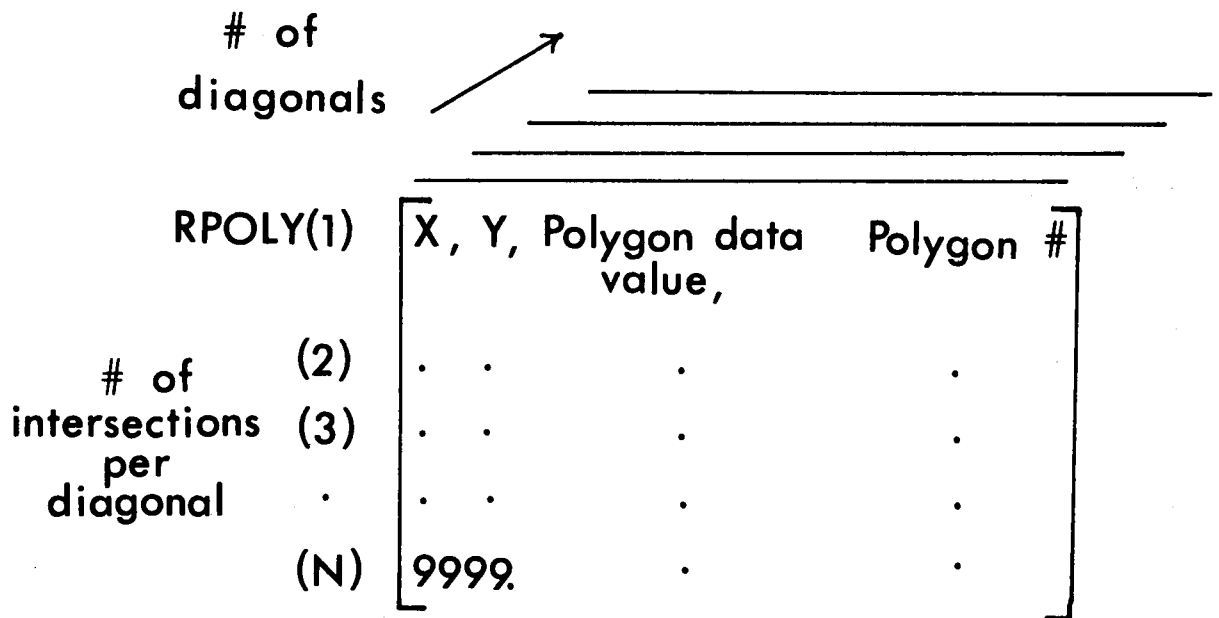


Figure 21. Data storage of the intersections along each diagonal.

SUBROUTINE INTER computes intersections of the diagonals with the polygon boundary segments. DISCON passes two line segments as defined by four end points, to INTER. INTER returns the XY point of intersection if the logical variables LINT (line intersection) is 1 or if LPAR (line parallel) is 1. To determine if there is an intersection a crude MINIMAX test is applied first to determine if the two line segments would have any points in common. If this test fails, INTER returns to DISCON indicating no intersection. If the test does not fail, the slopes of both lines are found by the comparison of the difference in Y values to the difference in X values.

$$\text{Slope} = \frac{\text{difference in Y-values of the two end points}}{\text{difference in X-values of the two end points}}$$

Tests for the cases where either of the line segments is horizontal, vertical or both are performed and flags set. If none of these cases exist, then the B-intercept is found for each line segment by substituting one of the end points of each line segment in the following equation: $B = Y - M \cdot X$ where B is the B-intercept, M is the slope, X and Y are the values of an end point. With the B-intercept for both segments, the X value of the intersection is just the difference of the B-intercepts divided by the difference in slopes. With the X value of the intersection, the Y value is found by substitution in the basic formula $Y = M \cdot X + B$. After these calculations, there is a check to make sure the intersection value of X and Y are really on one of the line segments and these results are returned to DISCON. There are four special cases which follow (Figure 22):

	LINE SEGMENT 1				LINE SEGMENT 2			
	nonhorizontal nonvertical nonparallel		parallel to Line 2		nonhorizontal nonvertical nonparallel		parallel to Line 1	
	horizontal	vertical	horizontal	vertical	horizontal	vertical	horizontal	vertical
case 1								
case 2	X			X			X	
case 3		X		X	X			
case 3		X		X				
case 3								
case 3	X							
case 3							X	
case 4								X

Figure 22. Special cases for line segment intersections.

- (1) if both lines are horizontal, check to see if they are the same;
- (2) if both lines are vertical, check to see if they are the same;
- (3) if one of the two lines is vertical it has no B-intercept; and
- (4) if the two lines are parallel with the same slope, but neither horizontal nor vertical. Each of these special cases is solved by means of variations of the basic formulae already mentioned.

E. SORTING AND DELETION OF DUPLICATE ENTRIES ON DIAGONALS

Arranging the intersection in an ascending order is necessary because plotting is done along the diagonal starting at one end point and working toward the other. Except for the outer edges of the rectangle there are two entries for each intersection: one where a polygon is exited and another where the next polygon is entered.

Sorting is accomplished by using a bubble sort to sort first on the X value in cases of 0° , 45° , and 135° diagonals. In the case of vertical or 90° diagonals the bubble sort is initiated on the Y values. This bubble sort is inefficient because it only sorts in one direction without setting any flags. This sort was used to solve the sorting problem so work could be started on the plotting and hidden-line problems of this thesis.

The next step is the deletion of duplicate entries. These duplicate entries arise when multiple defined end points intersect one of the diagonals. These end points occur when two segments of the same polygons meet. The test is to compare all three entries of X, Y and polygon numbers in `RPOLY(I,J,L)` to determine if duplicate entries have been

inserted. If a duplication occurs, delete one and move everything up one row for that diagonal.

The final step is to arrange the X, Y and polygon number entries in the correct sequence before sending them to the hidden-line algorithm. This sequence is where the end point of the diagonal is followed by two entries for each XY intersection between the diagonal and polygon segments. These XY points are arranged in an ascending order along that diagonal. The arrangement of the entries must be in the order where two XY paired entries are identical. The first being for the exit of the polygon and the second pair being the entrance of the next polygon in order along the diagonal. The last entry along each diagonal is the end point of that diagonal. This sorting is accomplished by keeping track of three values of the polygon number, the entry which is being examined and the entries preceding and after that entry. By comparing the polygon number along with the XY pairs, sorting the entries is just a matter of arranging the entries in the required manner so plotting can be accomplished in the hidden-line algorithm.

When complete, this file is stored for another interactive plot or may be passed to the rotation step of the program. This file of intersections is fixed as the number of diagonals, choice of vertical, or diagonal lines, and the width of the border around the extreme points of the coordinates. This file, however, can be rotated and given a certain viewing angle so the file of intersection will be maintained for subsequent use in lieu of the coordinate files of the polygons during the user's execution of DISCON.

The next step is to count the number of entries for each diagonal and save that number in an array called LINE(L) where L is the number of the diagonal. This information is used in control of the plotting routine.

F. X - Y - Z ROTATION

Rotation in three-dimension is accomplished by first rotating about any XY point in the XY data base file RPOLY(50,4,50). The next rotation is about the Y-axis will result in tilting the image left or right if looking straight at it. The third dimension of viewing angle is the last rotation and is accomplished in the hidden-line elimination section. It is necessary to perform first the XY rotation about any one point to preserve line ordering and line entries along each diagonal to keep track of the front edge. Homogeneous coordinates of a 4×4 nature to perform all necessary rotations in one matrix multiplication cannot be used because of the line ordering concept. Another very important reason why 4×4 homogeneous coordinates cannot be used is that for each XY point in the data base file of row I, there are two associated Z-values. One Z-value arises from the I-1 row and one from I row. With the data sorted and packed in this manner for hidden-line elimination and rotation, homogeneous coordinates of a 3×3 nature can only be used to work on the two-dimension base file of X and Y coordinates of intersections.

With the base file of the intersections of the diagonals defined, the polygons rotation around any XY point is accomplished by calling

SUBROUTINE ROTXY and passing to it the degree of rotation, the X coordinate, Y coordinate, local scaling parameters, shearing parameters, and vanishing points for the two point perspective.

Rotation is performed in the homogeneous coordinate space where the two-dimensional position vector is represented by a three-dimensional vector. Rotation is the multiplication of the matrix of XY position and one in the third column by the transformation matrix [8, 12]. Figure 23 represents this transformation matrix.

$$\begin{bmatrix} A & B & P \\ C & D & Q \\ M & N & S \end{bmatrix}$$

Figure 23. Transformation matrix.

From Figure 23, A and D cause scaling of the initial object in an X- and Y-direction respectively. B and C cause a shearing of the initial object in a Y- and X-direction respectively. When $A = \cos\theta$, $B = \sin\theta$, $C = \sin\theta$, and $D = \cos\theta$, $P = 0$, $Q = 0$, the transformation matrix produces a pure rotation through the angle θ around point M,N. The third column of P, Q, and S is added so matrix multiplication can be performed correctly by use of this square matrix. P and Q produce a projection while S produces overall scaling. With $P = 0$, $Q = 0$, and $S = 1$ there is no projection and scaling is unchanged. M and N produce the translation of the position vector [8, 12].

To accomplish rotation around an arbitrary point, first the center of rotation must be translated to the origin, the required rotation

performed, and then translate the results back to the original center of rotation [12]. The rotation of position vector $[X \ Y \ 1]$ about point M,N through an arbitrary angle is shown in Figure 24. By carrying out the two interior matrix products in Figure 24, the transformation matrix is developed (Figure 25). With this transformation matrix, where $M = -X \text{ POINT}(\cos\theta - 1.) + Y \text{ POINT}(\sin\theta)$, $N = -X \text{ POINT}(\sin\theta) - N(\cos\theta - 1)$, and the preceding substitutions, the rotation through θ around an arbitrary point is accomplished in SUBROUTINE ROTXY.

When rotation occurs, there are certain cases where the ordering of the entries and line ordering in RPOLY(I,J,L) may have to be reversed. The ordering is accomplished in two stages in SUBROUTINE CHANGE. In the first stage, CHANGE takes line number 1 and makes it the last line. The last line becomes the first line plotted. The order of the line numbers is reversed. In the second stage, the entries of each line are reversed so the first is the last and the last is the first. From Figure 26 it can be determined for what cases CHANGE must be called by examining the far right column with the particular diagonal and rotation used. A simple example clarifies the concept. Diagonals at 45° with rotation of 100° . From Figure 27 the line order and entry order need to be reversed which is accomplished through the SUBROUTINE CHANGE.

After the steps of rotation about an arbitrary point and line ordering correction have been completed, rotation about the Z-axis in three-dimensional space (tilt) is performed. The first step of this rotation is to determine the largest X value. The next step is to take each X point of each XY pair and add to the Y component a scaled amount depending on the ratio of the X point to the largest X value.

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -M & -N & 1 \end{bmatrix} \begin{bmatrix} (A) (\cos \theta) \\ (C) (-\sin \theta) \\ 0 \end{bmatrix} \begin{bmatrix} (B) (\sin \theta) \\ (D) (\cos \theta) \\ 0 \end{bmatrix} \begin{bmatrix} P \\ Q \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ M & N & 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ H \end{bmatrix}$$

Figure 24. Rotation of position vector about the point M,N through an arbitrary angle. Adapted from Rogers and Adams, p. 43 [12].

$$\begin{bmatrix} X \\ Y \\ H \end{bmatrix} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \begin{bmatrix} (A) (\cos \theta) + (P) (M) \\ (C) (-\sin \theta) + (Q) (M) \\ \{ (-M) (A) (\cos \theta) + (N) (C) (\sin \theta) \\ + (M) ((-M) (P) - (N) (Q) + 1) \} \end{bmatrix} \begin{bmatrix} (B) (\sin \theta) + (P) (N) \\ (D) (\cos \theta) + (Q) (N) \\ \{ (-M) (B) (\sin \theta) - (N) (D) (\cos \theta) (-M) (P) - (N) (Q) + 1 \} \\ + (N) ((-M) (P) - (N) (Q) + 1) \end{bmatrix}$$

Figure 25. Transformation matrix produced by carrying out the two interior matrix products of Figure 24. Adapted from Rogers and Adams, p. 44 [12].

TYPE OF DIAGONAL	ROTATION IN DEGREES	PLOT AS 45° or 135°	CHANGE ORDER YES OR NO
0. degrees	-270 to 360, 0 to 90	45	NO
	-180 to 270, 90 to 180	135	YES
	- 90 to -180, 180 to 270	45	YES
	0 to - 90, 270 to 360	135	NO
45. degrees	-315 to -360, 0 to 45	45	NO
	-225 to -315, 45 to 135	135	YES
	-135 to -225, 135 to 225	45	YES
	- 45 to -135, 225 to 315	135	NO
	0 to - 45, 315 to 360	45	NO
90. degrees	-270 to -360, 0 to 90	135	YES
	-180 to -270, 90 to 180	45	YES
	- 90 to -180, 180 to 270	135	NO
	0 to - 90, 270 to 360	45	NO
135. degrees	-315 to -360, 0 to 45	135	NO
	-225 to -315, 45 to 135	45	NO
	-135 to -225, 135 to 225	135	YES
	- 45 to -135, 225 to 315	45	YES
	0 to - 45, 315 to 360	135	NO

Figure 26. Diagonal plotting and change chart.

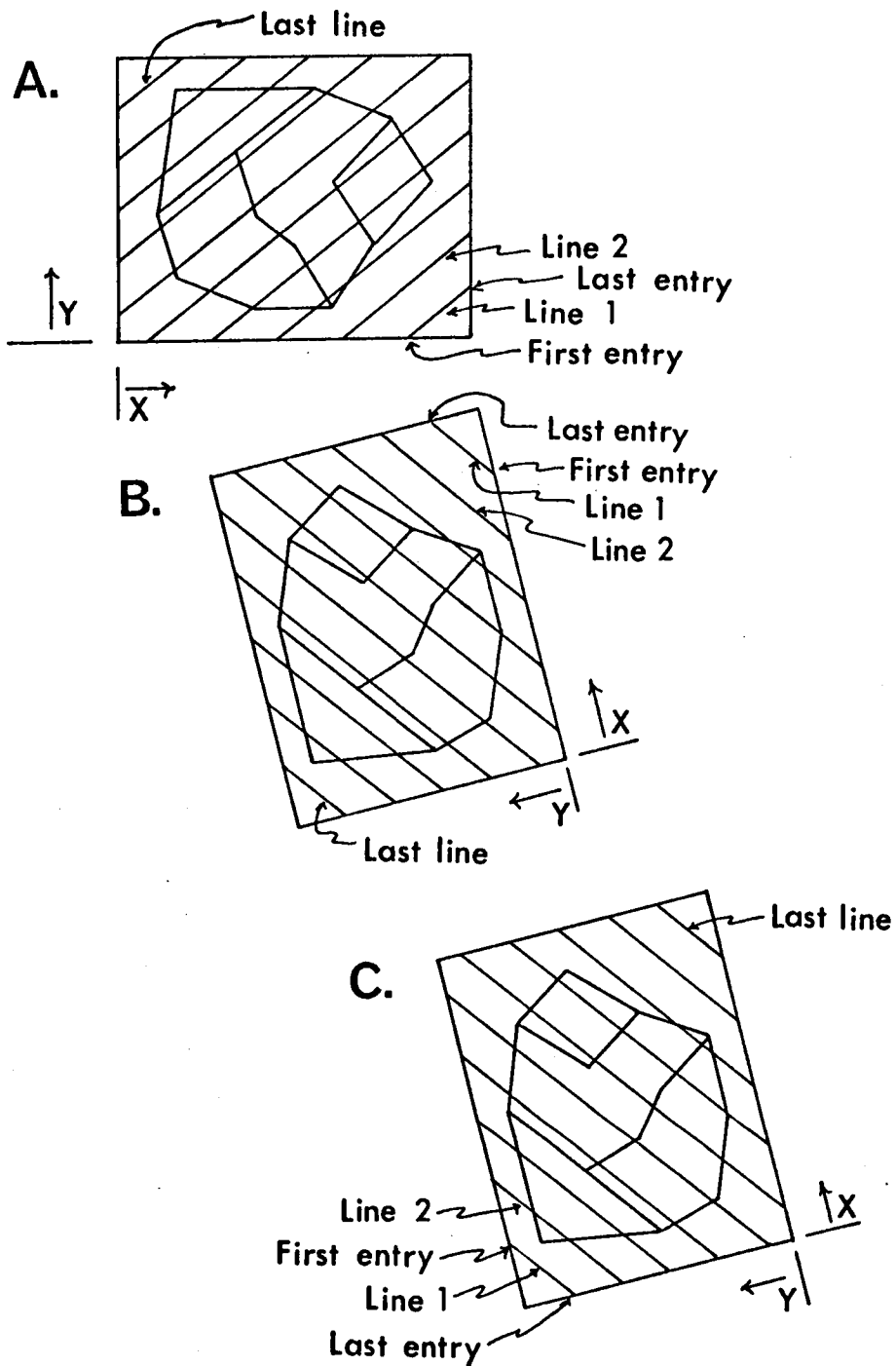


Figure 27. Example: Diagonals at 45 degrees and rotation of 100 degrees to show the effect of SUBROUTINE CHANGE. A. Before rotation. B. After rotation, before SUBROUTINE CHANGE. C. After rotation, after SUBROUTINE CHANGE.

Viewing angle or rotation about the X-axis is performed in the hidden-line elimination section. This is accomplished by taking the starting Y-point of a vertical segment (YN) and adding to it a corresponding portion of the Z-segment (ZN) using the following equation:

$$Y = \text{SIN}(YN) + \text{COS}(ZN)$$

Since the viewing rotation does not change the internal data set, no computations have to be reversed for this rotation. With the rotation about the Z-axis in three-dimensional space (tilt), the corresponding amount that was added to each Y point has to be subtracted which is the same computation except instead of adding the scale factor, it is subtracted. To get back to the original data from an XY rotation about an arbitrary point, DISCON sends to ROTXY the negative value of that rotation and the same M,N point which it was rotated about. Now the user can interactively input new data and change parameters to see a new plot from the same XY base file of the intersection and diagonals.

