To the Graduate Council:

I am submitting herewith a thesis written by Stephen P. Jolley entitled "An Input System for the Creation of Three-Dimensional Shaded Graphics Pictures." 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

Ernest L. Hall, Major Professor

We have read this thesis and recommend its acceptance:

James R. Carter

Robert J Plemmons

Accepted for the Council:

L Evans Roth

Vice Chancellor
Graduate Studies and Research

Thesis
T
J649
cop. 2.

AN INPUT SYSTEM FOR THE CREATION OF THREE-DIMENSIONAL SHADED GRAPHICS PICTURES

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Stephen P. Jolley

December 1977

1346217

ABSTRACT

The purpose of this study was to develop an interactive computer program which allows a user to input three-dimensional data and which constructs a model from that data for a three-dimensional shaded graphics picture. Two goals in the design of the program were that it utilize a minimal amount of graphical equipment and that the user would require little knowledge of the structure of the model of the three-dimensional object in order to use the program. Finally, the program had to be compatible with MOVIE.BYU, a system which generates shaded graphical pictures of three-dimensional objects.

TWODEE-THREEDEE, the program created for this study, uses merely a small tablet digitizer and a graphics display scope as hardware devices in the input of the three-dimensional data. The data is input from the tablet digitizer as a series of two-dimensional cross-sections of the object, with each cross-section having a constant height (Z-coordinate) which is input as the X-coordinate of a height point. As two consecutive cross-sections are input, a heuristic forms the surface of the object by a series of triangles connecting the points of the two cross-sections. The model formed in this fashion is saved on a file in a form usable by MOVIE.BYU which creates the shaded picture in the orientation desired by the user.

TABLE OF CONTENTS

SECTION		PAGE
I.	INTRODUCTION	1
II.	REPRESENTATION OF SURFACES AND INPUT OF THREE-DIMENSIONAL DATA	3
III.	TWODEE-THREEDEE: A SYSTEM FOR THREE-DIMENSIONAL DATA INPUT FOR THE RECONSTRUCTION OF A POLYGONAL REPRESENTATION OF AN OBJECT	25
IV.	CONCLUSIONS	56
LIST OF REFERENCES		59
APPENDICES		63
APPENDIX A.	DESCRIPTION OF MOVIE.BYU	64
APPENDIX B.	USER'S GUIDE AND PROGRAM LISTING	69
VITA		108

LIST OF FIGURES

FIGURE		PAGE
1.	Manually Drawn Curve with Estimated Control Points	7
2.	Z Coordinates of Space Curve	7
3.	Examples of the Combinatorial Geometry Method	13
4.	Polygon Connection in Sutherland's System	22
5.	Block House	30
6.	Sliced Block House	30
7.	Data on Block House as It Would Be Input	31
8.	Real Apple before Slicing	33
9.	View of Shaded Apple	33
10.	Instruction Menu	35
11.	Ball	46
12.	First Attempt at Shaded Apple	48
13.	Second View of Shaded Apple	49
14.	Contour Map	49
15.	Artist's Conception of the Landscape	50
16.	View of Shaded Landscape with Extra Polygons Removed	51
17.	View of Shaded Landscape with Smooth Shading	51
18.	Engineering Drawing of Sutherland's Castle Used as Three-Dimensional Input	52
19.	View of Shaded Castle	53
20.	Second View of Shaded Castle	53
21.	Third View of Shaded Castle	54

FIGURE PAGE

22. Object Coordinate System 55

23. Screen Coordinate System 55

I. INTRODUCTION

The field of interactive computer graphics involves and has become involved in many other areas and applications of computer science. In the design of interactive graphics systems--systems where the user interacts with the computer by means of pictures--such diverse and varied subjects in the discipline of computer science as operating systems, artificial intelligence, information structures, information retrieval and hardware design are brought together. Moreover, from this combination of subjects, an interactive graphics system emerges to tackle problems, not only in the previously mentioned subject areas, but also in computer-aided design, computer animation, computer art, medicine, chemistry, and many other diverse fields where the ability to produce and modify a picture quickly is extremely beneficial.

In the early 1960's, when the study of interactive computer graphics began in earnest, the available graphical display device was only capable of creating line drawings to solve application problems. With such a device the display of three-dimensional drawings was a problem because the viewer could not tell which lines represented the closer surfaces of the object. To alleviate this difficulty, researchers developed techniques to remove those lines which were hidden by an occluding surface. This provided greater reality to pictures of three-dimensional objects; however, it was realized that even greater reality could be produced if the surfaces of the object

were "colored in." This led to shaded graphical pictures which are produced on a raster-scan monitor.

One of the major difficulties encountered in the production of either line-drawn or shaded pictures of three-dimensional objects has been how to enter the data so that the data structures necessary for the pictures can be produced. Several attempts have been made to solve this problem for the various internal representations of three-dimensional objects and these are reviewed in Section II of this thesis. Many of these attempts have involved the use of special hardware and mechanical devices and are, therefore, impractical for use in an interactive computer graphics environment which does not possess such devices.