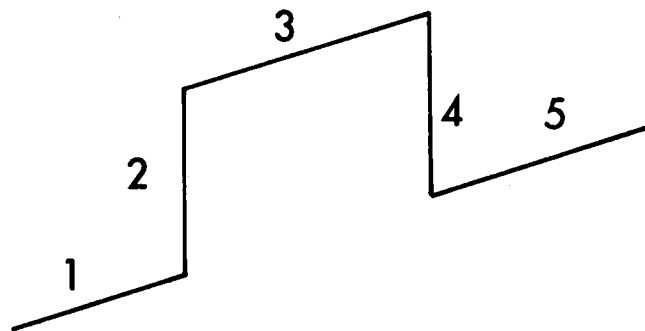
G. HIDDEN-LINE ELIMINATION ALGORITHM

Hidden-line elimination is performed through the use of keeping track of the visible top line. Since plotting begins in the lower front right corner for 45° diagonals and in the lower front left corner for 135°, it is only necessary to keep track of the segments that make up the top line. These segments are stored in an array called PX(150,4)

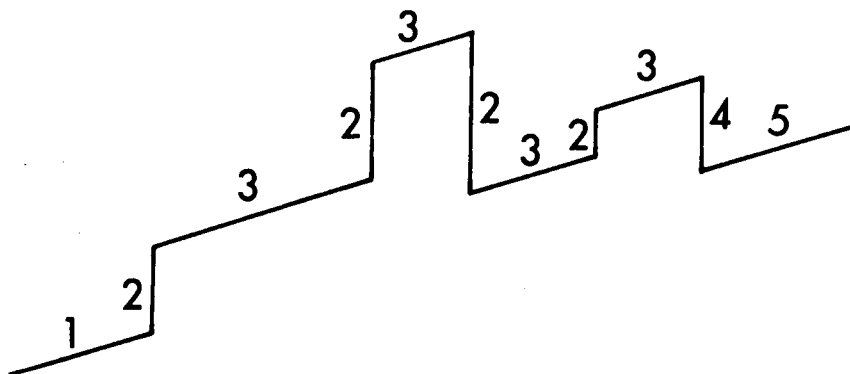
where the XY pair of the starting end point is in columns 1 and 2 respectively and the XY pair of the ending point is column 3 and 4 respectively. It is possible to store 150 such segments in this array. With the top line defined in this manner, DISCON keeps track of the existing new line being plotted as to whether it is visible to start with or invisible. If it is visible and then becomes invisible (i.e., crosses the top line), then PX(150,4) must be updated accordingly. If it is invisible, becomes visible, and then becomes invisible, PX(150,4) is updated. In this case the visible portion has been stored in PVISIB(80,4). This data structure is the same as PX(150,4), which must be inserted in PX(150,4) where the visible portion is correctly linked with the existing top line to eliminate the old visible portion of the top line because it is now invisible. The updating of the new line in this manner is accomplished in SUBROUTINE UPDATE. If there is no crossing of the top line with the existing line being plotted and the line was visible to start with, then the SUBROUTINE UPDATE is called which replaces that line as the new top line.

Plotting begins with the lower front corner. Plotting is always done from left to right. The calls to the plotting device are minimized by plotting long vectors, rather than increments as is done in most matrix-based programs. Plotting is done in six basic steps:

- (1) draw all diagonals with only two end points, that is only the diagonals that do not intersect any of the polygonals,
- (2) initiate the height of the intersections and check to see if the starting end point is visible
- (3) draw segment 1 of Figure 28,
- (4) draw segment 2 of



a. Simple case—one polygon.



b. Complex case—four polygons.

Figure 28. Plotting sequence.

Figure 28, (5) draw segment 3 of Figure 28, and (6) test to see if this is the last polygon for this diagonal, if not go to (4), if it is last, draw segments 4 and 5 of Figure 28, increment to next line, go to (2). Before each draw command, the segment is passed to SUBROUTINE HIDDEN which passes that segment with each segment of the top line to SUBROUTINE INTER to check for an intersection. When an intersection occurs, depending on whether the starting point was visible or not, the appropriate updating of the top line is necessary. Visibility occurs when the segment being plotted is above the visible top line. In this manner the top line is kept up to date and plotting continues until all diagonals have been plotted.

From this point the user can terminate the program or change the parameters and view another plot with this same base file of intersections between the diagonals and polygons.

H. HARDWARE AND SOFTWARE

DISCON was implemented on the UTCC DECsystem-10 Model 1080 which runs under TOPS-10 with Galaxy-10. This 36 bit machine has 256,000 words of memory. DISCON runs on the TEKTRONIX 4010, 4014, or 4015 which are all connected to the timesharing system at the University of Tennessee. These TEKTRONIX terminals are capable of displaying both alphanumeric characters and graphical data. Once written, the display remains visible until it is erased, and it is not necessary to continually refresh the information put upon the screen [10]. Hard copies can be obtained from either the TEKTRONIX 4963 Hard Copy Unit or the TEKTRONIX 4662 Plotter.

DISCON is written in machine-independent FORTRAN IV and utilizes TEKTRONIX Plot-10 Terminal Control System. This package aids in the drawing of bright and dark vectors (line segments) as well as points which may be displayed on the screen. A bright vector, one generated by "draw" routines, can be seen on the screen while a dark vector, one generated by "move" routines, is invisible. The routines which DISCON calls are as follows: (1) INITT—INitialization of the Terminal and the Terminal Status Area, (2) DWINDO—Define the Vertical WINDOW, (3) MOVEA—MOVE Absolute, (4) DRAWA—DRAW Absolute line drawing, (5) DASHA—DASH Absolute line drawing, (6) FINITT—Finish INITTialization or terminate program, (7) ERASE—ERASE the screen. With this routine DISCON can address the screen coordinates between 0 and 1023 on the X-axis and between 0 and 1023 on the Y-axis with visible portion of Y being 0 to 779. Also included in the TCS package are a choice of linear, logarithmic, or polar coordinate systems, automatic scaling of graphic data, and buffered input and output for faster, more efficient character handling [10].

CHAPTER IV

CONCLUSIONS

The goal of this thesis was to develop an interactive computer program to represent with parallel lines on the CRT a three-dimensional representation from two-dimensional data interactively since most existing computer generated line block diagrams do not accurately represent discontinuous surfaces. DISCON effectively produces such block diagrams. This problem is illustrated in Figure 29 by use of the population data for the State of Tennessee by counties. As can be seen from this figure, this plotting procedure does represent vertical lines as needed to correctly represent county units. This problem arises since only one value may exist for a given XY position in the particular matrix used to construct this model. To correct this misrepresentation of discontinuous surfaces (county units in this case), a vertical line must be perpendicular to the XY plane, and be determined by the two end points. DISCON, as seen in Figure 30, represents such surfaces in this way. Other examples of DISCON, such as rotation about X, Y, and Z planes are illustrated in Figures 31 through 34.

Another important area where DISCON is more effective than existing matrix-base programs deals with the final use of its file RPOLY(I,J,L). This file holds the intersections between the diagonals and polygon segments. Once RPOLY(I,J,L) has been created, it can be rotated and given a desired viewing angle. This file can be maintained for subsequent use if desired by the user. In this respect, DISCON has some similarity

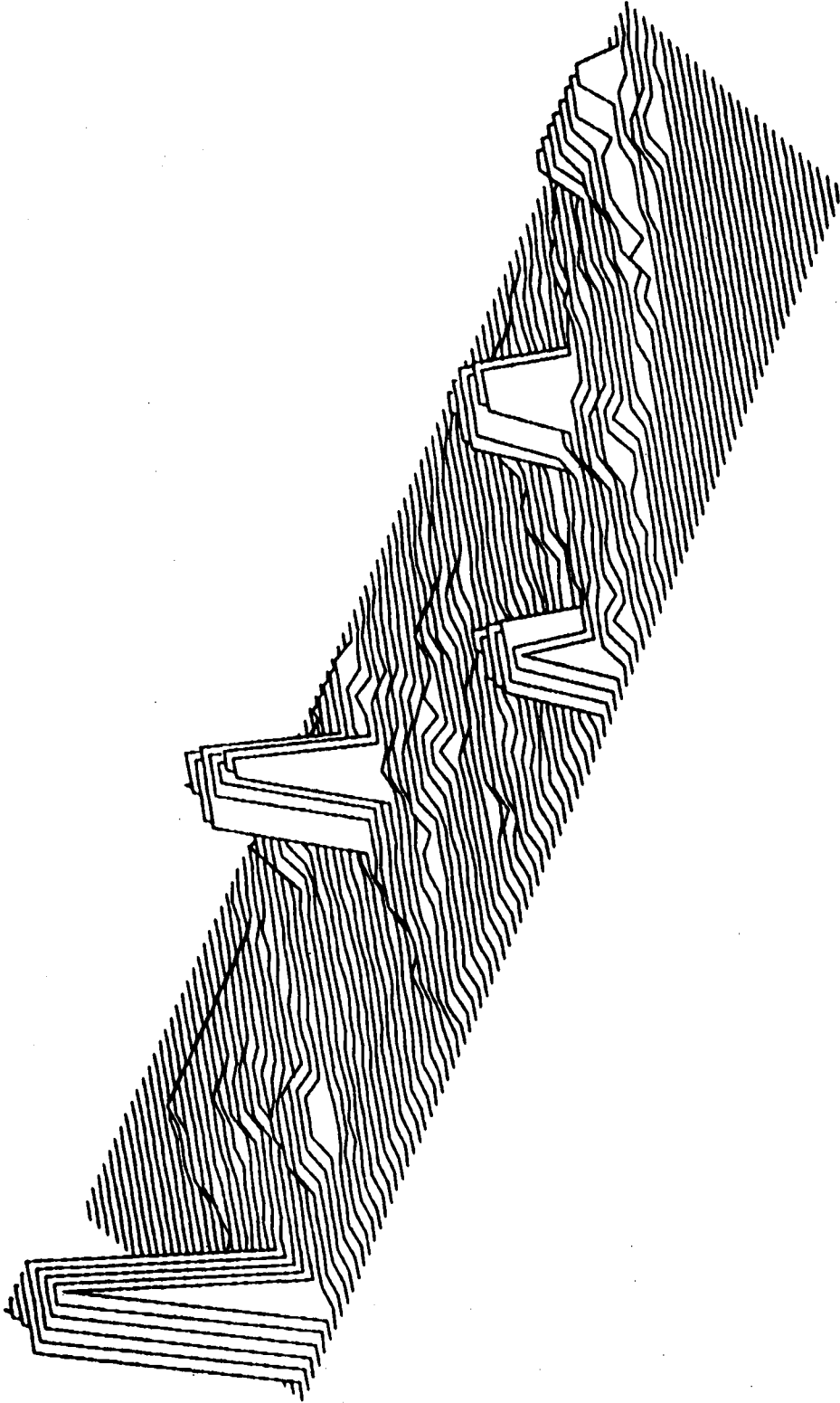


Figure 29. Population of Tennessee by counties represented by ASPEX using a matrix generated by SYMAP [courtesy Mike Hodgson, University of Tennessee].

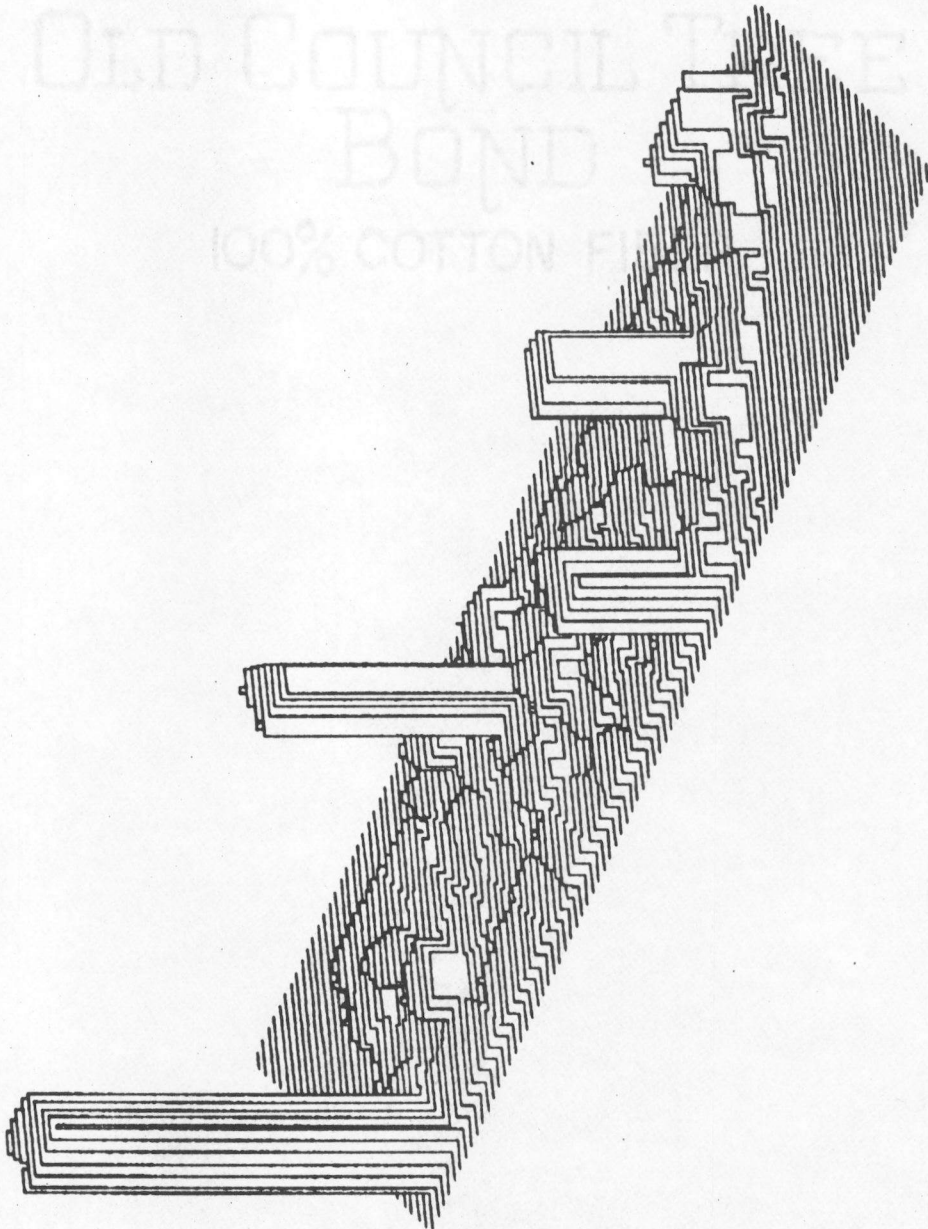
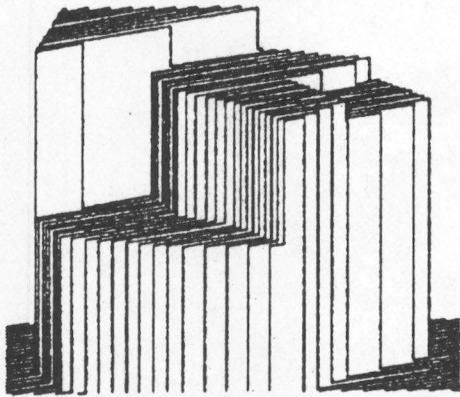
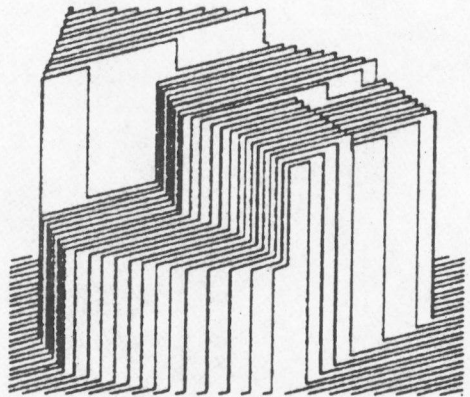


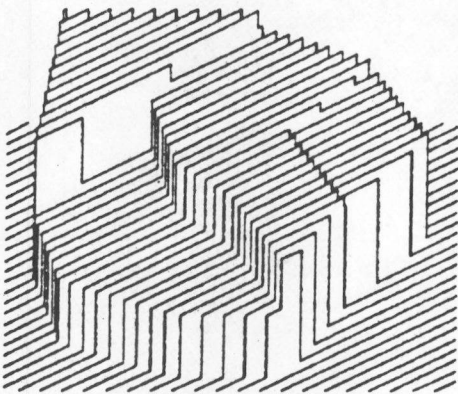
Figure 30. Population of Tennessee by counties represented by DISCON.



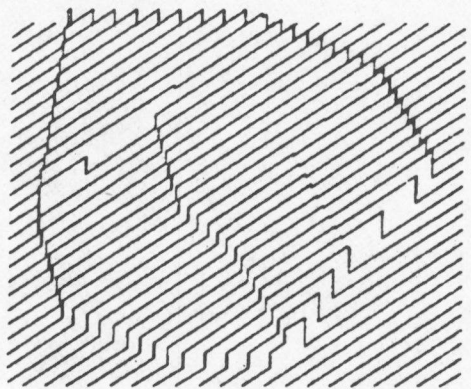
a. Viewing angle is 15.00.



b. Viewing angle is 30.00.



c. Viewing angle is 65.00.



d. Viewing angle is 85.00.

Figure 31. Examples of rotation about X-axis (viewing angle) with diagonals of 45 degrees.

There are 45 diagonals.

There are 4 polygons.

Type of columns or rows is: 3.

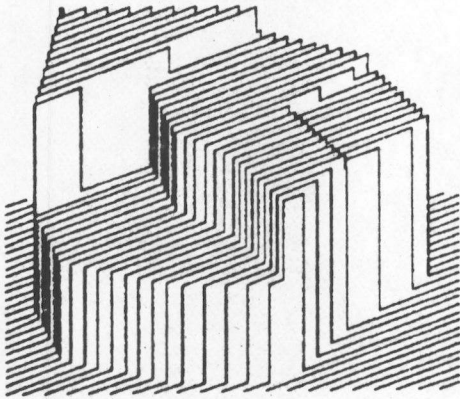
Local scaling in an X- and Y-direction, respectively: 1.00 1.00.

Shearing parameters in a Y- and X-direction, respectively: 1.00 1.00.

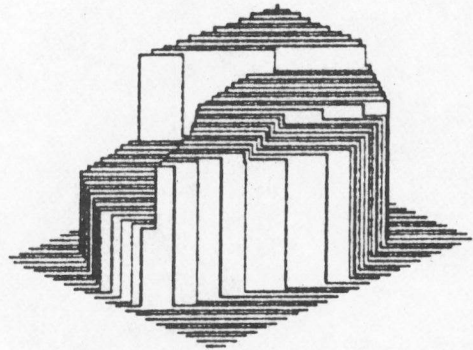
Vanishing Points for X- and Y-axis, respectively: 0.00 0.00.

Overall scaling is: 0.30.

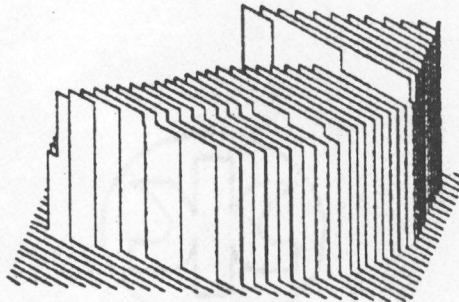
No rotation about Y- or Z-axis.



a. No rotations.



b. Rotate -45.00 degrees
about point 0.,0.



c. Rotate -100.0 degrees
about point 0.,0.

Figure 32. Examples of rotation about Y-axis with diagonals of 45 degrees.

There are 45 diagonals.

There are 4 polygons.

Type of columns or rows is: 3.

Local scaling in an X- and Y-direction, respectively: 1.00 1.00.

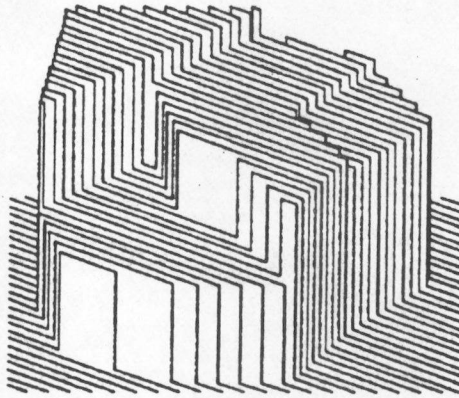
Shearing parameters in a Y- and X-direction, respectively: 1.00 1.00.

Vanishing points for X- and Y-axis, respectively: 0.00 0.00

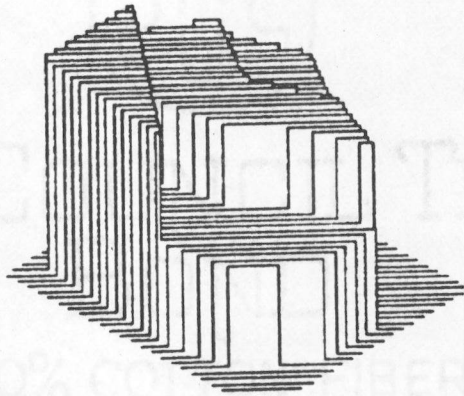
Overall scaling is: 0.30.

Viewing angle: 45.00.

No rotation about the Z-axis.



a. No rotation about Y- or Z-axis.



b. Rotate 45 degrees about point 0.,0.
No rotation about Z-axis.

Figure 33. Examples of rotation about Y-axis with diagonals of 135 degrees.

There are 45 diagonals.

There are 4 polygons.

Type of Columns or rows is: 4.

Local scaling in an X- and Y-direction, respectively: 1.00 1.00.

Shearing parameters in a Y- and X-direction, respectively: 1.00 1.00.

Vanishing point for X- and Y-axis, respectively: 0.00 0.00.

Overall scaling is: 0.30.

Viewing angle is: 45.00.

Figure 34. Examples of rotation about Z-axis or tilt.

There are 45 diagonals.

There are 4 polygons

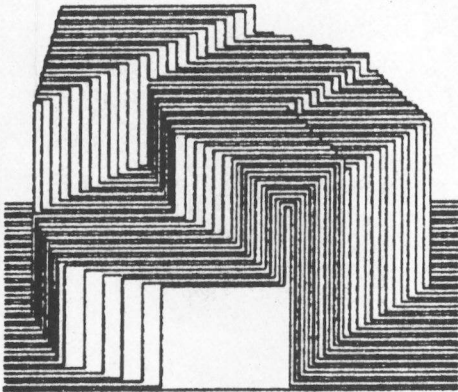
Type of columns or rows is: 2.

Local scaling in an X- and Y-direction, respectively: 1.00 1.00.

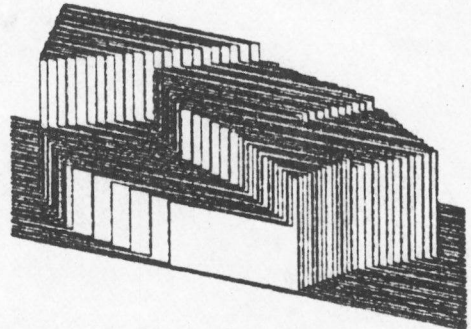
Shearing Parameters in a Y- and X-direction, respectively: 1.00 1.00.

Vanishing points for X- and Y-axis, respectively 0.00 0.00.

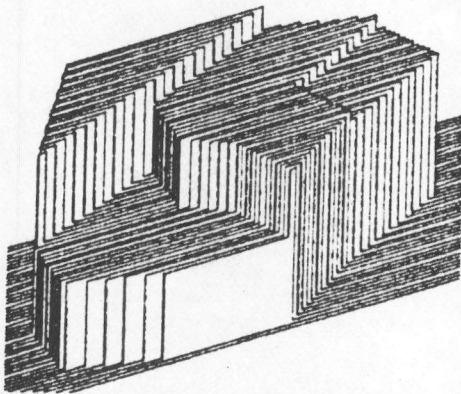
No rotations about Y-axis.



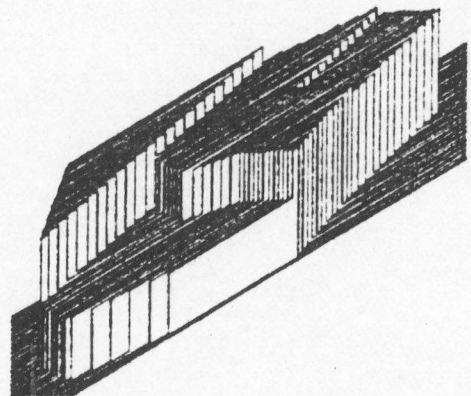
a. Overall scaling is 0.30.
Viewing angle is 45.00.



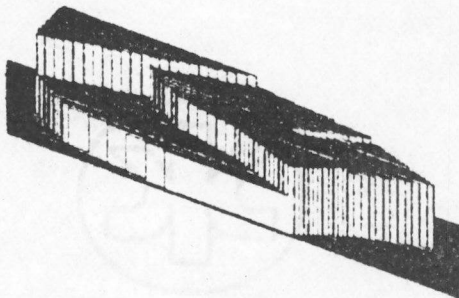
b. Overall scaling is 0.30.
Viewing angle is 45.00.
Rotation about Z-axis is 45.00.



c. Overall scaling is 0.30.
Viewing angle is 45.00.
Rotation about Z-axis is -45.00.



d. Overall scaling is 0.30.
Viewing angle is 45.00.
Rotation about Z-axis is -70.00.



e. Overall scaling is 0.30.
Viewing angle is 45.00.
Rotation about Z-axis is 70.00.

Figure 34.

to the block diagrams programs that work from a matrix, like SYMAP/SYMVU. During an intermediate stage, both produce a file at a fixed level of resolution that can be stored and subsequently transformed. In the matrix-type programs, however, the matrix contains data for one variable and if other values are to be displayed, a new matrix must be created. In DISCON, the line-by-line file RPOLY(I,J,L) of polygon intersections is independent of any data values and any data may be used with such a file. Once this file is plotted by the hidden-line elimination section of DISCON, the user can have different polygon data assigned and see the respective plot. This procedure saves computation time because the file RPOLY(I,J,L) can be reused as many times as required by the user.

Plotting is done in a unique way by DISCON. Calls to the plotting device are minimized by plotting long vectors, rather than increments as is done in most matrix-based programs. This form of plotting can best be appreciated when plotting is done on a time-sharing basis over a low BUAD-rate line.

There are three basic limitations of the program DISCON. The first limitation deals with the insertion of dummy polygons with the value of zero. These polygons are required to make sure whenever a diagonal enters the two-dimensional figure, that a diagonal cannot leave that figure until it reaches the last polygon along that diagonal. The responsibility is on the user to know his own data and to make sure dummy polygons are constructed in a unique way so as to have a successful completion of the diagonal overlays. To correct this limitation, major rewriting of the program is required and not feasible.

The second limitation deals with the bubble sort which is used in sorting the entries along each diagonal. The bubble sort takes approximately N^2 operations to sort N entries, but is performed on X-values, Y-values, and polygonal number. This sort is slow and inefficient because it sorts in only one direction and no flags are set. This sort can be improved by using a bubble in both directions with flag setting to determine when the sort is finished. The third limitation deals with the nature of the user's data. If the user's data is elongated in one direction, for example the State of Tennessee in the X-direction or east-west axis, this can cause a problem. The problem occurs when diagonals of 0 degrees are used. There are too many intersections between the diagonal overlays and polygon segments for the array RPOLY(50,4,50) to handle. It is up to the user to keep this limitation in mind if the data which is being used is elongated in one direction or change the dimension of the arrays. The dimensions of RPOLY(50,4,50) were increased to RPOLY(50,4,120) to produce Figure 30, page 63.

Three features are unique to this program. First, the program DISCON plots two Z-values for each XY intersection which results in the true representation of the vertical lines from the discontinuous surfaces. Second, the matrix of the intersections has the capability to be reused with new data and different parameters about Z, Y, and Z rotations to render perspective plots of the surface. Third, the problem of hidden-line elimination is solved by the visible top line concept and vector plotting versus incremental plotting. The significance

of this work is an efficient algorithm to produce a three-dimensional diagram from two-dimensional data with hidden-lines removed. Also, this program has the capabilities of changing the parameters interactively and seeing another view of the three-dimensional surface.

LIST OF REFERENCES

LIST OF REFERENCES

1. Bouknight, W. J. "A Procedure for Generation of Three-Dimensional Half-Toned Computer Graphics Representation," Comm. ACM, 13, 9, (Sept. 1970), 527.
2. Bouknight, W. J., "An Improved Procedure for Generation of Half-Tone Computer Graphics Representation," University of Illinois, Coordinated Science Laboratory, R-432 (Sept. 1969).
3. Carter, J. R. "A Collection of Computer Mapping Manuals: CMAP, SYMAP, FLEXIN, CALFORM, and CALSYM, and an Introduction to CAM and THE WORLD DATA BANK," Geography Department, University of Tennessee, Sept. 1975.
4. Cohen, D. "Incremental Methods for Computer Graphics," ESD-TR-69-193, Harvard University, April 1969 (Ph.D. Thesis).
5. Floyd, Robert W. "Nondeterministic Algorithms," Journal of the Association for Computing Machinery, XIV, No. 4 (October 1967), 637-644.
6. Galimberti, R., and Montanari, U., "An Algorithm for Hidden-Line Elimination," Comm. ACM, 12, 4 (April 1969), 206.
7. Gonzalez, R. C., and Wintz, P. W. Digital Image Processing. Addison-Wesley Publishing Company, Reading, Massachusetts, 1977.
8. Jolley, S. P. "An Input System for the Creation of Three-Dimensional Shaded Graphics Pictures," University of Tennessee, December 1977 (Master's Thesis).
9. Newman, W. M., and Sproull, R. F. Principles of Interactive Computer Graphics. McGraw-Hill Book Company, New York, New York, 1973, p. 223-332.
10. PLOT-10: Terminal Control System User's Manual. Information Display Division, Tektronix, Inc., Beaverton, Or., Doc. No. 062-1494-00, 1974.
11. Prueitt, M. L. Computer Graphics 118 Computer-Generated Designs. Dover Publications, Inc., New York, New York, 1975.
12. Rogers, D. F., and Adams, J. A. Mathematical Elements for Computer Graphics. McGraw-Hill Book Company, New York, New York, 1976.
13. Sproull, R. F., and Sutherland, J. E. "A Clipping Divider," Proceedings of the Fall Joint Computer Conference, 1968 Conf., Vol. 33, Part 1 (1968), 765-776.

14. Sutherland, J. E., Sproul, R. F., and Schumacker, R. A. "A Characterization of Ten Hidden-Surface Algorithms," ACM Computing Surveys, 6, 1 (1974), 1-55.
15. Tobler, W. R. "A Computer Program to Draw Perspective Views of Geographical Data," The American Cartographer, 1, 2 (1974), 124.
16. Warnock, J. E. "A Hidden-Surface Algorithm for Computer-Generated Half-Tone Pictures," Computer Science Department, University of Utah, TR-4-15, June 1969.
17. Watkins, G. S., "A Real-Time Visible Surface Algorithm," Computer Science Department, University of Utah, UTECH-CSC-70-101, June 1970.

APPENDICES

APPENDIX A

GLOSSARY

Glossary

Algorithm—the way a problem is solved: different algorithms may be used to do the same thing: a basic term for any computer programmer.

Area Coherence—a particular element of the output picture and its neighbor on all sides is likely to be influenced by the same face.

ASPEX—a program to draw block diagrams on a CRT or plotter: program is interactive in that you feed the program information as the program is running: will draw lines along rows, columns, or diagonals but will not give a fishnet plot: new program from the Harvard Laboratory.

Back Lines or Edge—one that cannot appear in the environment being rendered because it lies on the side of an object away from the observer.

Boundary Edge—Loutrel term for contour edge.

Bright Vector—one generated by a draw routine which is a visible vector.

CALFORM—a program for choropleth maps on the plotter or CRT: allows user to design and position features such as titles, scale bars, legend boxes, and north arrows: from the Harvard Laboratory.

Choropleth (Conformant) Map—the choropleth map is best suited for data, either qualitative or quantitative, which applies to an areal unit. It is assumed that the boundaries of the areal unit (polygon) are sufficient. The entire areal unit is given the same value and symbolism as assigned to the polygon according to its numeric class [3].

Clipping—segments of lines that lie off the screen or behind the observer are eliminated.

Cluster—collection of faces that can be treated as a group.

Constant-Depth Walk—tree transversal by FIFO (first in, first out) method.

Contour Edge—an edge that forms part of the outline of an object as seen by the observer.

Contour (Isopleth or Isoline) Map—the contour map consists of closed curves known variously as contours, isolines, isarithmic lines, isopleths, etc., connecting all points having the same numerical value. This kind of map is best employed to represent a surface which has a value at every population density [3].

CRT (Cathode Ray Tube)—a precision TV tube for output of computer information: storage tubes have one picture created on the screen and it stays there until the whole screen is erased (relatively cheap): refresh tubes have a new picture created about 60 times a second so the human does not see any flicker (relatively expensive): plasma panels, are not really CRTs but are used like CRTs in the Plato System—Panels, are a fine matrix of bubbles of gas which glow when switched on—lines and points can be selectively erased: CRTs can be obtained in color for a price.

Culling—a particular form of search, extract from a set of records all those that have a given property.

Dark Vector—one generated by a move routine which is an invisible vector.

DIME Files (Dual Independent Map Encoding)—from the U.S. Bureau of the Census: file linking street addresses to map zones or coordinates: available for most metropolitan areas, but completed at the expense of the local area so not available for all: data for an address are linked to a zone using ADMATCH.

DIMECO File—a digitalized coordinate file on magnetic tape of all of the county boundaries to the U.S.: available from the U.S. Bureau of Census.

Edge—a straight line segment connecting two adjacent vertices of a face.

Edge Coherence—the visibility of an edge changes only where it crosses another contour edge: scan lines that are next to each other are an example of this.

Frame-to-Frame Coherence—the fact that in a sequence of movie frames the successive frames are likely to be closely related.

Graphic Terminal—a CRT device with fine-enough resolution that curves look like curves and not a bunch of straight line segments although in reality in digital computers all curves are made up of straight line segments.

Homogeneous Coordinates—representing M-dimensional coordinates with $n + 1$ -component vectors for the purpose to scale arbitrarily the four components used.

Image Space—those algorithms performed where only the resolution of the screen is the limiting factor.

Line-Printer—the standard alphanumeric output device available at any computer installation: used for computer mapping by overprinting characters for darker tones.

CRT (Cathode Ray Tube)—a precision TV tube for output of computer information: storage tubes have one picture created on the screen and it stays there until the whole screen is erased (relatively cheap): refresh tubes have a new picture created about 60 times a second so the human does not see any flicker (relatively expensive): plasma panels, are not really CRTs but are used like CRTs in the Plato System—Panels, are a fine matrix of bubbles of gas which glow when switched on—lines and points can be selectively erased: CRTs can be obtained in color for a price.

Culling—a particular form of search, extract from a set of records all those that have a given property.

Dark Vector—one generated by a move routine which is an invisible vector.

DIME Files (Dual Independent Map Encoding)—from the U.S. Bureau of the Census: file linking street addresses to map zones or coordinates: available for most metropolitan areas, but completed at the expense of the local area so not available for all: data for an address are linked to a zone using ADMATCH.

DIMECO File—a digitalized coordinate file on magnetic tape of all of the county boundaries to the U.S.: available from the U.S. Bureau of Census.

Edge—a straight line segment connecting two adjacent vertices of a face.

Edge Coherence—the visibility of an edge changes only where it crosses another contour edge: scan lines that are next to each other are an example of this.

Frame-to-Frame Coherence—the fact that in a sequence of movie frames the successive frames are likely to be closely related.

Graphic Terminal—a CRT device with fine-enough resolution that curves look like curves and not a bunch of straight line segments although in reality in digital computers all curves are made up of straight line segments.

Homogeneous Coordinates—representing M-dimensional coordinates with $n + 1$ -component vectors for the purpose to scale arbitrarily the four components used.

Image Space—those algorithms performed where only the resolution of the screen is the limiting factor.

Line-Printer—the standard alphanumeric output device available at any computer installation: used for computer mapping by overprinting characters for darker tones.

List-Priority—computation is performed part in object space and part in image space.

Material Edge—Appel definition of relevant edge.

Minimax Test—eliminating whole collection of objects from comparison against other objects when their boundary boxes do not overlap.

"Nature" of the Point—Galimberti and Montanari concept of determining the set of faces that hide a point.

Object Coherence—individual bodies are confined to local volumes which may not conflict. Use of clusters is a form of object coherence.

Object Space—computations are performed to an arbitrary precision, usually the precision available in the computer executing the algorithm.

Order of Invisibility—Loutrel term for quantitative invisibility.

Path of Edges Algorithms—Watkins' definition of object-space.

Plotter—a device to draw lines on paper: controlled by the computer: may use any kind of pen or even a laser on film.

Prefix-Walk—tree transversal by LIFO (last in, first out) method.