This thesis describes a system, TWODEE-THREEDEE, for the interactive input and construction of data structures for three-dimensional objects which gives the small-scale computer graphics environment the capability to produce pictures of three-dimensional objects. The hardware that is necessary for TWODEE-THREEDEE is merely a graphics tablet and a graphical display device, which are the basic components of any interactive graphics environment. This system is described in Section III and some shaded pictures resulting from the data structures produced by it are shown.

II. REPRESENTATION OF SURFACES AND INPUT OF THREE-DIMENSIONAL DATA

Probably the most important decision in the creation of any three-dimensional graphic system is that of how to represent in the computer the surfaces which are to be displayed. The representation chosen is fundamental because all other aspects of the graphics system, such as hidden-line removal, shading, etc., are determined by this initial decision. Given the weight of this decision a brief discussion of some of the factors involved in reaching it would seem called for. Possibly the most significant factor to be considered is the environment in which the graphics system will be used. If, for example, the graphic system were to be used in a computer aided design laboratory, where the object would be physically formed by computer controlled milling machines, the internal representation of any surface would have to approximate very accurately the desired surface in every detail. However, in a computer animation environment, where the visual effect of the display of an object or scene is of greatest importance, the representation of a surface may be less accurate if the display and shading techniques are capable of transforming the inaccurate representation into an accurate display of the surface.

Another factor which can influence how a surface is internally represented is the general structure of the class of surfaces under consideration. If all surfaces under consideration were composed of planar polygons, then the most obvious choice for an internal

representation would consist of polygonal surfaces. However, if curved surfaces were predominant in the objects to be displayed, then the polygonal representation might not be the most useful, either because of inaccuracies in the model or poor display of it.

Finally, the time required for displaying the surface must be considered when choosing the internal representation of the surface. The length of time allowable for display is determined by the use made of the display. In the production of motion pictures by computer, the display of a scene may need to be generated in less than a thirtieth of a second. However, in other graphics environments such speed may not be required or even beneficial. The speed with which a surface can be displayed is determined largely by the techniques of hidden line- or surface-removal and shading available for the particular internal representation of the surface used. Hence, the relation between the internal representation of a surface and the display techniques used is two-way, especially important if speed of display is a significant factor.

The decision regarding how one represents a picture of an object internally significantly affects the method used to generate that internal representation. Consequently, when the fundamental decision concerning internal representation of a surface has been reached, the means for producing the internal representation have been limited. Although considerable research has been devoted to producing realistic pictures of the internal representations of objects, far less time has been spent on the problem of producing the representation. This problem is far more difficult for the creation of three-dimensional object

representations than for those of two-dimensional objects. Hence, this section will review past efforts of the generation of three-dimensional objects with regard to their method of surface representation before proceeding to describe this author's method in the following section.

Patch Representations

Some of the most significant approaches to the description of surfaces, especially in the field of computer aided design, are the parametric patch techniques largely developed by Coons and his colleagues (1,2). In these techniques a surface is represented by an assembly of smaller surfaces which are joined at the edges to form adjacent patches with any desired order of continuity. The patch techniques have several good attributes. First, they can be used to fit a surface to a grid of three-dimensional points and because the surfaces are parametric, the curves on which these points lie may be three-dimensional space curves instead of planar curves. Secondly, when a change must be made in the surface, its effect can be restricted to one patch, one patch and its adjacent patches, or as many patches as the designer requests, depending upon the desired order on continuity. Thirdly, if more detail is required in a localized area of the surface, patches may be subdivided to permit more control. However, a serious problem with parametric patching techniques is the maintenance of continuity along the edges of the patches (3, p. 1182).

One implementation of the parametric patching techniques is Bézier's Système Unisurf, which is used in the design of Renault car

bodies. The hardware consists of a large (5' × 24') drawing board and a 3-axis NC milling machine on-line to an 8K computer. The method for generation of the surface involves creating a sufficient number of space curves to define a patch, after which, further patches are created one at a time and joined to the previously created assembly of patches.

Since the fundamental unit of a patch is the space curve, the method for inputting it should be discussed. First, the designer draws a freehand curve with a pencil on the large drawing board and estimates the control points (beginning and ending points of the curve and points defining the slope of the curve at the endpoints) (Fig. 1). Then the drawing head of the machine is placed on each of the control points and the X-Y coordinates are noted on paper from a digital readout. The computer may then be given the coordinates of these points and asked to draw the curve. Whereupon, if the curve is significantly different from that sketched by the designer, the points defining the slopes are modified and the model redrawn. Once the designer is satisfied with the model, he completes it by repeating the process and specifying the Z coordinates of the control points (Fig. 2).

The system in its present prototype state has several disadvantages. One is that a data structure of the model is not maintained during the design process, except on paper by the designer. Secondly, it is up to the designer to maintain continuity conditions as he moves from one patch to another. However serious these disadvantages are, they are somewhat offset by the ability of the system quickly and cheaply to cut the surface designed (3, pp. 1192-1194).