Proximal Map—the proximal is similar in appearance to the conformant map in that the boundaries between zones are often angular. However, the map zones are not predetermined boundaries but are defined by nearest neighbor methods from point information. Each character location on the output map is assigned the value of the data nearest to it. Depending on the class intervals, boundaries are located along the lines where the values change [3].

Quantitative Invisibility—of a point is the number of relevant faces that lie between the point and the viewpoint.

Relevant Edge or Face—edge or face which survives an initial culling operation in most cases visible edges or face which are not visible have been culled out.

Scan Line Coherence—the fact that in a scene the scan lines are closely related.

Scan Lines—where the scan plane crosses the view plane.

Search—an operation that is used to identify exactly one element in a set of records.

Span—segment of a scan line.

SURFACE II—packaged program from the Kansas Geological Survey:
produces contour maps from irregularly spaced data using algorithms
selected by the user: maps may be displayed on the plotter, CRT,
or line printer: can also display the contoured surface as a
fishnet block diagram.

SYMPA—the first big packaged program: from the Harvard Laboratory for
Computer Graphics: can generate choropleth, contour, or proximal
maps: many electives are available to change scales, borders,
shadings, etc.: output is only to the line-printer.

SYMVU—Harvard's block diagram program for output to a plotter: less
interactive than ASPEX, but probably more versatile.

Tektronix—the company which makes lower-cost graphics hardware such
as the 4006, 4110, 4014, 4015, and 4051 graphics terminals.

Viewpoint—position from where an object or scene is viewed.

X-Sort—on each scan line sorting edges from left to right.

Y-Sort—edges are sorted top to bottom.

Z-Depth Search—segments that fall within a span are searched to find
which one is visible.

APPENDIX B

USER'S GUIDE AND PROGRAM LISTING

USER'S GUIDE TO DISCON

All text enclosed in {} is printed on the screen by the system. It is assumed the user is logged in. With the program DISCON and the input data files in the user's area, the program is executed by the following command:

```
.EX DISCON.FOR,/SEARCH FOR:TCS.REL
```

which links the TEKTRONIX TCS package to DISCON.

Figure 35 shows a simple case where type of input is 1, type of data input is 1, and number of polygons is 4. The four input files are:

(1) ICOOR.DAT—number of vertex points in each polygon, (2) XCOOR.DAT—X-coordinates of polygons, (3) YCOOR.DAT—Y-coordinates of polygons, and (4) ZVAL1.DAT—Z-data values of polygons. The output file FOR20.DAT is where the user's output (parameters which were specified) and any errors that might occur are printed. To get a listing on the screen for this output file the user enters:

```
.TYPE FOR20.DAT
```

To get a hard copy the user enters:

```
.PRINT FOR20.DAT
```

Examples of test Case 1 are illustrated in Figures 31-34, pages 64-68.

The program is governed by the user's parameters inputs and questions by DISCON must be answered if printed on the screen. A listing is as follows, with the necessary responses requested:

INPUT FILES

ICOOR.DAT

Number of vertex points in each polygon

Polygon #

1	5
2	9
3	5
4	8

XCOOR.DAT

X-coordinates of polygons

1	10.	8.5	7.5	9.0	10.				
2	9.0	7.5	8.5	7.5	6.5	5.5	5.0	7.0	9.0
3	7.0	5.0	3.0	3.5	7.0				
4	7.5	5.5	3.5	3.0	5.0	5.5	6.5	7.5	

YCOOR.DAT

Y-coordinates of polygons

1	5.0	3.0	5.0	7.0	5.0				
2	7.0	5.0	3.0	1.0	3.0	4.0	6.0	8.0	7.0
3	8.0	6.0	4.0	8.0	8.0				
4	1.0	1.0	2.0	4.0	6.0	4.0	3.0	1.0	

ZVAL1.DAT

Z-data values of polygons

1	25.0
2	27.0
3	31.0
4	15.0

OUTPUT FILE

FOR20.DAT

User's Output

Parameters which were specified

Any errors, copy of screen output

Figure 35. Data structures for input and output Case 1 example.

1. {WHAT BAUD RATE? 30 or 120}

User enters 30 or 120 depending on the baud rate of the terminal being used. {ENTER DATA FOLLOWED BY CARRIAGE RETURN}

2. {WHAT TYPE OF DATA? 1 IF POLYGONAL, 0 IF CIRCULAR}

User enters a 1, if using polygonal data such as the 16 counties for East Tennessee Development Districts or 95 counties for entire state. User enters a 0, if a circular data such as for the filter plot (Figure 5, page 7).

3. {HOW MANY DIAGONALS? 1 to 50}

User enters a number between 1 and 50. This is the number of diagonals in the plot.

If polygonal data are used branch to 6.

4. {ENTER DIAMETER OF CIRCLE < 140.0 UNITS}

User enters the diameter of the circle which must be less than 140.0. The rectangle to be fitted around this circle has a length of 100.0 units (see Chapter III, Section C for description of rectangle).

5. {ENTER VALUE OUTSIDE CIRCLE, VALUE INSIDE}

User enters the desired heights for outside the circle and inside the circle (see Figure 10, page 14, for display).

GO TO 9 Flow of control branches around polygonal data.

6. {WHAT TYPE OF DATA INPUT? 1 for X, Y, DATA(I), ICOOR(I), 0 FOR X, ICOOR(I)}

User enters 1 when data inputs of X coordinates and Y coordinates of polygons, DATA(data value for polygons), and ICOOR(number of X, Y enters for each polygon) are in four separate files (see Chapter III, Section B).

User enters 0 when data X, Y coordinate inputs of the polygons and polygon data (see Chapter III, Section B).

7. {HOW MANY POLYGONS?}

User enters number of polygons in the data set.

8. {WHAT TYPE OF DIAGONALS? ENTER 1-LINE ALONG COLUMNS OF 90 DEGREES, 2-LINES ALONG ROWS OF 0 DEGREES, 3-LINES AT 45 DEGREES, 4-LINES AT 135 DEGREES}

User enters desired number to produce diagonals (see Chapter III, Section C).

9. {ENTER 1 TO SET LOCAL SCALING PARAMETER, 2 TO SET SHEARING, 3 TO SET VANISHING POINTS, 4 TO CONTINUE}
 User enters 1, 2, 3, or 4 depending on desired branch. If 4 is used default options are set, 1. for scaling and shearing, 0. for vanishing points and branch to 13.
10. {ENTER LOCAL SCALING PARAMETERS OF INITIAL OBJECT IN AN X- AND Y-DIRECTION RESPECTIVELY}
 User enters local scaling parameters greater than 0. If a 1. is entered, then scaling is unchanged. If the magnitudes are unequal, a distortion occurs. Numbers greater than 1. cause enlargement while numbers less than 1. cause reduction.
 GO TO 9
11. {ENTER SHEARING PARAMETERS OF THE INITIAL OBJECT IN A Y- AND X-DIRECTION, RESPECTIVELY}
 User enters shearing parameter greater than 0.. These parameters will pull the initial object in the given unit along the respective axis.
 GO TO 9
12. {ENTER VALUE OF VANISHING POINT FOR X- AND Y-AXIS RESPECTIVELY, FOR TWO POINT PERSPECTIVE}
 User enters the desired value of the vanishing point for the two-point perspective. If either one is 0., then a single-point perspective is created. If both are 0., then no perspective is created and the presentation is orthographic.
 GO TO 9
13. If data is circular or input is of case 1 (see Chapter III, Section B), GO TO 15.
14. {ENTER DATA FORMAT}
 User enters the format which he wishes the READ statements to use. With this option new data can be read in for each plot interactively.
15. {ENTER OVERALL SCALING OF FINAL BLOCK DIAGRAM $\leq 1.$ and $> 0.$ }
 User enters scaling factor which reduces final output by that desired amount.
16. {ENTER VIEWING ANGLE IN DEGREES}
 User enters viewing angle from 0.0 to 90.0. The degree angle of 0.0 is defined as looking straight at the object from ground level. The degree angle 90.0 is thus looking straight down on the object where 90.0 degrees would result in seeing just diagonals.

17. {DO YOU WANT ROTATION ABOUT THE Y-AXIS? TYPE 1 IF YES, 0 IF NO}
 User enters 1 or 0 depending if rotation of the polygon and diagonal's intersection is desired (see Chapter III, Section F).
 If rotation is not desired branch to 19.
18. {THE LIMIT OF THE X-AXIS ARE:} 2F7.2 {LIMITS OF THE Y-AXIS ARE}
 2F7.2 {WHAT ROTATION ABOUT AN X - Y POINT (F7.2) WHAT X - Y
 POINT?}
 User enters in desired rotation, -360.0 to +360.0, and what
 X - Y point rotation is to be performed on.
19. {DO YOU WANT ROTATION ABOUT THE Z-AXIS? TYPE 1 IF YES, 0 IF NO}
 User enters 1 or 0 depending if rotation about the Z-axis is
 required (see Chapter III, Section F).
 If rotation is not desired branch to 21.
20. {ENTER ANGLE OF ROTATION ABOUT Z-AXIS}
 User enters degree of rotation, -90.0 to +90.0.
 CONTINUE with computation of program. After these initial
 parameters are set up, execution of the program will continue
 until the final plot is finished. The following control
 statement then appears on the screen.
21. Enter a 1 to continue. {TYPE 1 TO EXIT, 0 TO CHANGE
 PARAMETERS AND REDRAW}
 User enters desired course of action. If 1 is entered, the
 program terminates. If 0 is entered, control of program
 goes to number 13 and normal execution follows.

```

00100 C*****
00200 C
00300 C DISCON: BLOCK DIAGRAM REPRESENTATION OF DISCONTINUOUS SURFACES
00400 C
00500 C WRITTEN BY: EDWIN F. ABERCROMBIE
00600 C JUNE 1978
00700 C
00800 C DISCON IS AN INTERACTIVE COMPUTER PROGRAM TO DRAW A THREE-
00900 C DIMENSIONAL BLOCK DIAGRAM FROM TWO-DIMENSIONAL DATA.
01000 C WITH THE PROGRAM DISCON AND THE INPUT DATA FILES(SEE CHAPTER III
01100 C SECTION B) IN THE USER'S AREA, THE PROGRAM IS EXECUTED BY THE
01200 C FOLLOWING COMMAND:
01300 C
01400 C .EX DISCON.FOR,/SEARCH FOR:TCS.REL
01500 C
01600 C USER'S PARAMETERS AND ERROR MESSAGES WILL BE LISTED IN FOR20.DAT
01700 C TO SEE THIS LISTING, EXECUTE THE FOLLOWING COMMAND:
01800 C
01900 C .TYPE FOR20.DAT
02000 C
02100 C TO GET A HARD COPY OF THIS LISTING, EXECUTE THE FOLLOWING COMMAND:
02200 C
02300 C .PRINT FOR20.DAT
02400 C
02500 C THE USER'S INTERACTION BY WAY OF THE PARAMETER INPUTS IS DESCRIBE
02600 C IN CHAPTER III SECTION A. PARAMETERS INPUTS. THE CODE
02700 C CORRESPONDING TO DIFFERENT SECTIONS OF CHAPTER III ARE MARKED
02800 C THROUGHOUT THE PROGRAM.
02900 C
03000 C SUBROUTINES CALLED:
03100 C
03200 C CHANGE-ORDERS THE DIAGONALS AND DIAGONAL ENTRIES CORRECTLY
03300 C DUE TO ROTATION
03400 C HIDDEN-FINDS AND RETURNS INFORMATION ABOUT THE INTERSECTIONS
03500 C OF THE SEGMENTS BEING PLOTTED AND THE SEGMENTS OF
03600 C THE VISIBLE TOP LINE
03700 C INTER-FINDS AND RETURNS INFORMATION ABOUT THE INTERSECTION
03800 C OF THE TWO LINE SEGMENTS PASSED TO IT
03900 C ROTXY-PERFORMS TWO-DIMENSIONAL ROTATION ABOUT AN X - Y POINT,
04000 C LOCAL SCALING AND SHEARING OF INITIAL OBJECT IN A X-
04100 C AND Y- DIRECTION, AND TWO POINT PERSPECTIVE WITH THE
04200 C VANISHING POINT ON X- AND Y-AXIS.
04300 C UPDATE-UPDATES THE VISIBLE TOP LINE
04400 C UPDT-REPLACES THE TOP LINE WITH ENTIRE LINE IN PVISIB(80,4)
04500 C SINCE NO INTERSECTION EXIST
04600 C VISIBL-SAVES THE VISIBLE PORTION OF THE SEGMENTS BEING PLOTTED
04700 C BEFORE THOSE SEGMENTS BECOME INVISIBLE
04800 C
04900 C*****
05000 C COMMON PX(150,4),PVISIB(8),4),RPOLY(5),4,50),LINE(50),LINES
05100 C LOGICAL LRX,LRY,JMPX,JMPY
05200 C DIMENSION ICODR(102),FORM(16)
05300 C DIMENSION DIAGLR(50,4),X(102,18),Y(102,18),DATA(102)
05400 C*****
05500 C
05600 C SECTION A. PARAMETERS INPUTS
05700 C
05800 C*****
05900 C WRITE(5,10)
06000 C 10 FORMAT(1X,' WHAT BAUD RATE? 30 OR 120')
06100 C READ(5,*) IBAUD

```

```

06200      CALL INITT(IRAUD)
06300      C MINTER - MAXIMUM NUMBER OF INTERSECTIONS
06400      C MLINE$ - MAXIMUM NUMBER OF LINES
06500      C IPRT - ERROR MESSAGES SENT TO FOR20.DAT
06600          MINTER=50
06700          MLINE$=120
06800          IPRT=20
06900          CALL ERASE
07000          WRITE(5,20)
07100      20  FORMAT(1X,'ENTER DATA FOLLOWED BY CARRIAGE RETURN')
07200          WRITE(5,30)
07300      30  FORMAT(1X,' WHAT TYPE OF DATA? 1 IF POLYGONAL, 0 IF CIRCLE')
07400          READ(5,*) IDATA
07500          WRITE(5,40)
07600      40  FORMAT(1X,' HOW MANY DIAGONALS? ')
07700          READ(5,*) LINES
07800          WRITE(5,50) LINES
07900          WRITE(IPRT,50) LINES
08000      50  FORMAT(1X,' THERE ARE ',I3,' DIAGONALS')
08100          IF(IDATA.EQ.1) GO TO 140
08200      C THIS SECTION OF THE PROGRAM COMPUTES DIAGONALS FOR A CIRCLE
08300      C CENTERED INSIDE A BOX, OUTPUT IS USED FOR ENTRY INTO RPOLY(50,4,50)
08400      C ASSUME A BOX OF LENGTH 100. UNITS ON A SIDE, THE CIRCLE WILL BE
08500      C CENTERED INSIDE THE BOX
08600      C USER MAY REWRITE THIS SECTION OF CODE TO MEET ANY SPECIAL
08700      C CASE DESIRED AS LONG AS RPOLY(50,4,50) IS FILLED CORRECTLY
08800          WRITE(5,60)
08900      60  FORMAT(1X,' ENTER DIAMETER OF CIRCLE < 140.0 ')
09000          READ(5,*) DIA
09100          WRITE(5,70) DIA
09200          WRITE(IPRT,70) DIA
09300      70  FORMAT(1X,' DIAMETER OF THE CIRCLE IS ',F7.2)
09400          RAD=DIA/2.
09500      C FIND THE INCREMENT BETWEEN THE DIAGONALS, THE INCREMENT IS
09600      C MEASURED ALONG THE VERTICAL AXIS
09700          CRE=200./((LINES+1)
09800          B=100.*(-1.)+CRE
09900      C SOLVE FOR Y GIVEN A VALUE FOR B
10000      C FIND DIAGONALS STARTING IN THE LOWER RIGHT-HAND CORNER
10100          WRITE(5,80)
10200      80  FORMAT(1X,' ENTER VALUE OUTSIDE CIRCLE, VALUE INSIDE ')
10300          READ(5,*) VO,VI
10400          WRITE(5,90) VO,VI
10500          WRITE(IPRT,90) VO,VI
10600      90  FORMAT(1X,' VALUE OUTSIDE CIRCLE IS ',F7.2,' VALUE INSIDE ',F7.2)
10700          ZMAX=VI
10800          DO 130 M=1,LINES
10900              I=0
11000              IF(B.GT.0.) GO TO 100
11100              XB=B*(-1.)
11200              YB=0.
11300              XE=100.
11400              YE=B+100.
11500              GO TO 110
11600      120  XB=0.
11700              YB=B
11800              XE=B*(-1.)+100.
11900              YE=100.
12000      110  CONTINUE
12100          R2=(RAD**2)*2.
12200          B2=B**2

```

```

12300      IF (ABS(R2).LT.82) GO TO 120
12400      YY = SQRT ((R2 - B2) / 4.) + B/2.
12500      XX=YY-B
12600      XIN=YY*(-1.)+50.
12700      YIN=50.-XX
12800      XOUT=50.+XX
12900      YOUT=50.+YY
13000      I=I+1
13100      RPOLY(I,1,M)=XIN
13200      RPOLY(I,2,M)=YIN
13300      RPOLY(I,3,M)=VI
13400      I=I+1
13500      RPOLY(I,1,M)=XOUT
13600      RPOLY(I,2,M)=YOUT
13700      RPOLY(I,3,M)=VI
13800 120    I=I+1
13900      RPOLY(I,1,M)=XB
14000      RPOLY(I,2,M)=YB
14100      RPOLY(I,3,M)=VB
14200      I=I+1
14300      RPOLY(I,1,M)=XE
14400      RPOLY(I,2,M)=YE
14500      RPOLY(I,3,M)=VE
14600      I=I+1
14700      RPOLY(I,1,M)=9999.
14800      RPOLY(I,2,M)=0.
14900      RPOLY(I,3,M)=0.
15000      B=B+CRE
15100 130    CONTINUE
15200      ITYPE=3
15300      GO TO 600
15400  C*****
15500  C
15600  C SECTION B. COORDINATE DATA INPUT
15700  C
15800  C*****
15900  C COME HERE FOR POLYGONAL DATA
16000 140    WRITE(5,150)
16100 150    FORMAT(1X,' WHAT TYPE OF DATA INPUT?',/,
16200           1'      1 FOR X,Y DATA(I),ICOR(I)      0 FOR X, ICOR(I)')
16300           READ(5,*) INPUT
16400           IF (INPUT.EQ.0) GO TO 160
16500  C IRDX - INTEGER READ X
16600  C IRDY - INTEGER READ Y
16700  C IRDD - INTEGER READ DATA
16800  C IRDNPP - INTEGER READ NUMBER OF POINTS PER POLYGON
16900           IROX=21
17000           IRDY=22
17100           IRDD=23
17200           IRDNPP=24
17300           OPEN(UNIT=IROX,FILE='XCOORD.DAT')
17400           OPEN(UNIT=IRDY,FILE='YCOORD.DAT')
17500           OPEN(UNIT=IRDD,FILE='ZVAL1.DAT')
17600           OPEN(UNIT=IRDNPP,FILE='ICOR.DAT')
17700           GO TO 170
17800 160    IRO=21
17900           IRDD=23
18000           OPEN(UNIT=IRO,FILE='XY.DAT')
18100           OPEN(UNIT=IRDD,FILE='ZVAL2.DAT')
18200 170    WRITE(5,180)
18300 180    FORMAT(1X,'HOW MANY POLYGONS?')

```

```

18400      READ(5,*) IPOLY
18500      WRITE(5,190) IPOLY
18600      WRITE(IPRT,190) IPOLY
18700 190   FORMAT(1X,' THERE ARE ',I3,' POLYGONS')
18800 200   WRITE(5,210)
18900 210   FORMAT(1X,'WHAT TYPE OF COLUMNS OR ROWS? ',/,
19000      1' ENTER 1-LINES ALONG COLUMN OF 90 DEG.',/,
19100      2'      2-LINES ALONG ROWS OF 0 DEG.',/,
19200      3'      3-LINES AT DIAGONALS OF 45 DEG.',/,
19300      4'      4-LINES AT DIAGONALS OF 135 DEG.')
```

```

19400      READ(5,*) ITYPE
19500      WRITE(5,220) ITYPE
19600      WRITE(IPRT,220) ITYPE
19700 220   FORMAT(1X,' TYPE OF COLUMNS OR ROWS IS: ',I3)
19800 230   IF(IDATA.EQ.0) GO TO 600
19900      C
20000      C READ IN DATA
20100      C PUT X COORDINATES OF POLYGONS IN FILE XCOOR.DAT
20200      C PUT Y COORDINATES OF POLYGONS IN FILE YCOOR.DAT
20300      C PUT DATA IN FILE ZVAL.DAT
20400      C PUT NUMBER OF COORDINATES OF THAT POLYGON IN FILE ICOOR.DAT
20500      C
20600      C CASE 1 - DATA INPUT IS X,Y,DATA(I),ICOOR(I)
20700      IF(INPUT.EQ.0) GO TO 250
20800      DO 240 I=1,IPOLY
20900      READ(IRDNPP,*) ICOOR(I)
21000      K=ICOOR(I)
21100      READ(IRDX,*) (X(I,J),J=1,K)
21200      READ(IRDY,*) (Y(I,J),J=1,K)
21300      READ(IROD,*) DATA(I)
21400 240   CONTINUE
21500      GO TO 310
21600      C CASE 2 - DATA INPUT IS X,ICOOR(I)
21700 250   READ(IPD,260) IA,XX,YY
21800 260   FORMAT(9X,I1,2F10.3)
21900      DO 310 I=1,IPOLY
22000      K=0
22100      X(I,1)=XX
22200      Y(I,1)=YY
22300      DO 280 J=2,MINTER
22400      READ(IRD,270) IA,XX,YY
22500 270   FORMAT(9X,I1,2F10.3)
22600      K=K+1
22700      IF(IA.EQ.1) GO TO 290
22800      X(I,J)=XX
22900      Y(I,J)=YY
23000 280   CONTINUE
23100 290   JM1=J-1
23200      ICOOR(I)=K
23300      IF(X(I,1).EQ.X(I,JM1).AND.Y(I,1).EQ.Y(I,JM1)) GO TO 310
23400      WRITE(IPRT,300) I,JM1
23500 300   FORMAT(1X,' ERROR IN INPUT, FIRST ENTRY DOES NOT MATCH LAST',/,
23600      1' I,J ',2I4)
23700 310   CONTINUE
23800      C NORMALIZE X AND Y DATA FROM SMALLEST TO 100
23900      C FIND LARGEST X AND Y VALUES
24000      YLARGE=Y(1,1)
24100      XLARGE=X(1,1)
24200      DO 330 I=1,IPOLY
24300      K=ICOOR(I)
24400      DO 320 J=1,K
```

```

24500         IF(Y(I,J).GT.YLARGE) YLARGE=Y(I,J)
24600         IF(X(I,J).GT.XLARGE) XLARGE=X(I,J)
24700     320     CONTINUE
24800     330     CONTINUE
24900     C ADJUST Y VALUES FROM SMALLEST TO 100
25000         DO 350 I=1,IPOLY
25100         K=ICORR(I)
25200         DO 340 J=1,K
25300         Y(I,J)=((Y(I,J))/YLARGE)*100.
25400         X(I,J)=((X(I,J))/XLARGE)*100.
25500     340     CONTINUE
25600     350     CONTINUE
25700     C*****
25800     C
25900     C SECTION C. COMPUTE THE END POINTS OF THE DIAGONALS
26000     C
26100     C*****
26200     C
26300     C FIND MAX AND MIN OF X-Y VALUES FOR RECT. TO FIT AROUND FIGURE
26400     C
26500         XMAX=X(1,1)
26600         XMIN=X(1,1)
26700         YMAX=Y(1,1)
26800         YMIN=Y(1,1)
26900         DO 360 I=1,IPOLY
27000         N=ICORR(I)
27100         DO 360 J=1,N
27200         IF(X(I,J).GT.XMAX) XMAX=X(I,J)
27300         IF(X(I,J).LT.XMIN) XMIN=X(I,J)
27400         IF(Y(I,J).GT.YMAX) YMAX=Y(I,J)
27500         IF(Y(I,J).LT.YMIN) YMIN=Y(I,J)
27600     360     CONTINUE
27700         XMAX=XMAX+5.
27800         XMIN=XMIN-5.
27900         YMAX=YMAX+5.
28000         YMIN=YMIN-5.
28100     370     CONTINUE
28200     C
28300     C DETERMINE BRANCH FOR TYPE OF DIAGONALS
28400     C COMPUTES THE DIAGONALS BEFORE INTERSECTIONS ARE SOUGHT.
28500     C BASED ON NUMBER OF DIAGONALS AND TYPE OF DIAGONALS, CAN BE
28600     C MODIFIED TO TAKE INTO ACCOUNT ANY DEGREE DIAGONAL BY
28700     C FINDING THE TAN OF THE ANGLE OF THE DIAGONAL.
28800     C DIAGLR(50,4) IS FILLED LEFT TO RIGHT.
28900     C
29000         GO TO (380,400,420,490),ITYPE
29100     C
29200     C LINES ALONG COLUMNS TYPE=1
29300     C
29400     380     RLINES=FLOAT(LINES)
29500         XDIV=(XMAX-XMIN)/(RLINES+1.)
29600         DIAGLR(1,1)=XMAX-XDIV
29700         DIAGLR(1,2)=YMIN
29800         DIAGLR(1,3)=DIAGLR(1,1)
29900         DIAGLR(1,4)=YMAX
30000         DO 390 I=2,LINES
30100         J=[-1
30200         DIAGLR(I,1)=DIAGLR(J,1)-XDIV
30300         DIAGLR(I,2)=YMIN
30400         DIAGLR(I,3)=DIAGLR(I,1)
30500         DIAGLR(I,4)=YMAX

```

```

30600 390    CONTINUE
30700      GO TO 560
30800  C
30900  C LINE ALONG ROWS TYPE=2
31000  C
31100  400    RLines=FLOAT(LINES)
31200      YDIV=(YMAX-YMIN)/(RLines+1.)
31300      DIAGLR(1,1)=XMIN
31400      DIAGLR(1,2)=YMIN+YDIV
31500      DIAGLR(1,3)=XMAX
31600      DIAGLR(1,4)=DIAGLR(1,2)
31700      DO 410 I=2,LINES
31800        J=I-1
31900        DIAGLR(I,1)=XMIN
32000        DIAGLR(I,2)=DIAGLR(J,2)+YDIV
32100        DIAGLR(I,3)=XMAX
32200        DIAGLR(I,4)=DIAGLR(I,2)
32300  410    CONTINUE
32400      GO TO 560
32500  C
32600  C LINES AT DIAGONALS OF 45 DEG. TYPE=3
32700  C
32800  420    RLines=FLOAT(LINES)
32900      TAN45=SIND(45.)/COSD(45.)
33000      COT45=COSD(45.)/SIND(45.)
33100      XDIV=COT45*(YMAX-YMIN)
33200      XDIV=((XMAX-XMIN)+XDIV)/(RLines+1.)
33300      JMPX=.FALSE.
33400      JMPY=.FALSE.
33500      DIAGLR(1,1)=XMAX-XDIV
33600      DIAGLR(1,2)=YMIN
33700      YDIV=TAN45*XDIV
33800      DIAGLR(1,3)=XMAX
33900      DIAGLR(1,4)=YMIN+YDIV
34000      DO 480 I=2,LINES
34100        J=I-1
34200        IF(JMPX)      GO TO 460
34300        IF(DIAGLR(J,1)-XDIV.LT.XMIN)      GO TO 450
34400        DIAGLR(I,1)=DIAGLR(J,1)-XDIV
34500        DIAGLR(I,2)=YMIN
34600  430    IF(JMPY)      GO TO 470
34700        IF(DIAGLR(J,4)+YDIV.GT.YMAX)      GO TO 440
34800        DIAGLR(I,3)=XMAX
34900        DIAGLR(I,4)=DIAGLR(J,4)+YDIV
35000        GO TO 480
35100  440    SY=DIAGLR(J,4)+YDIV-YMAX
35200        SX=COT45*SY
35300        JMPY=.TRUE.
35400        DIAGLR(I,3)=XMAX-SX
35500        DIAGLR(I,4)=YMAX
35600        GO TO 480
35700  450    SX=XDIV-(DIAGLR(J,1)-XMIN)
35800        SY=TAN45*SX
35900        JMPX=.TRUE.
36000        DIAGLR(I,1)=XMIN
36100        DIAGLR(I,2)=YMIN+SY
36200        GO TO 430
36300  460    DIAGLR(I,1)=XMIN
36400        DIAGLR(I,2)=DIAGLR(J,2)+YDIV
36500        GO TO 430
36600  470    DIAGLR(I,3)=DIAGLR(J,3)-XDIV

```

```

36700      DIAGLR(I,4)=YMAX
36800  480      CONTINUE
36900      GO TO 560
37000      C
37100  C LINES AT DIAGONALS OF 135 DEG. TYPE=4
37200      C
37300  490      RLines=FLCAT(LINES)
37400      TAN45=SIND(45.)/COSD(45.)
37500      COT45=COSD(45.)/SIND(45.)
37600      XDIS=COT45*(YMAX-YMIN)
37700      XDIV=(XMAX-XMIN+XDIS)/(RLINES+1.)
37800      JMPX=.FALSE.
37900      JMPY=.FALSE.
38000      YDIV=TAN45*XDIV
38100      DIAGLR(I,1)=XMIN
38200      DIAGLR(I,2)=YMIN+YDIV
38300      DIAGLR(I,3)=XMIN+XDIV
38400      DIAGLR(I,4)=YMIN
38500      DO 550 I=2,LINES
38600      J=I-1
38700      IF(JMPY)      GO TO 530
38800      IF(DIAGLR(J,2)+YDIV.GT.YMAX)  GO TO 520
38900      DIAGLR(I,1)=XMIN
39000      DIAGLR(I,2)=DIAGLR(J,2)+YDIV
39100  500      IF(JMPX)  GO TO 540
39200      IF(DIAGLR(J,3)+XDIV.GT.XMAX)  GO TO 510
39300      DIAGLR(I,3)=DIAGLR(J,3)+XDIV
39400      DIAGLR(I,4)=YMIN
39500      GO TO 550
39600  510      SX=DIAGLR(J,3)+XDIV-XMAX
39700      SY=TAN45*SX
39800      JMPX=.TRUE.
39900      DIAGLR(I,3)=XMAX
40000      DIAGLR(I,4)=YMIN+SY
40100      GO TO 550
40200      SY=DIAGLR(J,2)+YDIV-YMAX
40300  520      SX=COT45*SY
40400      JMPY=.TRUE.
40500      DIAGLR(I,1)=XMIN+SX
40600      DIAGLR(I,2)=YMAX
40700      GO TO 500
40800  530      DIAGLR(I,1)=DIAGLR(J,1)+XDIV
40900      DIAGLR(I,2)=YMAX
41000      GO TO 500
41100      DIAGLR(I,3)=XMAX
41200  540      DIAGLR(I,4)=DIAGLR(J,4)+YDIV
41300  550      CONTINUE
41400  C*****
41500      C
41600  C SECTION D.  POLYGONS AND DIAGONALS INTERSECTIONS
41700      C
41800  C*****
41900      C
42000  C LOAD INTERSECTIONS OF ALL DIAGONALS AND LINE SEGMENTS
42100  C AND ADD IN VALUE OF POLYGON
42200  C ADD IN FIRST AND LAST POINTS OF LINE SEGMENT
42300      C
42400  560      DO 600 L=1,LINES
42500      I=0
42600      DO 590 LP=1,IPOLY
42700      N=ICODR(LP)-1

```

```

42800      DO 580 M=1,N
42900      MP1=M+1
43000      CALL INTER(DIAGLR(L,1),DIAGLR(L,2),DIAGLR(L,3),
43100      1DIAGLR(L,4),X(LP,M),Y(LP,M),X(LP,MP1),Y(LP,MP1),
43200      2XX,YY,LINT,LPAR)
43300      ZLP=FLOAT(LP)
43400      IF(LPAR.EQ.1) GO TO 570
43500      IF(LINT.EQ.0) GO TO 580
43600      I=I+1
43700      RPOLY(I,1,L)=XX
43800      RPOLY(I,2,L)=YY
43900      RPOLY(I,4,L)=ZLP
44000      IF(LPAR.EQ.0) GO TO 580
44100 570 I=I+1
44200      RPOLY(I,1,L)=X(LP,M)
44300      RPOLY(I,2,L)=Y(LP,M)
44400      RPOLY(I,4,L)=ZLP
44500      I=I+1
44600      RPOLY(I,1,L)=X(LP,MP1)
44700      RPOLY(I,2,L)=Y(LP,MP1)
44800      RPOLY(I,4,L)=ZLP
44900 580 CONTINUE
45000 590 CONTINUE
45100      I=I+1
45200      RPOLY(I,1,L)=DIAGLR(L,1)
45300      RPOLY(I,2,L)=DIAGLR(L,2)
45400      RPOLY(I,4,L)=0.
45500      I=I+1
45600      RPOLY(I,1,L)=DIAGLR(L,3)
45700      RPOLY(I,2,L)=DIAGLR(L,4)
45800      RPOLY(I,4,L)=0.
45900      I=I+1
46000      RPOLY(I,1,L)=9999.
46100 600 CONTINUE
46200 C*****
46300 C
46400 C SECTION E. SORTING AND DELETION OF DUPLICATE ENTRIES ON DIAGONAL
46500 C
46600 C*****
46700 C
46800 C SORT ALL DUPLICATE ENTRIES IN RPOLY(I,J,L) ON X-VALUE
46900 C ADJUST FOR SORT ON Y VALUE IF ITYPE = 1 - VERTICAL LINES
47000      IF(ITYPE.EQ.1) ISORT=2
47100      IF(ITYPE.GE.2) ISORT=1
47200      DO 810 L=1,LINES
47300      NMI=MINTER-1
47400 C # OF INTERSECTIONS - 1 = 49
47500      DO 680 K=1,NMI
47600      KP1=K+1
47700 C CHECK FOR FLAG - END OF DATA
47800      IF(RPOLY(K,1,L).EQ.9999.) GO TO 697
47900      IF(RPOLY(KP1,1,L).EQ.9999.) GO TO 690
48000 610 DO 670 I=KP1,MINTER
48100 C # OF INTERSECTION = 50
48200      IF(RPOLY(I,1,L).EQ.9999.) GO TO 680
48300 C CHECK FOR DUPLICATE ENTRIES
48400      IF(ABS(RPOLY(K,1,L)-RPOLY(I,1,L)).LE..0001.AND.ABS(RPOLY(K,2,L)-
48500      1RPOLY(I,2,L)).LE..0001.AND.ABS(RPOLY(K,4,L)-RPOLY(I,4,L))
48600      2.LE..0001)GO TO 640
48700 C CHECK IF LESS THAN
48800      IF(RPOLY(K,ISORT,L).LE.RPOLY(I,ISORT,L))GO TO 670

```

```

48900 620 DO 630 J=1,4
49000 TEMP=RPOLY(K,J,L)
49100 C GREATER THAN SO EXCHANGE
49200 RPOLY(K,J,L)=RPOLY(I,J,L)
49300 RPOLY(I,J,L)=TEMP
49400 630 CONTINUE
49500 GO TO 610
49600 C DELETE ONE OF DUPLICATE ENTRIES
49700 640 DO 660 II=I,MINTER
49800 IIP1=II+1
49900 DO 650 J=1,4
50000 RPOLY(II,J,L)=RPOLY(IIP1,J,L)
50100 650 CONTINUE
50200 IF(RPOLY(II,1,L).EQ.9999.) GO TO 610
50300 660 CONTINUE
50400 670 CONTINUE
50500 680 CONTINUE
50600 C CHECK FOR ONLY ONE ENTRY BY POLYGON #
50700 690 DO 740 KK=1,NM1
50800 IF(RPOLY(KK,1,L).EQ.9999.) GO TO 750
50900 DO 700 M=1,MINTER
51000 IF(M.EQ.KK) GO TO 700
51100 IF(RPOLY(M,1,L).EQ.9999.) GO TO 710
51200 IF(RPOLY(KK,4,L).EQ.RPOLY(M,4,L)) GO TO 740
51300 700 CONTINUE
51400 C DELETE ONE ENTRY
51500 710 DO 730 MM=KK,MINTER
51600 MMP1=MM+1
51700 DO 720 J=1,4
51800 RPOLY(MM,J,L)=RPOLY(MMP1,J,L)
51900 720 CONTINUE
52000 C CHECK FOR FLAG - END OF DATA
52100 IF(RPOLY(MM,J,L).EQ.9999.) GO TO 690
52200 730 CONTINUE
52300 740 CONTINUE
52400 C SORT INTERSECTION LINE ON HEIGHT VALUE
52500 750 ITEST=0
52600 DO 770 N=2,NM1
52700 NP1=N+1
52800 NMP1=N-1
52900 C CHECK FOR FLAG - END OF DATA
53000 IF(RPOLY(NP1,1,L).EQ.9999.) GO TO 780
53100 C CHECK TO SEE IF X AND Y PAIRS ARE NOT EQUAL
53200 IF(RPOLY(N,1,L).NE.RPOLY(NP1,1,L).OR.RPOLY(N,2,L).NE.RPOLY(NP1,2
53300 1L)) GO TO 770
53400 C CHECK TO SEE IF POLYGON # ARE EQUAL
53500 IF(RPOLY(N,4,L).EQ.RPOLY(NMP1,4,L)) GO TO 770
53600 ITEST=1
53700 C SWITCH ENTRIES
53800 DO 760 J=1,4
53900 TEMP=RPOLY(N,J,L)
54000 RPOLY(N,J,L)=RPOLY(NP1,J,L)
54100 RPOLY(NP1,J,L)=TEMP
54200 760 CONTINUE
54300 770 CONTINUE
54400 IF(ITEST.EQ.1) GO TO 750
54500 C TO GET RID OF SAME X-Y ENTRY BUT OUT OF ORDER Y+POLY VALUE
54600 780 DO 800 N=2,NM1,2
54700 NP1=N+1
54800 NP2=N+2
54900 C CHECK FOR FLAG - END OF DATA