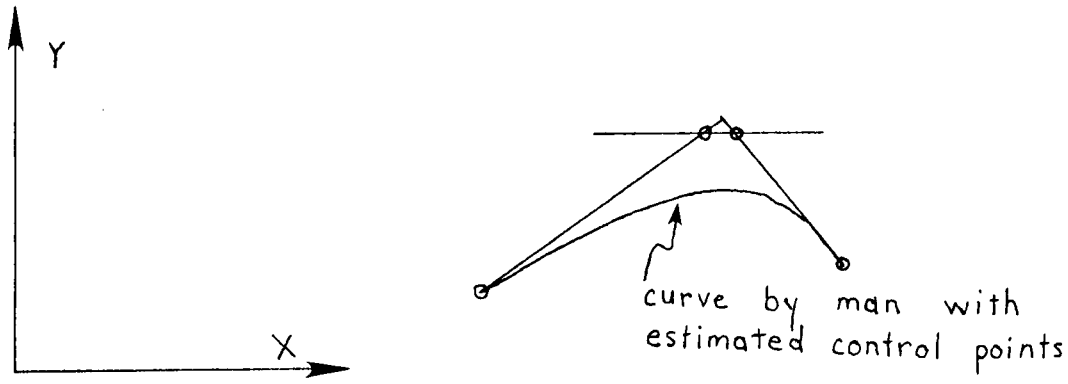


Fig. 1. Manually drawn curve with estimated control points (3).

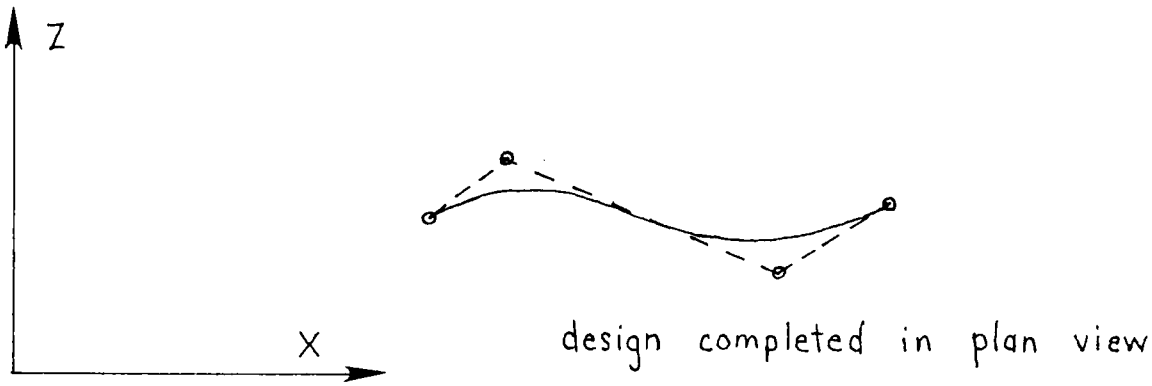


Fig. 2. Z coordinates of space curve (3).

Another system which uses a patching technique in its representation of surfaces is the Multipatch Design System at Cambridge University (4). This system uses bicubic patches, or Coons' patches as they are better known (1), which require a small amount of information for their definition, but a single patch is restricted in the range of shapes it can adopt. With this system, as with the Système Unisurf, the initial design of a surface patch must be formulated by the user without help from the system. The user brings to the system the approximate patch coordinates and the layout of patches on a sheet of paper. Using commands which are either typed in at a terminal or selected with a light pen, depending upon which version of the system is used, the user creates a patch with vertices labeled A, B, C, and D and a name for the patch. Then with the commands he may modify the point coordinates, the slope vectors at the end of each cubic edge or the twist vectors at each vertex. This information completely defines a patch. The Multipatch System provides some advantages over the Système Unisurf: (1) the 8K PDP7 on which the system runs maintains a data structure for the surface, and (2) the system maintains the desired continuity conditions between patches. However, the surface cannot be milled, but only displayed with three 3rd angle views and an arbitrarily rotated view (4,1).

A third and more recent system involving patching techniques is the one created by James H. Clark at the University of Utah (5). As has been noted in the previous two systems, mathematical knowledge of the surface patches (i.e., slope vectors and/or twist vectors) utilized by these systems were required.

However, in Clark's system one of the major goals was to reduce and, if possible, eliminate the requirement that the designer possess mathematical knowledge of the surface. In addition, Clark desired that the formulation must automatically, or intrinsically maintain surface continuity to C^2 (continuous second derivative), while allowing changes in the shape information to be specified by the movement of control points which intuitively affects the surface. He found that this could be done by using surface patches whose edges were parametric basis splines, specifically B-splines.

A second major goal of Clark's system was that the design take place in a three-dimensional environment, i.e., that the designer should be able to view the environment with a 3-D viewing device and design the surface with the aid of a 3-D input device. Hence, the system requires some novel hardware to meet this goal. The 3-D viewing device used in this system is the head-mounted display designed by Ivan Sutherland (6). The display device consists of two small CRT's and a special hardware interface which senses the position and angular orientation of the user's head with mechanical position sensors. The user is able simultaneously to see both the room and the virtual images of the CRT faces about two feet in front of his eyes. The position sensors allows the computer to "know" what the user ought to be able to see and hence presents the appropriate image on the CRT's. This allows "the observer to see it, walk around it, move closer or farther away from it, or walk inside it" (5, p. 457).

The 3-D input device was a 3-wire wand. Consisting of a wand handle with three monofilament wires attached to three spring loaded

pulleys on the ceiling, the 3-wire wand sends the length of each wire to the computer, where the wand's position is calculated as the intersection of three spheres whose centers are the pulleys and whose radii are the lengths of the wires. The wand also has several buttons which communicate to the system such things as when a control point of the surface has been grasped (5,7).

Given these special hardware devices, Clark's system required that the user select a surface description file that had previously been stored away by typing commands on a teletype. This file could be either a file from a previous working session or a simple surface such as a plane or sphere to be used to begin a more complex surface. Once an initial surface file has been selected, the display processor displays isoparametric curves, rendered as collections of straight line segments, to represent the surface. The control points are displayed as bright dots and the position of the wands as a 3/4-inch cube. In order to modify the surface, the user grasps a control point by moving the wand until the point is within the cube and pressing one of the wand buttons. As long as the button is pressed, the control point will follow the wand. The surface patches are altered as the point is moved, giving the user a sensation of stretching a flexible sheet of material (5).

The 3-D devices used in Clark's system make it very attractive. However, there are a few problems. One was that the proximity of the mechanical sensing devices of the head-mounted display and the wand made it difficult to maneuver in the room. Secondly, the B-spline surface boundaries are limited in that they cannot be of arbitrary shape.

However, the prospects of designing 3-D surfaces in a 3-D environment are very exciting.

Object Synthesis

Many common objects have surfaces which are composed of large numbers of simple faces. An object may have hundreds of faces, each of which may be planar, cylindrical or some other elemental surface form. One method with which such objects may be described is by adding together or subtracting primitive objects such as cubes or cylinders. Such a method has some advantages. First, the quantity of input is less than the patch techniques or the polygonal techniques (to be discussed subsequently). Secondly, some designers prefer to think in terms of volumes of material as opposed to faces (8).

One such object synthesis method is referred to as the Combinatorial Geometry Method and was used by Goldstein and Nazel in a system for the simulation of three-dimensional vision (9). A prime consideration in the development of this method was to make it easy for a model to be constructed with little or no mathematics. The system has available to the model designer nine bodies: a sphere, right circular cylinder, right elliptic cylinder, truncated circular cone, truncated elliptic cone, ellipsoid, rectangular parallelepiped, right-angle wedge, and arbitrary polyhedron (any convex figure bounded by up to six plane surfaces). Each of these bodies are described by certain properties, e.g., the sphere by the location of its center and its radius. Using these parameters the bodies may be positioned, oriented or scaled as

desired. A region can then be formed by the combination of two or more bodies using the Boolean operators and (+) and not (-). Figs. 3b-3f show five regions (the shaded areas) formed by the combination of a sphere (body 1) and a right circular cylinder (body 2). The model builder can employ as many of each body type as he needs and may form quite complex objects by repeated combinations of bodies.

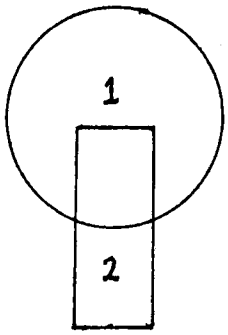
I. C. Braid (8) has also created a system in which quite short sequences of operations suffice to build up complex solids bounded by many faces. The primitive surfaces which can be used to build objects are cubes, wedges, tetrahedrons, cylinders, sectors and fillets. In addition, an arbitrary primitive can also be created by giving a sequence of line segments and axes in the XY plane. From this sequence an object of constant thickness and similar profile can be created by the program. These primitives can be scaled or rotated by using a modify command. Objects can be combined using two forms of addition which arise out of Braid's internal description of primitives, and hence objects.

At any point on the surface of an object there are two normal vectors pointing in opposite directions, either toward the interior or the exterior of the object. According to Braid's reasoning, there are then both positive and negative objects. A positive object is a mapping Q from three-dimensional Euclidian space onto the set (1, 1/2, 1) such that:

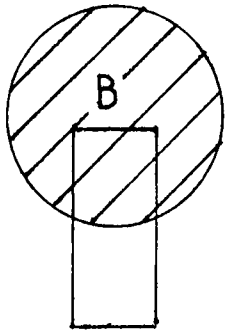
Q(q) = 1 if q is a point in the interior of the region

Q(q) = 1/2 if q is a point on the surface of the region

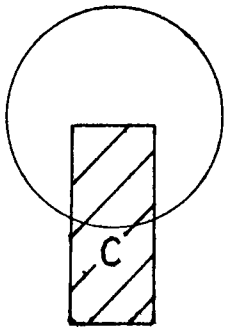
Q(q) = 0 if q is a point in the exterior of the region.