```

```

55000      IF(RPOLY(NP1,1,L).EQ.9999.) GO TO 810
55100      IF(RPOLY(N,4,L).EQ.RPOLY(NP1,4,L)) GO TO 800
55200      IF(RPOLY(N,4,L).NE.RPOLY(NP2,4,L)) GO TO 800
55300      C SWITCH NP1 AND NP2
55400      DO 790 J=1,4
55500      TEMP=RPOLY(NP2,J,L)
55600      RPOLY(NP2,J,L)=RPOLY(NP1,J,L)
55700      RPOLY(NP1,J,L)=TEMP
55800      790 CONTINUE
55900      GO TO 780
56000      800 CONTINUE
56100      810 CONTINUE
56200      C
56300      C FIND THE NUMBER OF INTERSECTION IN EACH LINE
56400      C
56500      DO 930 L=1,LINES
56600      820 N=0
56700      DO 830 I=1,MINTER
56800      C # OF INTERSECTIONS = 50
56900      C CHECK FOR FLAG - END OF DATA
57000      IF(RPOLY(I,1,L).EQ.9999.) GO TO 840
57100      N=N+1
57200      830 CONTINUE
57300      840 LINE(L)=N
57400      C IF LINE(N) IS ODD - MEANS 1 INTERSECTION SOMEWHERE
57500      850 DO 860 J=2,MINTER,2
57600      IODD=N-J
57700      IF(IODD.EQ.0) GO TO 930
57800      IF(IODD.LT.0) GO TO 870
57900      860 CONTINUE
58000      C CHECK FOR ONLY 1 INTERSECTION BY LOOKING AT Z VALUE
58100      870 DO 920 K=1,N
58200      HEIGHT=RPOLY(K,4,L)
58300      DO 880 I=1,N
58400      IF(I.EQ.K) GO TO 880
58500      IF(HEIGHT.EQ.RPOLY(I,4,L)) GO TO 920
58600      880 CONTINUE
58700      C DELETE ROW OF Z AND MOVE EVERYTHING UP 1
58800      890 DO 910 II=K,MINTER
58900      IIP1=II+1
59000      DO 900 J=1,4
59100      RPOLY(II,J,L)=RPOLY(IIP1,J,L)
59200      900 CONTINUE
59300      IF(RPOLY(II,1,L).EQ.9999.) GO TO 820
59400      910 CONTINUE
59500      920 CONTINUE
59600      930 CONTINUE
59700      C FIND PARALLEL LINES OF DIAGONAL AND POLYGON SEGMENTS
59800      C WITH SEPERATED Z VALUES
59900      940 DO 1040 L=1,LINES
60000      IF(LINE(L).LE.4) GO TO 1040
60100      950 LL=LINE(L)-1
60200      IOVER=0
60300      DO 1000 K=2,LL,2
60400      KP1=K+1
60500      IF(RPOLY(K,4,L).EQ.RPOLY(KP1,4,L)) GO TO 1000
60600      ISTART=KP1
60700      START=RPOLY(K,4,L)
60800      NEXT=KP1+1
60900      DO 960 KK=NEXT,LL
61000      IF(RPOLY(KK,4,L).EQ.START) GO TO 970

```

```

61100 960 CONTINUE
61200 970 INEXT=KK-1
61300 IOVER=1
61400 DO 990 II=ISTART,MINTER
61500 INEXT=INEXT+1
61600 DO 980 J=1,4
61700 RPOLY(II,J,L)=RPOLY(INEXT,J,L)
61800 980 CONTINUE
61900 C CHECK FOR FLAG - END OF DATA
62000 IF(RPOLY(II,1,L).EQ.9999.) GO TO 1010
62100 990 CONTINUE
62200 1000 CONTINUE
62300 IF(IOVER.EQ.0) GO TO 1040
62400 C COUNT NUMBER OF INTERSECTION IN THIS LINE
62500 1010 N=0
62600 DO 1020 I=1,MINTER
62700 C CHECK FOR FLAG - END OF DATA
62800 IF(RPOLY(I,1,L).EQ.9999.) GO TO 1030
62900 N=N+1
63000 1020 CONTINUE
63100 1030 LINE(L)=N
63200 GO TO 950
63300 1040 CONTINUE
63400 C PUT DATA VALUES IN BY POLYGON # FOR CASE 1
63500 IF(IDATA.EQ.0.OR.INPUT.EQ.0) GO TO 1080
63600 DO 1080 L=1,LINES
63700 N=LINE(L)
63800 DO 1070 I=1,N
63900 LP=IFIX(RPOLY(I,4,L))
64000 IF(LP.EQ.0) GO TO 1060
64100 RPOLY(I,3,L)=DATA(LP)
64200 GO TO 1070
64300 1060 RPOLY(I,3,L)=0.
64400 1070 CONTINUE
64500 1080 CONTINUE
64600 C ENTER PARAMETER INPUTS FOR TRANSFORMATION MATRIX
64700 P=0.
64800 Q=0.
64900 SCALE=1.
65000 A=1.
65100 D=1.
65200 B=1.
65300 C=1.
65400 1090 WRITE(5,1100)
65500 1100 FORMAT(1X,' ENTER 1 TO SET LOCAL SCALING PARAMETER',/,
65600 1' 2 TO SET SHEARING',/,
65700 2' 3 TO SET VANISHING POINT',/,
65800 3' 4 TO CONTINUE')
65900 READ(5,*) ICONT
66000 GO TO (1110,114),1170,1200), ICONT
66100 1110 WRITE(5,1120)
66200 1120 FORMAT(1X,' ENTER LOCAL SCALING PARAMETERS OF INITIAL',/,
66300 1' OBJECT IN A X- AND Y-DIRECTION, RESPECTIVELY.')
66400 READ(5,*) A,D
66500 WRITE(5,1130) A,D
66600 WRITE(IPRT,1130) A,D
66700 1130 FORMAT(1X,' LOCAL SCALING IN A X- AND Y-DIRECTION',/,
66800 1' RESPECTIVELY:',2F7.2)
66900 GO TO 1090
67000 1140 WRITE(5,1150)
67100 1150 FORMAT(1X,' ENTER SHEARING PARAMETERS OF THE INITIAL OBJECT',/,

```

```

67200      1' IN A Y- AND X-DIRECTION, RESPECTIVELY.')
67300      READ(5,*) B,C
67400      CALL ERASE
67500      WRITE(5,1160) B,C
67600      WRITE(IPRT,1160) B,C
67700  1160    FORMAT(1X,' SHEARING PARAMETERS IN A Y- AND X-DIRECTION','/,
67800      1' RESPECTIVELY:',2F7.2)
67900      GO TO 1090
68000  1170    WRITE(5,1180)
68100  1180    FORMAT(1X,' ENTER VALUE OF VANISHING POINT FOR X- AND Y-AXIS',/,
68200      1' RESPECTIVELY, FOR TWO POINT PERSPECTIVE.')
68300      READ(5,*) P,Q
68400      WRITE(5,1190) P,Q
68500      WRITE(IPRT,1190) P,Q
68600  1190    FORMAT(1X,' VANISHING POINTS FOR X- AND Y-AXIS','/,
68700      1' RESPECTIVELY:',2F7.2)
68800      GO TO 1090
68900  C READ IN NEW DATA FROM DATA FORMAT
69000  1200    IF((DATA.EQ.0.OR.INPUT.EQ.1) GO TO 1230
69100      WRITE(5,1210)
69200  1210    FORMAT(1X,' ENTER DATA FORMAT')
69300      READ(5,1220) FORM
69400  1220    FORMAT(16A5)
69500      DO 1230 I=1,IPOLY
69600      READ(IPDD,FORM) DATA(I)
69700  1230    CONTINUE
69800      WRITE(5,1240)
69900  1240    FORMAT(1X,' ENTER OVERALL SCALING OF FINAL BLOCK DIAGRAM <= 1.',
70000      1' AND > 0.')
70100      READ(5,*) SCALE
70200      WRITE(5,1250) SCALE
70300      WRITE(IPRT,1250) SCALE
70400  1250    FORMAT(1X,' OVERALL SCALING IS:',F7.2)
70500      WRITE(5,1260)
70600  1260    FORMAT(1X,' WHAT VIEWING ANGLE DO YOU WANT, IN DEGREES? ')
70700      READ(5,*) ALP
70800      WRITE(5,1270) ALP
70900      WRITE(IPRT,1270) ALP
71000  1270    FORMAT(1X,' VIEWING ANGLE IS ',F7.2)
71100      SINALP=SIND(ALP)
71200      COSALP=COSD(ALP)
71300      WRITE(5,1280) XMIN,XMAX,YMIN,YMAX
71400  1280    FORMAT(1X,' THE LIMITS OF THE X-AXIS ARE:',2F7.2,/,
71500      1' THE LIMITS OF THE Y-AXIS ARE:',2F7.2)
71600      WRITE(5,1290)
71700  1290    FORMAT(1X,' DO YOU WANT ROTATION ABOUT THE Y-AXIS?',/,
71800      1' TYPE 1 IF YES, 0 IF NO')
71900      READ(5,*) IROT
72000      IF(IPOT.EQ.0) GO TO 1320
72100      WRITE(5,1300)
72200  1300    FORMAT(1X,' WHAT ROTATION ABOUT AN X-Y POINT?',/
72300      1' WHAT X-Y POINT?')
72400      READ(5,*) XYROT,XPOINT,YPOINT
72500      WRITE(5,1310) XYROT,XPOINT,YPOINT
72600      WRITE(IPRT,1310) XYROT,XPOINT,YPOINT
72700  1310    FORMAT(1X,' ROTATE ',F7.2,' DEGREES ABOUT POINT ',2(F7.2,1X))
72800      GO TO 1330
72900  1320    XYROT=0.
73000      XPOINT=0.
73100      YPOINT=0.
73200  1330    CALL POTXY(XYROT,XPOINT,YPOINT,A,B,C,D,P,Q)

```

```

73300      CALL ERASE
73400      1340      IF(IROT.EQ.0) GO TO 1450
73500      C SEE IF LINE ORDERING AND ENTRIES PER DIAGONAL ARE
73600      C CORRECT DUE TO ROTATION
73700          ICHANG=0
73800          GO TO (1350,1360,1370,1380),ITYPE
73900      C ITYPE = 1 90 DEG. COLUMNS
74000      1350      IF(XYROT.GT.0..AND.XYROT.LT.180.) GO TO 1390
74100          IF(XYROT.LT.-180..AND.XYROT.GT.-360.) GO TO 1390
74200          GO TO 1400
74300      C ITYPE = 2 0 DEG. ROWS
74400      1360      IF(XYROT.GT.90..AND.XYROT.LT.270) GO TO 1390 -
74500          IF(XYROT.LT.-90..AND.XYROT.GT.-270) GO TO 1390
74600          GO TO 1400
74700      C ITYPE = 3 DIAGONALS AT 45 DEG.
74800      1370      IF(XYROT.GT.45..AND.XYROT.LT.225.) GO TO 1390
74900          IF(XYROT.LT.-135..AND.XYROT.GT.-315.) GO TO 1390
75000          GO TO 1400
75100      C ITYPE = 4 DIAGONALS AT 135 DEG.
75200      1380      IF(XYROT.GT.135..AND.XYROT.LT.315.) GO TO 1390
75300          IF(XYROT.LT.-45..AND.XYROT.GT.-225.) GO TO 1390
75400          GO TO 1400
75500      1390      ICHANG=1
75600          CALL CHANGE
75700      1400      IP135=0
75800      C COMPUTE IF PLOTTING IS TO DONE AT 135 DEG.
75900          GO TO (1410,1420,1430,1440),ITYPE
76000      C ITYPE = 1 90 DEG. COLUMNS
76100      1410      IF(XYROT.GT.0..AND.XYROT.LT.90.) IP135=1
76200          IF(XYROT.GT.-360..AND.XYROT.LT.-270.) IP135=1
76300          IF(XYROT.GT.180..AND.XYROT.LT.270.) IP135=1
76400          IF(XYROT.GT.-180..AND.XYROT.LT.-90.) IP135=1
76500          GO TO 1450
76600      C ITYPE = 2 0 DEG. ROWS
76700      1420      IF(XYROT.GT.90..AND.XYROT.LT.180.) IP135=1
76800          IF(XYROT.GT.-270..AND.XYROT.LT.-180.) IP135=1
76900          IF(XYROT.GT.270..AND.XYROT.LT.360.) IP135=1
77000          IF(XYROT.GT.-90..AND.XYROT.LT.0.) IP135=1
77100          GO TO 1450
77200      C ITYPE = 3 DIAGONALS AT 45 DEG.
77300      1430      IF(XYROT.GT.45..AND.XYROT.LT.135.) IP135=1
77400          IF(XYROT.GT.-315..AND.XYROT.LT.-225.) IP135=1
77500          IF(XYROT.GT.225..AND.XYROT.LT.315.) IP135=1
77600          IF(XYROT.GT.-135..AND.XYROT.LT.-45.) IP135=1
77700          GO TO 1450
77800      C ITYPE = 4 DIAGONALS AT 135 DEG.
77900      1440      IF(XYROT.GT.0..AND.XYROT.LT.45.) IP135=1
78000          IF(XYROT.GT.-360..AND.XYROT.LT.-315.) IP135=1
78100          IF(XYROT.GT.135..AND.XYROT.LT.225.) IP135=1
78200          IF(XYROT.GT.-225..AND.XYROT.LT.-135.) IP135=1
78300          IF(XYROT.GT.315..AND.XYROT.LT.360.) IP135=1
78400          IF(XYROT.GT.-45..AND.XYROT.LT.0.) IP135=1
78500      1450      WRITE(5,1460)
78600      1460      FORMAT(1X,' DO YOU WANT ROTATION ABOUT THE Z-AXIS?',/,
78700          1'      TYPE 1 IF YES, 0 IF NO')
78800          READ(5,*) IROTZ
78900          IF(IROTZ.EQ.0) GO TO 1520
79000          WRITE(5,1470)
79100      1470      FORMAT(1X,' ENTER ANGLE OF ROTATION ABOUT Z-AXIS.')
```

```

79400      WRITE(IPRT,1480) ZROT
79500      FORMAT(1X,' ROTATION ABOUT Z-AXIS IS:',F7.2)
79600      C FIND XMAXZ AND XMINZ
79700          XMAXZ=RPOLY(1,1,1)
79800          XMINZ=XMAXZ
79900          DO 1500 L=1,LINES
80000              IVAL=LINE(L)
80100              DO 1490 I=1,IVAL
80200                  IF(RPOLY(I,1,L).GT.XMAXZ) XMAXZ=RPOLY(I,1,L)
80300      1490      CONTINUE
80400      1500      CONTINUE
80500          TANZ=SIND(-ZROT)/COSD(-ZROT)
80600          YTALLZ=TANZ*XMAXZ
80700          DO 1520 L=1,LINES
80800              IVAL=LINE(L)
80900              DO 1510 I=1,IVAL
81000                  YADDZ=(YTALLZ*RPOLY(I,1,L))/XMAXZ
81100                  RPOLY(I,2,L)=RPOLY(I,2,L)+YADDZ
81200      1510      CONTINUE
81300      1520      CONTINUE
81400          IF(IDATA.EQ.0) GO TO 1540
81500      C NORMALIZE DATA SO NOT TOO BIG FOR SCREEN BETWEEN 0 TO 100
81600          DATMAX=0.
81700          DO 1530 I=1,IPOLY
81800              IF(DATA(I).GT.DATMAX) DATMAX=DATA(I)
81900      1530      CONTINUE
82000          DO 1540 I=1,IPOLY
82100              DATA(I)=(DATA(I)/DATMAX)*100.
82200      1540      CONTINUE
82300      C SET DWINDO FOR FINAL PLOT
82400          XMAF=RPOLY(1,1,1)
82500          XMIF=XMAF
82600          YMAF=RPOLY(1,2,1)
82700          YMIF=YMAF
82800          VMAF=XMAF*SINALP
82900          VMAF=YMAF
83000          DO 1580 L=1,LINES
83100              N=LINE(L)
83200              DO 1570 I=1,N
83300                  IF(RPOLY(I,1,L).GT.XMAF) XMAF=RPOLY(I,1,L)
83400                  IF(RPOLY(I,1,L).LT.XMIF) XMIF=RPOLY(I,1,L)
83500                  IF(RPOLY(I,2,L).GT.YMAF) YMAF=RPOLY(I,2,L)
83600                  IF(RPOLY(I,2,L).LT.YMIF) YMIF=RPOLY(I,2,L)
83700                  IF(IDATA.EQ.0.OR.INPUT.EQ.0) GO TO 1560
83800                  LP=FIX(RPOLY(I,4,L))
83900                  IF(LP.EQ.0) GO TO 1550
84000                  RPOLY(I,3,L)=DATA(LP)
84100                  GO TO 1560
84200      1550      RPOLY(I,3,L)=0.
84300      1560      CONTINUE
84400          YTEMP=RPOLY(1,2,L)*SINALP+RPOLY(1,3,L)*COSALP
84500          IF(YTEMP.GT.VMAF) VMAF=YTEMP
84600      1570      CONTINUE
84700      1580      CONTINUE
84800      C SET DWINDO FOR FINAL PLOT
84900          XMA2=XMAF+2.
85000          XM12=XMIF-2.
85100          YMA2=YMAF+2.
85200          YM12=YMIF-2.
85300          CALL DWINDO(XM12,XMA2,YM12,YMA2)
85400      C ADJUST FOR OVERALL SCALING BY SETTING TWINDO