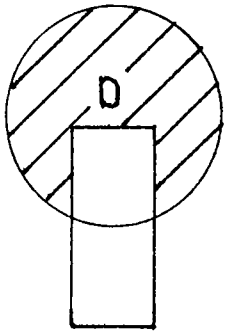
(a)



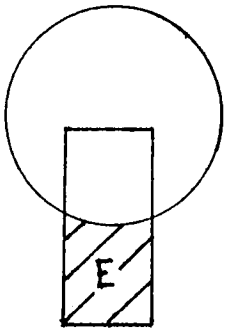
(b) B = +1



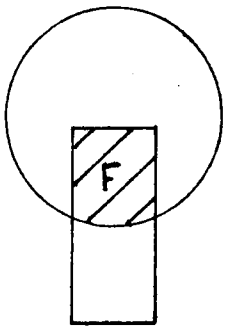
(c) C = +2



(d) D = +1-2



(e) E = +2-1



(f) F = +1+2

Fig. 3. Examples of the Combinatorial Geometry Method (9, p. 26)

For a positive object the unit normal vector of any point on the surface of an object points outward. A negative object is defined similarly as a mapping Q´ from three-dimensional Euclidian space onto the set (0, -1/2, -1). And for a negative object the unit normal vectors at any point on the surface (Q´(q) = -1/2 points inward (8).

Given these definitions of positive and negative objects, the operations of unary negation and addition can be defined. Unary negation has the effect of merely reversing the sense of the normals at the surface of the object, while not changing that surface at all. The two types of addition which Braid has defined depend upon the positioning of the two objects on which the addition is to be performed. In type-1 addition the two objects must meet at two juxtaposed flat surfaces. The effect of type-1 addition is to fuse the objects at the surfaces where they touch. This produces either a single positive or negative object whose edges are edges, or pieces of edges, of the constituent objects. In type-2 addition the objects are permitted to intersect; however, there are some restrictions. One object can be either positive or negative, but the other must be a negative cube or a cylinder. Also, they must be positioned so that no two curved faces intersect. The result is either the positive or negative sum, depending on whether the first object was positive or negative. The command for the two types of addition are COMBINE and INTERSECT for type-1 and type-2 addition, respectively (8).

Both systems for object synthesis begin with a small number of primitives and maintain that through their combination any desired object

can be represented. However, the authors of both systems admit that some restrictions have been placed on the combinations of primitives so that the operations may be performed quickly. Hence, not every object can be created by this method, and therein lies a severe restriction.

Polygonal Techniques

Probably the most popular representation of three-dimensional surfaces is their approximation by planar polygons. One of the most significant reasons for its popularity lies in the fact that no higher order equations must be solved, as is the case with mathematically curved surface approaches (the patch techniques) (10). Because of this fact, many hidden line and surface removal techniques for polygonal representations of objects have been developed (11,12,13,14,15,16,17,18). This representation "offers the only means of reducing hidden surface computation to within reasonable bounds, without restricting the class of surfaces that can be represented" (11, p. 312). Also several techniques have been devised for the shading of polygonal approximation of surfaces (12,19,10).

With such sophisticated techniques as those mentioned above, the quality of computer-produced pictures has increased dramatically. However, until recently the method for getting the necessary data into the computer has remained quite primitive (20). The following paragraphs will survey the available techniques.

For polyhedral scenes Shirai and Tsuji describe a method for constructing the description by photographing the scene with sequential

illumination from several directions. The construction process involves picture-processing techniques, such as spatial differentiation, filtering, thinning and tracking, and also such artificial intelligence processes as line-fitting and detecting the edges. The picture-processing techniques are applied to four images of the scene, corresponding to four orthogonal angles of illumination and the images are input to the computer by means of a scanner. The main goal of this work was the construction of a model of an object, where the only available representation is two-dimensional (21).

B. G. Baumgart has developed a system called GEOMED which deals with scenes not polyhedral in nature and which accepts input to the computer by a video interface. The system also uses several views of the same scene and many of the same picture-processing techniques as Shirai and Tsuji in the preprocessing stage. Then a polyhedral approximation to the surface of the object is obtained. This technique is referred to as geometrical modeling (21).

Negroponte is interested in the area of computer sketch recognition, one aspect of which is to enable the computer to input orthogonal views of an object which are drawn freehand and to construct a model of the three-dimensional object. This involves computer "understanding" of line drawings and "graphical gestures," and hence an interesting application of artificial intelligence techniques in computer graphics (22).

Chandrasekaran and Parent (21) approach the modeling of three-dimensional objects from the perspective of a sculptor. Because

of this approach, they have attempted to model their entire system after a sculptor's studio. Hence, they rarely communicate to the computer by means of command language, but instead utilize a joystick, 16 function buttons, sonic pen and 10 dials for such operations as scaling, rotating, translating, joining, intersecting, choice of "tools," etc. Since the sculptor usually begins with a lump of clay, this system provides the option of beginning with a system-supplied primitive or a primitive created by the sculptor by one of two means.

With this primitive polyhedral object displayed in a wire-frame mode, the sculptor may do many things. He may rotate and cut it, with several possible tools; he may shape the object utilizing a warping routine which gives him a movable cursor which he can attach to the object at any point and "pull" or "push" it as he desires; he may temporarily put an object aside, sculpt another one and then position the two objects for joining, overlapping or intersecting; or, he may scale the object along any one of the axes. Such varied options, in fact, present more capabilities to a user than a sculptor normally possesses.

One of the most available devices for input for a graphics system is a tablet or digitizer. Lafue uses this device for reconstruction of three-dimensional objects from two or three orthogonal projections of wire-frame drawings. The projections are drawn on the tablet, with the requirement that if a face obscures a face producing identical lines, such as in a top view of a cube, the face must be drawn as many times as it occurs. This system uses artificial intelligence techniques, such

as in a top view of a cube, the face must be drawn as many times as it occurs. This system uses artificial intelligence techniques, such as heuristics based on the properties of three-dimensional objects and a mini-theorem-prover, in its reconstruction of the object (21).

Ivan E. Sutherland (20) has developed a method for inputting three-dimensional data from a tablet given either multiple orthogonal projections (mechanical drawings) or multiple perspective views (photographs). His system utilizes a large (3' × 4') tablet which is capable of digitizing the location of up to seven pens simultaneously. Because of these characteristics the user may divide the tablet into areas for menus of command functions and several views of the object. The areas for each view are established by digitizing the lower left and upper right corners. For each view the user must specify the viewing parameters. This is done by "exemplary specification," which involves digitizing the origin and the unit vector on each of four axes for orthographic projection (the fourth axis indicates which one will be ignored in that view) and six noncoplanar points for perspective views.

With the multiple views of an object, the coordinates of a point's position in three dimensions is indicated by pointing simultaneously to its projection in two views. For orthographic projects the three-dimensional coordinates are obtained by multiplying the tablet data $[x_a\ y_a\ x_b\ y_b]$ by a matrix m:

$$\begin{vmatrix} x_{ax} & y_{ax} & x_{bx} & y_{bx} \\ x_{ay} & y_{ay} & x_{by} & y_{by} \\ x_{az} & y_{az} & y_{bz} & y_{bz} \\ x_{aw} & y_{aw} & x_{bw} & y_{bw} \end{vmatrix}^{-1} = m,$$

where a, b are the two views and x, y, z, w are the four unit vectors from the pair of views.

For perspective views Sutherland has developed a "magic matrix" technique for calculating three-dimensional coordinates from two different sets of two-dimensional coordinates. This method mandates that photographs be taken of the object so that every point of interest on the surface is visible in at least two views of the object. For each photograph, measurements must be made of the coordinates of at least six reference points from the reference cube. Using these coordinates, x, y, z, for points a, b, c, d, e, f, and the two-dimensional coordinates of those points, u, v, the transformation matrix from three-dimensional space to two-dimensional space, T, can be solved by setting $T_{43} = 1$ and solving the following system:

$$
\begin{vmatrix}
x_a & y_a & z_a & 1 & 0 & 0 & 0 & 0 & -u_a*y_a & -u_a*y_a & -u_a*z_a \\
x_b & y_b & z_b & 1 & 0 & 0 & 0 & 0 & -u_b*y_b & -u_b*y_a & -u_b*z_b \\
x_c & y_c & z_c & 1 & 0 & 0 & 0 & 0 & -u_c*x_c & -u_c*y_c & -u_c*z_c \\
x_d & y_d & z_d & 1 & 0 & 0 & 0 & 0 & -u_d*x_d & -u_d*y_d & -u_d*z_d \\
x_e & y_e & z_e & 1 & 0 & 0 & 0 & 0 & -u_e*x_e & -u_e*y_e & -u_e*z_e \\
x_f & y_f & z_f & 1 & 0 & 0 & 0 & 0 & -u_f*x_f & -u_f*y_f & -u_f*z_f \\
0 & 0 & 0 & 0 & x_a & y_a & z_a & 1 & -v_a*x_a & -v_a*y_a & -v_a*z_a \\
0 & 0 & 0 & 0 & x_b & y_b & z_b & 1 & -v_b*x_b & -v_b*y_b & -v_b*z_b \\
0 & 0 & 0 & 0 & x_c & y_c & z_c & 1 & -v_c*x_c & -v_c*y_c & -v_c*z_c \\
0 & 0 & 0 & 0 & x_d & y_d & z_d & 1 & -v_d*x_d & -v_d*y_d & -v_d*z_d \\
0 & 0 & 0 & 0 & x_e & y_e & z_e & 1 & -v_e*x_e & -v_e*y_e & -v_e*z_e \\
0 & 0 & 0 & 0 & x_f & y_f & z_f & 1 & -v_f*x_f & -v_f*x_f & -v_f*z_f
\end{vmatrix}
\begin{vmatrix} T_{11} \\ T_{21} \\ T_{31} \\ T_{41} \\ T_{12} \\ T_{22} \\ T_{32} \\ T_{42} \\ T_{13} \\ T_{23} \\ T_{33} \end{vmatrix}
=
\begin{vmatrix} u_a \\ u_b \\ u_c \\ u_d \\ u_e \\ u_f \\ v_a \\ v_b \\ v_c \\ v_d \\ v_e \\ v_f \end{vmatrix}
$$

Once the transformation matrices have been calculated for each point, the three-dimensional coordinates for any point can be calculated. The mapping of the three-dimensional scene onto the photograph is described using homogeneous coordinate techniques as:

$$
[x\ y\ z\ 1]
\begin{vmatrix}
T_{11} & T_{12} & T_{13} \\
T_{21} & T_{22} & T_{23} \\
T_{31} & T_{32} & T_{33} \\
T_{41} & T_{42} & T_{43}
\end{vmatrix}
= [x'\ y'\ w] = w[u\ v\ 1]
$$

where $u = x'/w$ and $v = y'/w$.