```

```

85500      MINX=512-IFIX(SCALE*512.)
85600      MAXX=512+IFIX(SCALE*512.)
85700      MINY=390-IFIX(SCALE*390.)
85800      MAXY=390+IFIX(SCALE*390.)
85900      CALL TWINDO(MINX,MAXX,MINY,MAXY)
86000      C*****
86100      C
86200      C SECTION G. HIDDEN-LINE ELIMINATIONS
86300      C
86400      C*****
86500      C DRAW FIRST LINE UP TO FIRST INTERSECTION
86600      1590      M=0
86700              DO 1600 L=1,LINES
86800              IF(LINE(L).GE.3)      GO TO 1610
86900      C SET END POINTS OF SEGMENTS
87000              X1=RPOLY(1,1,L)
87100              YN=RPOLY(1,2,L)
87200              ZN=RPOLY(1,3,L)
87300              Y1=YN*SINALP+ZN*COSALP
87400              X2=RPOLY(2,1,L)
87500              YN=RPOLY(2,2,L)
87600              ZN=RPOLY(2,3,L)
87700              Y2=YN*SINALP+ZN*COSALP
87800              CALL MOVEA(X1,Y1)
87900              CALL DRAWA(X2,Y2)
88000      1600      CONTINUE
88100      C SET UP VISIBLE TOP LINE IN PX(1,4)
88200      1610      IPX=1
88300              PX(1,1)=X1
88400              PX(1,2)=Y1
88500              PX(1,3)=X2
88600              PX(1,4)=Y2
88700      C DRAW SET OF LINES WHERE DIAGONALS INTERSECT POLYGONS
88800              DO 1920 L=L,LINES
88900              XX=0.
89000              YY=0.
89100              IFIN=0
89200              IPVIS=0
89300              IUPDAT=0
89400              ISAVE=1
89500              I135ST=0
89600              IND135=0
89700              ISDRAW=1
89800      C SET SET END POINTS OF # 1 PLOTTING SEQUENCE
89900              SX0=RPOLY(1,1,L)
90000              YN=RPOLY(1,2,L)
90100              ZN=RPOLY(1,3,L)
90200              SY0=YN*SINALP+ZN*COSALP
90300              SX1=RPOLY(2,1,L)
90400              YN=RPOLY(2,2,L)
90500              ZN=RPOLY(2,3,L)
90600              SY1=YN*SINALP+ZN*COSALP
90700      C CHECK FOR 135 DEG. DRAW TO SEE IF STARTING POINT VISIBLE
90800              DO 1640 I=1,IPX
90900              IF(PX(I,1).LT.PX(I,3))      GO TO 1620
91000              GO TO 1640
91100      1620      IF(SX0.LT.PX(I,1).OR.SX0.GT.PX(I,3))      GO TO 1640
91200              IF(SY0.GE.PX(I,2))      GO TO 1640
91300              IF(SY0.LE.PX(I,4))      GO TO 1630
91400              SYOM5=SY0-5.
91500              CALL INTER(SX0,SY0,SX0,SYOM5,PX(I,1),PX(I,2),PX(I,3),

```

```

91600      IPX(I,4),XX,YY,LINT,LPAR)
91700      IF(LINT.EQ.1) GO TO 1640
91800 1630  ISDRAW=0
91900      I135ST=1
92000      GO TO 1650
92100 1640  CONTINUE
92200 1650  CONTINUE
92300 C DRAW PLOTTING SEQUENCE # 1
92400 1660  CALL HIDDEN(IPR,IPX,SX0,SY0,SX1,SY1,XX,YY,LINT,LPAR)
92500      INTERA=1
92600      IF(LINT.EQ.0) GO TO 1680
92700 C START POINT VISIBLE
92800      IF(ISDRAW.EQ.1) GO TO 1670
92900 C START POINT INVISIBLE
93000      SX0=XX+0.00003
93100      SY0=YY
93200      ISDRAW=1
93300      ISAVE=IPR
93400      GO TO 1660
93500 1670  CALL MOVEA(SX0,SY0)
93600      CALL DRAWA(XX,YY)
93700      CALL VISIBL(IPVIS,SX0,SY0,XX,YY)
93800      INTERA=0
93900      CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPP,XX,YY,IP135,
94000      I1135ST,IND135,INTERA)
94100      IF(LINE(L).EQ.2) INTERA=1
94200      SX0=XX+0.00003
94300      SY0=YY
94400      ISDRAW=0
94500      GO TO 1660
94600 C NO INTERSECTION, START POINT INVISIBLE, BRANCH AROUND
94700 1680  IF(ISDRAW.EQ.0) GO TO 1700
94800 C NO INTERSECTION, START POINT VISIBLE
94900 1690  CALL MOVEA(SX0,SY0)
95000      CALL DRAWA(SX1,SY1)
95100      CALL VISIBL(IPVIS,SX0,SY0,SX1,SY1)
95200 C UPDATE ONLY IS TWO POINTS
95300      IF(LINE(L).GT.2) GO TO 1700
95400      IF(IUPDAT.EQ.0.AND.I135ST.EQ.1)CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,
95500      I1UPDAT,IPR,XX,YY,IP135,I135ST,IND135,INTERA)
95600 1700  IF(LINE(L).EQ.2) GO TO 1900
95700      LL=LINE(L)-1
95800      DO 1890 I=2,LL,2
95900          IM1=I-1
96000          IP2=I+2
96100          IP1=I+1
96200 C SET UP END POINTS FOR PLOTTING SEQUENCE 2-5
96300      SX1=RPOLY(I,1,L)
96400      YN=RPOLY(I,2,L)
96500      ZN=RPOLY(IM1,3,L)
96600      SY1=YN*SINALP+ZN*COSALP
96700      SX2=SX1
96800      YN=RPOLY(I,2,L)
96900      ZN=RPOLY(I,3,L)
97000      SY2=YN*SINALP+ZN*COSALP
97100      SX3=RPOLY(IP1,1,L)
97200      YN=RPOLY(IP1,2,L)
97300      ZN=RPOLY(IP1,3,L)
97400      SY3=YN*SINALP+ZN*COSALP
97500      SX4=SX3
97600      YN=RPOLY(IP1,2,L)

```

```

97700      ZN=RPOLY(IP2,3,L)
97800      SY4=YN*SINALP+ZN*COSALP
97900      SX5=RPOLY(IP2,1,L)
98000      YN=RPOLY(IP2,2,L)
98100      ZN=RPOLY(IP2,3,L)
98200      SY5=YN*SINALP+ZN*COSALP
98300      1710  CONTINUE
98400      C DRAW PLOTTING SEQUENCE # 2
98500      1720  CALL HIDDEN(IPR,IPX,SX1,SY1,SX2,SY2,XX,YY,LINT,LPAR)
98600          INTERA=0
98700          IF(LPAR.EQ.1)      GO TO 1790
98800          IF(LINT.EQ.0)      GO TO 1780
98900      C START POINT VISIBLE
99000          IF(ISDRAW.EQ.1)      GO TO 1750
99100      C START POINT INVISIBLE
99200          ISDRAW=1
99300          SX1=XX
99400      C DETERMINE IF PLOTTING GOES UP OR DOWN
99500          IF(SY1.GT.SY2) GO TO 1730
99600          SY1=YY+0.00003
99700          GO TO 1740
99800      1730  SY1=YY-0.00003
99900      1740  ISAVE=IPR

00100      GO TO 1720
00200      1750  CALL MOVEA(SX1,SY1)
00300          CALL DRAWA(XX,YY)
00400          CALL VISIBL(IPVIS,SX1,SY1,XX,YY)
00500          CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPR,XX,YY,IPL35,I135ST,
00600          IIND135,INTERA)
00700          SX1=XX
00800          IF(SY1.GT.SY2) GO TO 1760
00900          SY1=YY+0.00003
01000          GO TO 1770
01100      1760  SY1=YY-0.00003
01200      1770  ISDRAW=0
01300          GO TO 1720
01400      C NO INTERSECTION, START POINT INVISIBLE, BRANCH AROUND
01500      1780  IF(ISDRAW.EQ.0)      GO TO 1790
01600      C NO INTERSECTION, START POINT VISIBLE
01700          CALL MOVEA(SX1,SY1)
01800          CALL DRAWA(SX2,SY2)
01900          CALL VISIBL(IPVIS,SX1,SY1,SX2,SY2)
02000      C DRAW PLOTTING SEQUENCE # 3
02100      1790  CALL HIDDEN(IPR,IPX,SX2,SY2,SX3,SY3,XX,YY,LINT,LPAR)
02200          INTERA=0
02300          IF(LPAR.EQ.1)      GO TO 1820
02400          IF(LINT.EQ.0)      GO TO 1810
02500      C START POINT VISIBLE
02600          IF(ISDRAW.EQ.1)      GO TO 1800
02700      C START POINT INVISIBLE
02800          SX2=XX+0.00003
02900          SY2=YY
03000          ISDRAW=1
03100          ISAVE=IPR
03200          GO TO 1790
03300      1800  CALL MOVEA(SX2,SY2)
03400          CALL DRAWA(XX,YY)
03500          CALL VISIBL(IPVIS,SX2,SY2,XX,YY)
03600          CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPR,XX,YY,IPL35,I135ST,
03700          IIND135,INTERA)

```

```

03800      SX2=XX+0.00003
03900      SY2=YY
04000      ISDRAW=0
04100      GO TO 1790
04200      C NO INTERSECTION, START POINT INVISIBLE, BRANCH AROUND
04300      1810      IF(ISDRAW.EQ.0)      GO TO 1820
04400      C NO INTERSECTION, START POINT VISIBLE
04500      CALL MOVEA(SX2,SY2)
04600      CALL DRAWA(SX3,SY3)
04700      CALL VISIBL(IPVIS,SX2,SY2,SX3,SY3)
04800      C CHECK TO SEE IF LAST POLYGON
04900      1820      IF((LL-I).GT.1) GO TO 1890
05000      C DRAW PLOTTING SEQUENCE # 4
05100      1830      CALL HIDDEN(IPR,IPX,SX3,SY3,SX4,SY4,XX,YY,LINT,LPAR)
05200      INTERA=0
05300      IF(LPAR.EQ.1)      GO TO 1860
05400      IF(LINT.EQ.0)      GO TO 1850
05500      C START POINT VISIBLE
05600      IF(ISDRAW.EQ.1)      GO TO 1840
05700      C START POINT INVISIBLE
05800      SX3=XX
05900      ISDRAW=1
06000      ISAVE=IPR
06100      SY3=YY-.00003
06200      GO TO 1830
06300      1840      CALL MOVEA(SX3,SY3)
06400      CALL DRAWA(XX,YY)
06500      CALL VISIBL(IPVIS,SX3,SY3,XX,YY)
06600      CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPR,XX,YY,IP135,I135ST,
06700      IIND135,INTERA)
06800      SX3=XX
06900      ISDRAW=0
07000      SY3=YY-.00003
07100      GO TO 1830
07200      C NO INTERSECTION, START POINT INVISIBLE, BRANCH AROUND
07300      1850      IF(ISDRAW.EQ.0)      GO TO 1860
07400      C NO INTERSECTION, START POINT VISIBLE
07500      CALL MOVEA(SX3,SY3)
07600      CALL DRAWA(SX4,SY4)
07700      CALL VISIBL(IPVIS,SX3,SY3,SX4,SY4)
07800      C DRAW PLOTTING SEQUENCE # 5
07900      1860      CALL HIDDEN(IPR,IPX,SX4,SY4,SX5,SY5,XX,YY,LINT,LPAR)
08000      INTERA=1
08100      IF(LPAR.EQ.1)      GO TO 1890
08200      IF(LINT.EQ.0)      GO TO 1880
08300      C START POINT VISIBLE
08400      IF(ISDRAW.EQ.1)      GO TO 1870
08500      C START POINT INVISIBLE
08600      SX4=XX+0.00003
08700      SY4=YY
08800      ISDRAW=1
08900      ISAVE=IPR
09000      GO TO 1860
09100      1870      CALL MOVEA(SX4,SY4)
09200      CALL DRAWA(XX,YY)
09300      CALL VISIBL(IPVIS,SX4,SY4,XX,YY)
09400      INTERA=0
09500      CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPR,XX,YY,IP135,I135ST,
09600      IIND135,INTERA)
09700      INTERA=1
09800      SX4=XX+0.00003

```

```

09900      SY4=YY
10000      ISDRAW=0
10100      GO TO 1860
10200      C NO INTERSECTION, START POINT INVISIBLE, BRANCH AROUND
10300      1880      IF(ISDRAW.EQ.0)      GO TO 1890
10400      C NO INTERSECTION, START POINT VISIBLE
10500      CALL MOVEA(SX4,SY4)
10600      CALL DRAWA(SX5,SY5)
10700      CALL VISIBL(IPVIS,SX4,SY4,SX5,SY5)
10800      IF(IUPDAT.EQ.0.AND.I135ST.EQ.1)CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,
10900      IUPDAT,IPR,XX,YY,IP135,I135ST,IND135,INTERA)
11000      GO TO 1890
11100      C FOR FINISH UP WHEN IP135=1 AND STARTING LINE INVISIBLE
11200      C THEN COMES VISIBLE AT 5. INTERATION
11300      1890      CONTINUE
11400      1900      IF(IUPDAT.EQ.1) GO TO 1910
11500      IF(ISDRAW.EQ.0.AND.IUPDAT.EQ.0) GO TO 1920
11600      CALL UPDT(IPX,IPVIS)
11700      GO TO 1920
11800      1910      IF(ISDRAW.EQ.0) GO TO 1920
11900      IF(IND135.EQ.1) GO TO 1920
12000      IFIN=1
12100      C FOR FINISH OF LINE UPDATE TOP LINE WITH PVISIB
12200      IF(IPVIS.GT.0) CALL UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPR,XX,Y
12300      I135ST,IND135,INTERA)
12400      1920      CONTINUE
12500      C DUMP BUFFER
12600      CALL TSEND
12700      1930      READ(5,*) IC
12800      IF(IC.NE.1) GO TO 1930
12900      CALL MOVEA(0,780)
13000      WRITE(5,1940)
13100      1940      FORMAT(1X,' TYPE 1 TO EXIT, 0 TO CHANGE PARAMETERS AND REDRAW')
13200      READ(5,*) IFEXIT
13300      IF(IFEXIT.EQ.1) GO TO 1980
13400      IF(IROTZ.EQ.0) GO TO 1960
13500      C UNPACK DATA FROM ROTATION ABOUT Z-AXIS
13600      DO 1960 L=1,LINES
13700      IVAL=LINE(L)
13800      DO 1950 I=1,IVAL
13900      YADDZ=(YTALLZ*RPOLY(I,1,L))/XMAXZ
14000      RPOLY(I,2,L)=RPOLY(I,2,L)-YADDZ
14100      1950      CONTINUE
14200      1960      CONTINUE
14300      CALL ERASE
14400      IF(IROT.EQ.0) GO TO 1970
14500      IF(ICHANG.EQ.1) CALL CHANGE
14600      XYROT=-XYROT
14700      C ASSIGN VALUE TO VARIABLE THAT ARE SET BY USERS PARAMETERS
14800      C INPUTS SO NOT TO CHANGE PLOT NEXT TIME ROTXY IS CALLED
14900      1970      A=1.
15000      B=1.
15100      C=1.
15200      D=1.
15300      P=0.
15400      Q=0.
15500      CALL ERASE
15600      C ROTATE BACK TO STARTING POSITION IF ROTATION TOOK PLACE
15700      IF(IROT.EQ.1) CALL ROTXY(XYROT,XPOINT,YPOINT,A,B,C,D,P,Q)
15800      GO TO 1200
15900      1980      CALL FINITT(0,0)
16000      STOP
16100      END

```

```

16200
16300 C*****
16400 C
16500 C THE SUBROUTINE HIDDEN FINDS AND RETURNS THE INFORMATION ABOUT
16600 C THE INTERSECTIONS OF THE SEGMENT BEING PLOTTED AND THE VISIBLE
16700 C TOP LINE.
16800 C
16900 C*****
17000 SUBROUTINE HIDDEN(IPR,IPX, SX1,SY1, SX2,SY2, XX,YY,LINT,LPAR)
17100 COMMON PX(150,4),PVISIB(80,4),RPOLY(50,4,50),LINE(120),LINES
17200 DO 20 IPR=1,IPX
17300 PX1=PX(IPR,1)
17400 PY1=PX(IPR,2)
17500 PX2=PX(IPR,3)
17600 PY2=PX(IPR,4)
17700 CALL INTER(SX1,SY1, SX2,SY2, PX1,PY1,
17800 PX2,PY2, XX,YY,LINT,LPAR)
17900 IF(LINT.EQ.0) GO TO 10
18000 10 IF(LPAR.EQ.1) GO TO 20
18100 IF(LINT.EQ.0) GO TO 20
18200 RETURN
18300 20 CONTINUE
18400 RETURN
18500 END
18600
18700 C*****
18800 C
18900 C THE SUBROUTINE INTER FINDS AND RETURNS THE INFORMATION ABOUT THE
19000 C INTERSECTIONS OF THE TWO LINE SEGMENTS PASSED TO IT
19100 C
19200 C*****
19300 SUBROUTINE INTER(X1L,Y1L,X1R,Y1R,X2L,Y2L,X2R,Y2R,
19400 LXX,YY,LINT,LPAR)
19500 LOGICAL FLAG1V,FLAG1H,FLAG2V,FLAG2H
19600 FLAG1V=.FALSE.
19700 FLAG1H=.FALSE.
19800 FLAG2V=.FALSE.
19900 FLAG2H=.FALSE.
20000 LINT=0
20100 LPAR=0
20200 C MINIMAX TEST
20300 XMIN1=X1R
20400 IF(X1L.LT.X1R) XMIN1=X1L
20500 YMIN1=Y1R
20600 IF(Y1L.LT.Y1R) YMIN1=Y1L
20700 XMAX1=X1R
20800 IF(X1L.GT.X1R) XMAX1=X1L
20900 YMAX1=Y1R
21000 IF(Y1L.GT.Y1R) YMAX1=Y1L
21100 XMIN2=X2R
21200 IF(X2L.LT.X2R) XMIN2=X2L
21300 YMIN2=Y2R
21400 IF(Y2L.LT.Y2R) YMIN2=Y2L
21500 XMAX2=X2R
21600 IF(X2L.GT.X2R) XMAX2=X2L
21700 YMAX2=Y2R
21800 IF(Y2L.GT.Y2R) YMAX2=Y2L
21900 IF(XMAX1.LT.XMIN2) RETURN
22000 IF(XMAX2.LT.XMIN1) RETURN