Carrying out the multiplication and making a substitution produces the following equations:

$$(T_{11}-T_{13}*u)*x+(T_{21}-T_{23}*u)*y+(T_{31}-T_{33}*u)*z+(T_{41}-T_{43}*u) = 0$$

$$(T_{12}-T_{13}*v)*x+(T_{22}-T_{23}*v)*y=(T_{32}-T_{33}*v)*z+(T_{42}-T_{43}*v) = 0$$

Because T and u, v are known, the above equations are of the form ax + by + cz + d = 0. With at least two views of a point known, the following overdetermined system can be solved, if a solution exists.

$$\begin{vmatrix} a & b & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{vmatrix}$$

This produces the three-dimensional coordinates. Given these the polyhedral representation can be formulated.

Sutherland's system requires that the user indicate how the points are connected to define the polygons which form the "skin" of the object. His first attempt was a two-pass approach, where in the first pass the three-dimensional coordinates of the points were calculated and stored, and in the second pass the user retouched the points in sequence to define each polygon. This proved to be too slow and cumbersome. Hence, a second approach was tried which involved only one pass. This involves defining rows of polygons by indicating the points that lie on either side of a row. For example, in Fig. 4, if all the A points were defined and then all the B points, the program would automatically generate polygons beginning with A1, A2, B2, B1 and ending with A4, A5, B5, B4. Triangles may be forced by repeating one point twice. Otherwise, all polygons will be quadrilaterals.

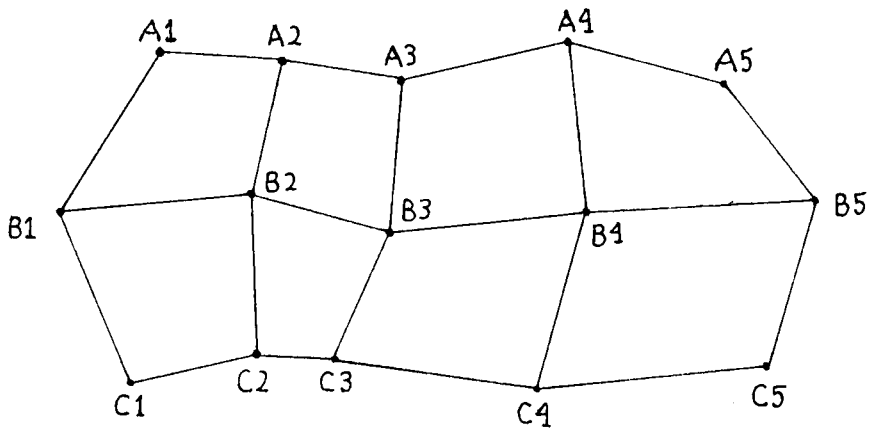


Fig. 4. Polygon connection in Sutherland's system

Frederic I. Parke (23) has developed a method for inputting three-dimensional data which involves the use of both standard photography and a tablet, and he has utilized it in obtaining data for the polygonal representation of several human faces and a human body. Parke views the camera as a device for mapping a three-dimensional space into a two-dimensional space. Using the same mathematics as Sutherland, Parke is able to utilize computing power to determine the three-dimensional coordinates of points on the surface of an object given at least two two-dimensional views (photographs) of every digitized point. In addition to the two-dimensional coordinates of points obtained in the photographs, the three-dimensional coordinates of at least six reference points are required. Parke has met this requirement by placing a reference cube about the object and physically measuring the three-dimensional coordinates of the corners of the cube. It is also necessary to digitize the same points on the photograph. This requirement was met on smooth surfaces by drawing a grid on the surface before photographing it.

Errors can quite easily enter this process. The obvious source of error is in measuring the three-dimensional coordinates of the reference cube. Another source is in digitizing the two-dimensional coordinates from the photographs. Parke has minimized this source by enlarging the photographs to take advantage of maximum tablet resolution. A third error source, computational errors, is reduced by using good numerical techniques such as Gaussian Elimination with partial pivoting.

Conclusion

After surveying the various representations used in graphics of three-dimensional objects it would appear that the polygonal surface representation is best suited to the graphics environment. This is due in large measure to the simple mathematics involved in its use; yet also it has proven to be easy to work with in hidden-surface removal and shading, producing very realistic pictures. However, the means of inputting data for polygonal representations require special equipment which is unavailable to small graphics installations or require that the user, to a large extent, indicate how the polygons will be formed. The following section will describe the author's system for inputting three-dimensional data for polygonal representations which avoids these difficulties.