```

```

22100      IF(YMAX1.LT.YMIN2) RETURN
22200      IF(YMAX2.LT.YMIN1) RETURN
22300      IF(ABS(X1L-X1R).GT..000002) GO TO 10
22400      SM1=0.
22500      FLAG1V=.TRUE.
22600      GO TO 20
22700      C FIND SLOPE
22800      10      SM1=(Y1L-Y1R)/(X1L-X1R)
22900      IF(SM1.EQ.0.) FLAG1H=.TRUE.
23000      20      IF(ABS(X2L-X2R).GT..000002) GO TO 30
23100      SM2=0.
23200      FLAG2V=.TRUE.
23300      GO TO 40
23400      C FIND SLOPE
23500      30      SM2=(Y2L-Y2R)/(X2L-X2R)
23600      IF(SM2.EQ.0.) FLAG2H=.TRUE.
23700      40      IF(FLAG2H.AND.FLAG1H) GO TO 50
23800      IF(FLAG2V.AND.FLAG1V) GO TO 60
23900      IF(FLAG2V.OR.FLAG1V) GO TO 70
24000      IF(SM1.EQ.SM2) GO TO 90
24100      C FIND B-INTERCEPT
24200      B1=Y1L-SM1*X1L
24300      B2=Y2L-SM2*X2L
24400      XX=(B2-B1)/(SM1-SM2)
24500      YY=SM1*XX+B1
24600      GO TO 100
24700      C BOTH LINE HORIZONTAL- CHECK TO SEE IF SAME
24800      50      IF(ABS(Y1L-Y2L).GT..000002) RETURN
24900      LINT=1
25000      LPAR=1
25100      XX=X2L
25200      YY=Y2L
25300      RETURN
25400      C BOTH LINES VERTICAL- CHECK TO SEE IF SAME
25500      60      IF(ABS(X1L-X2L).GT..000002) RETURN
25600      LINT=1
25700      LPAR=1
25800      XX=X2L
25900      YY=Y2L
26000      RETURN
26100      C ONE OF THE 2 LINES IS VERTICAL, THEREFOR NO B-INTERCEPT
26200      70      IF(FLAG2V) GO TO 80
26300      B2=Y2L-SM2*X2L
26400      XX=X1L
26500      YY=SM2*XX+B2
26600      GO TO 100
26700      80      XX=X2L
26800      B1=Y1L-SM1*X1L
26900      YY=SM1*XX+B1
27000      GO TO 100
27100      C LPARALLEL LINES- SAME SLOPE BUT HORIZONTAL OR VERTICAL
27200      90      DISDIA=SQRT((X1R-X1L)**2+(Y1R-Y1L)**2)
27300      DIS1=SQRT((X1L-X2L)**2+(Y1L-Y2L)**2)
27400      DIS2=SQRT((X2L-X2R)**2+(Y2L-Y2R)**2)
27500      DIS3=SQRT((X2R-X1R)**2+(Y2R-Y1R)**2)
27600      TDIS=DIS1+DIS2+DIS3
27700      IF(ABS(DISDIA-TDIS).GT..000002) RETURN
27800      LINT=1
27900      LPAR=1
28000      XX=X1R
28100      YY=Y1R

```

```

28200      RETURN
28300      C CHECK IF INTERSECTION IS WITHIN LINE SEGMENTS
28400      C TEST SECOND LINE X2L,Y2L,X2R,Y2R
28500      100      IF(X2L.GT.X2P) GO TO 120
28600              IF(XX.LT.X2L.OR.XX.GT.X2R)      RETURN
28700              IF(ABS(Y2L-Y2P).LT..000002.AND.ABS(YY-Y2L).LT..000002) GO TO 14
28800              IF(Y2L.LT.Y2R)      GO TO 110
28900              IF(YY.GT.Y2L.OR.YY.LT.Y2R)      RETURN
29000              GO TO 140
29100      110      IF(YY.LT.Y2L.OR.YY.GT.Y2R)      RETURN
29200              GO TO 140
29300      120      IF(XX.LT.X2R.OR.XX.GT.X2L)      RETURN
29400              IF(ABS(Y2L-Y2P).LT..000002.AND.ABS(YY-Y2L).LT..000002) GO TO 14
29500              IF(Y2L.GT.Y2P) GO TO 130
29600              IF(YY.GT.Y2R.OR.YY.LT.Y2L)      RETURN
29700              GO TO 140
29800      130      IF(YY.LT.Y2P.OR.YY.GT.Y2L)      RETURN
29900      C TEST FIRST LINE X1L,Y1L,X1R,Y1R
30000      140      IF(X1L.GT.X1P) GO TO 170
30100              IF(XX.LT.X1L.OR.XX.GT.X1P)      RETURN
30200              IF(ABS(Y1L-Y1P).LT..000002.AND.ABS(YY-Y1L).LT..000002) GO TO 15
30300              IF(Y1L.LT.Y1R) GO TO 160
30400              IF(YY.GT.Y1L.OR.YY.LT.Y1R)      RETURN
30500      150      LINT=1
30600              RETURN
30700      160      IF(YY.LT.Y1L.OR.YY.GT.Y1R)      RETURN
30800              LINT=1
30900              RETURN
31000      170      IF(XX.LT.X1R.OR.XX.GT.X1L)      RETURN
31100              IF(ABS(Y1L-Y1P).LT..000002.AND.ABS(YY-Y1L).LT..000002) GO TO 15
31200              IF(Y1L.GT.Y1P) GO TO 180
31300              IF(YY.GT.Y1P.OR.YY.LT.Y1L)      RETURN
31400              LINT=1
31500              RETURN
31600      180      IF(YY.LT.Y1R.OR.YY.GT.Y1L)      RETURN
31700              LINT=1
31800              RETURN
31900      END
32000
32100      C*****
32200      C
32300      C THE SUBROUTINE VISIBL SAVES THE VISIBLE PORTION OF THE SEGMENTS
32400      C BEING PLOTTED BEFORE THOSE SEGMENTS BECOME INVISIBLE
32500      C
32600      C*****
32700      SUBROUTINE VISIBL(IPVIS,X1,Y1,X2,Y2)
32800      C TO ADD TO PVISIB(80,4)
32900      COMMON PX(150,4),PVISIB(80,4),RPOLY(50,4,120),LINE(50),LINES
33000      IPVIS=IPVIS+1
33100      PVISIB(IPVIS,1)=X1
33200      PVISIB(IPVIS,2)=Y1
33300      PVISIB(IPVIS,3)=X2
33400      PVISIB(IPVIS,4)=Y2
33500      RETURN
33600      END

```

```

33700
33800 C*****
33900 C
34000 C THE SUBROUTINE UPDATE PERFORMS THE COMPUTATION NECESSARY TO
34100 C UPDATE THE TOP LINE
34200 C
34300 C*****
34400 SUBROUTINE UPDATE(ISAVE,IFIN,IPX,IPVIS,IUPDAT,IPR,XX,YY,IP135,
34500 I135ST,IND135,INTERA)
34600 COMMON PX(150,4),PVISIB(80,4),RPOLY(50,4,120),LINE(50),LINES
34700 DIMENSION TEMP(150,4)
34800 ISAME=0
34900 C FOR FINISH UP WHEN IP135=1 AND STARTING LINE INVISIBLE THEN COMES
35000 C VISIBLE AT 1. OR 5. ITERATION
35100 IF(I135ST.EQ.1.AND.INTERA.EQ.1) GO TO 130
35200 IF(IFIN.EQ.1.AND.INTERA.EQ.1.AND.IUPDAT.EQ.0)GO TO 130
35300 IUPDAT=1
35400 IF(ISAVE.EQ.1) GO TO 20
35500 XSAVPI=PX(ISAVE,3)
35600 YSAVPI=PX(ISAVE,4)
35700 PX(ISAVE,3)=PVISIB(1,1)
35800 PX(ISAVE,4)=PVISIB(1,2)
35900 IF(IPR.NE.ISAVE) GO TO 10
36000 C CASE WHERE INTERSECTION IS IN SAME LINE - ISAVE=IPR
36100 ISAME=1
36200 XSAVE=PX(IPR,1)
36300 YSAVE=PX(IPR,2)
36400 10 IF(IFIN.EQ.1) GO TO 90
36500 20 PX(IPR,1)=XX
36600 PX(IPR,2)=YY
36700 C STORE BOTTOM OF PX(150,4) INTO TEMP(150,4)
36800 DO 40 I=IPR,IPX
36900 DO 30 J=1,4
37000 TEMP(I,J)=PX(I,J)
37100 30 CONTINUE
37200 40 CONTINUE
37300 C PLACE PVISIB(80,4) INTO MIDDLE OF PX(150,4)
37400 IF(ISAVE.EQ.1) ISAVE=0
37500 DO 60 I=1,IPVIS
37600 II=ISAVE+I
37700 DO 50 J=1,4
37800 PX(II,J)=PVISIB(I,J)
37900 50 CONTINUE
38000 60 CONTINUE
38100 C PLACE BOTTOM OF PX(150,4) WHICH IS IN TEMP(150,4) BACK ONTO PX(150,4)
38200 DO 80 I=IPR,IPX
38300 II=II+1
38400 DO 70 J=1,4
38500 PX(II,J)=TEMP(I,J)
38600 70 CONTINUE
38700 80 CONTINUE
38800 IPX=II
38900 GO TO 110
39000 C ADD IN AT END OF PX(150,4) IF IFIN = 1
39100 90 DO 100 I=1,IPVIS
39200 II=ISAVE+I
39300 DO 100 J=1,4
39400 PX(II,J)=PVISIB(I,J)
39500 100 CONTINUE
39600 IFIN=0
39700 IPX=II
39800 110 IF(ISAME.EQ.0) GO TO 120
39900 IP1=ISAVE+IPVIS+1
40000 PX(IP1,3)=XSAVPI
40100 PX(IP1,4)=YSAVPI
40200 PX(IPR,1)=XSAVE
40300 PX(IPR,2)=YSAVE

```

```

40400 120 CONTINUE
40500 IPVIS=0
40600 GO TO 160
40700 130 PX(ISAVE,3)=PVISIB(1,1)
40800 PX(ISAVE,4)=PVISIB(1,2)
40900 DO 150 I=1,IPVIS
41000 ISPI=ISAVE+I
41100 DO 140 J=1,4
41200 PX(ISPI,J)=PVISIB(I,J)
41300 140 CONTINUE
41400 150 CONTINUE
41500 IPX=ISPI
41600 IUPDAT=1
41700 INDI35=1
41800 160 CONTINUE
41900 RETURN
42000 END
42100
42200 C*****
42300 C
42400 C THE SUBROUTINE UPDT REPLACES THE TOP LINE WITH THE LINE
42500 C REPRESENTED BY PVISIB(80,4) SINCE THERE WAS NO INTERSECTION
42600 C WITH THE EXISTING VISIBLE TOP LINE
42700 C
42800 C*****
42900 SUBROUTINE UPDT(IPX,IPVIS)
43000 C NO INTERSECTION (IUPDAT=0) TOP LINE IS ALL OF PVISIB(80,4)
43100 COMMON PX(150,4),PVISIB(80,4),RPOLY(50,4,50),LINE(50),LINES
43200 DO 10 I=1,IPVIS
43300 DO 10 J=1,4
43400 PX(I,J)=PVISIB(I,J)
43500 10 CONTINUE
43600 IPX=IPVIS
43700 IPVIS=0
43800 RETURN
43900 END
44000
44100 C*****
44200 C
44300 C THE SUBROUTINE ROTXY PERFORMS THE COMPUTATIONS FOR A
44400 C TWO-DIMENSIONAL ROTATION AROUND A X - Y POINT,
44500 C LOCAL SCALING AND SHEARING OF INITIAL OBJECT IN A X-
44600 C AND Y-DIRECTION, AND FOR TWO POINT PERSPECTIVE WITH THE
44700 C VANISHING POINT ON X- AND Y-AXIS.
44800 C TAKEN FROM ROGER, D. F. AND DAVIS, J. A. MATHEMATICAL ELEMENTS
44900 C FOR COMPUTER GRAPHICS
45000 C
45100 C*****
45200 C*****
45300 C
45400 C SECTION F. X - Y - Z ROTATION
45500 C
45600 C*****
45700 SUBROUTINE ROTXY(XYROT,XPOINT,YPOINT,A,B,C,D,P,Q)
45800 COMMON PX(150,4),PVISIB(80,4),RPOLY(50,4,50),LINE(50),LINES
45900 DIMENSION S(3),QQ(3),TRAN(3,3)
46000 C CONVERT TO RADIANS
46100 T2=XYROT/57.2957795
46200 C FILL TRANSFORMATION MATRIX
46300 TRAN(1,1)=A*COS(T2)+P*XPOINT
46400 TRAN(2,2)=D*COS(T2)+Q*YPOINT

```

```

46500      TRAN(1,2)=R*SIN(T2)+P*YPOINT
46600      TRAN(2,1)=C*(-SIN(T2))+Q*XPOINT
46700      TRAN(3,1)=-XPOINT*A*COS(T2)+YPOINT*C*SIN(T2)+XPOINT*(-XPOINT*P
46800      1-YPOINT*Q+1.)
46900      TRAN(3,2)=-XPOINT*B*SIN(T2)-YPOINT*D*COS(T2)+YPOINT*(-XPOINT*P
47000      1-YPOINT*Q+1.)
47100      TRAN(3,3)=-XPOINT*P-YPOINT*Q+1.
47200  C CHECK TO SEE IF P OR Q ARE EQUAL TO 0. AND FILL TRAN(3,3)
47300      IF(P.EQ.0) GO TO 10
47400      TRAN(1,3)=1./P
47500      GO TO 20
47600  10      TRAN(1,3)=0.
47700  20      IF(Q.EQ.0.) GO TO 30
47800      TRAN(2,3)=1./Q
47900      GO TO 40
48000  30      TRAN(2,3)=0.
48100      S(3)=1.
48200  40      DO 80 L=1,LINES
48300      IVAL=LINE(L)
48400      DO 70 K=1,IVAL
48500  C READ IN A ROW FROM RPOLY(50,3,50) INTO S(3)
48600      S(1)=RPOLY(K,1,L)
48700      S(2)=RPOLY(K,2,L)
48800  C MATRIX MULTIPLY QQ=S*TRAN
48900      DO 60 N=1,3
49000      SUM=0.
49100      DO 50 J=1,3
49200      SUM=SUM+S(J)*TRAN(J,N)
49300  50      CONTINUE
49400      QQ(N)=SUM
49500  60      CONTINUE
49600  C PLACE FIRST TWO ELEMENTS OF QQ(3) BACK INTO RPOLY(50,3,50)
49700      RPOLY(K,1,L)=QQ(1)
49800      RPOLY(K,2,L)=QQ(2)
49900  70      CONTINUE
50000  80      CONTINUE
50100      RETURN
50200      END
50300
50400  C*****
50500  C
50600  C THE SUBROUTINE CHANGE PERFORMS THE COMPUTATIONS NECESSARY TO
50700  C ORDER THE DIAGONALS AND DIAGONAL ENTRIES CORRECTLY DUE TO
50800  C ROTATION
50900  C
51000  C*****
51100      SUBROUTINE CHANGE
51200      COMMON PX(150,4),PVISIB(80,4),RPOLY(50,4,50),LINE(50),LINES
51300      DIMENSION TSAVE(50,3)
51400      MINTER=50
51500  C LAST LINE TO FIRST LINE
51600      LP1=LINES+1
51700      DO 100 L=1,LINES
51800      ILIN1=0
51900      ILIN2=0
52000      LML=LP1-L
52100      IDIFF=LML-L
52200  C CHECK FOR EVEN & ODD # OF LINES
52300      IF(IDIFF.LE.0) GO TO 110
52400  C CHANGE LINE L TO LINE LML, VICE VERSA THROUGH TSAVE
52500      LINE1=LINE(L)+1

```

```

52600      LINE2=LINE(LM1)+1
52700      DO 20 I=1,LINE1
52800      DO 10 J=1,3
52900      TSAVE(I,J)=RPOLY(I,J,L)
53000      10  CONTINUE
53100      20  CONTINUE
53200      30  DO 50 I=1,LINE2
53300      DO 40 J=1,3
53400      RPOLY(I,J,L)=RPOLY(I,J,LM1)
53500      40  CONTINUE
53600      50  CONTINUE
53700      60  DO 80 I=1,LINE1
53800      DO 70 J=1,3
53900      RPOLY(I,J,LM1)=TSAVE(I,J)
54000      70  CONTINUE
54100      80  CONTINUE
54200      90  ITEMP=LINE(L)
54300      LINE(L)=LINE(LM1)
54400      LINE(LM1)=ITEMP
54500      100 CONTINUE
54600      C REVERSE ENTRY IN LINE
54700      110 DO 140 L=1,LINES
54800      IMAX=LINE(L)+1
54900      DO 130 I=1,MINTER
55000      LMAX=IMAX-I
55100      IDIFF=LMAX-I
55200      IF(IDIFF.LE.0) GO TO 140
55300      DO 120 J=1,3
55400      TEMP=RPOLY(I,J,L)
55500      RPOLY(I,J,L)=RPOLY(LMAX,J,L)
55600      RPOLY(LMAX,J,L)=TEMP
55700      120 CONTINUE
55800      130 CONTINUE
55900      140 CONTINUE
56000      END

```

VITA

Edwin Fulton Abercrombie was born in Atlanta, Georgia, on October 25, 1951. He attended elementary school in McLean, Virginia. From 1963 to 1965 he attended Cooper Intermediate School in McLean. He was graduated from McLean High School in June 1969. The following September he entered Lock Haven State College for one semester. From January 1970 to January 1971 he attended the College of William and Mary. In March 1971 he entered the U.S. Navy as an electronic technician and served until March 1973. That March he entered the University of Tennessee and was graduated with honors in December 1975 with a Bachelor of Arts degree with a major in Computer Science and a minor in Geography. In January 1976 he received a graduate assistantship with the University of Tennessee Athletic Department and began working toward a Master of Science degree in Computer Science. This degree was received in June 1978.

The author is a member of the University of Tennessee Student Chapter of the Association for Computing Machinery and SIGGRAPH Subcommittee. He began employment with Union Carbide, Nuclear Division in June 1978 in Oak Ridge, Tennessee.