III. TWODEE-THREEDEE: A SYSTEM FOR THREE-DIMENSIONAL DATA INPUT FOR THE RECONSTRUCTION OF A POLYGONAL REPRESENTATION OF AN OBJECT

Purposes of Research

From the survey of previous work done on inputting three-dimensional data in a graphics environment which was given in the preceding section, it was noted that quite often the methods employed used either special equipment or required a large amount of computer time. For example, Clark used a three-dimensional wand and a head-mounted display and Sutherland used a large tablet with multiple-pen input. Meanwhile, several other, such as Shirai and Tsuji, and Baumgart, used picture processing techniques to input data for their systems which required a scanner. These systems with their special equipment would be unsuitable in an environment where three-dimensional graphics are only sometimes used because the equipment would not be cost-effective.

However, almost any graphics environment can afford and utilize in other graphics applications a small graphics tablet with a single input pen. Hence, one of the major purposes of TWODEE-THREEDEE was to devise a method for inputting three-dimensional data using only a small tablet with single-pen input capability.

It was noted, particularly in Sutherland's work, that the means for forming the polygons which make up the surfaces of an object were largely dependent upon the user. In that system the user input two rows of points and the program formed the polygons outlined by those two rows. Hence, the user always had to be concerned with the surface polygons as

he input an object. This would seem to put an additional burden on the user in the input process. To alleviate this burden TWODEE-THREEDEE takes the points which were input as horizontal slices of the object and formulates the polygons which make the surface of the object, connecting two slices.

Hardware and Software

In its present implementation TWODEE-THREEDEE involves two different computers on The University of Tennessee campus. The program actually runs on the DEC system-10 which is a 36-bit machine with a KL-10 processor and 256 thousand words of memory in a 1080 configuration. The input to the program comes from a TEKTRONIX 4953 Tablet and a TEKTRONIX 4010 Computer Display Terminal located in the Image and Pattern Analysis Laboratory of the Electrical Engineering Department. These hardware devices constitute the minimal amount of equipment necessary for data input. The larger TEKTRONIX 4954 Tablet and either the TEKTRONIX 4014 or 4015 Computer Display Terminals may also be used. The 4953 Tablet has a flat writing surface with a useable area of approximately 11 x 11 inches and processes a grid of 1024 x 1024 points with 100 points equalling one inch on the tablet. The pen which is used in conjunction with the tablet contains a small electromagnet which enables the tablet to detect its position on the surface when it is in "presence" (within 1/4 inch of the tablet surface). It operates in two modes: (1) in pen mode a point is sent only when the pen is depressed, and (2) in presence mode points are digitized and transmitted as fast as the baud rate will permit when the pen is in presence (24). The 4010 Computer

Display Terminal is capable of displaying both alphanumeric characters and graphical data. Once written upon, the display remains visible until it is erased, and it is not necessary to refresh continually the information put upon the screen. The addressable coordinates in both the X and Y directions range from 0 to 1023. However, in the Y direction 780 is the maximum displayable point coordinate (25).

The graphic display of line drawings can also be viewed on the Computer Display Terminal; however, for shaded graphical output other hardware must be used. At present when a shaded picture is requested a file is created on the DEC system-10 and transferred to magnetic tape. This tape is taken to the Image Processing Laboratory and input to a disk on a PDP 11/40. There the picture is displayed on either a TEKTRONIX 670A-1 Color Monitor or a TEKTRONIX 632 Picture Monitor, both of which are raster scan devices capable of displaying 512 × 512 pixel pictures.

The input system is written largely in standard FORTRAN IV and utilizes two software packages made available by TEKTRONIX. These are release three of the PLOT-10 Terminal Control System and the TABLET Utility Routines, which are both written in machine-independent FORTRAN. PLOT-10 is capable of displaying bright and dark vectors as well as points on the screen. (Dark vectors are created by move commands which draw invisible lines.) Also included in the 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. The TABLET routines permit the input of single or

multiple points when the tablet is in pen mode and multiple points when it is in presence mode (25).

For outputting the final picture MOVIE.BYU, a general purpose computer display system created at and available from Brigham Young University, was used. MOVIE.BYU is capable of producing output pictures for both line-drawing display terminals or raster scan display devices with hidden-lines or -surfaces removed. It is described in more detail in Appendix A.

The Slice Method for Inputting Data

Because the information processed by a tablet is necessarily two-dimensional, the data describing a three-dimensional object must be reduced to data about a two-dimensional object. Some systems discussed in Section II used photography to do this (Parke, Sutherland, et al.) and others used multiple orthographic projections (Sutherland). With either of these techniques every point which was digitized had to be input from two pictures or projections. Because the size of the tablet used was too small to place two pictures on it at the same time and still have adequate resolution, neither of these techniques was found to be useful. It could be argued that the pictures might be placed on the tablet one at a time so that the user could digitize the points that a pair of pictures had in common. However, this would require the user to maintain an account of which points had been digitized and from which views and to maintain the same order in digitizing the common points in a pair of pictures. Hence, that technique would place quite a responsibility on the user.

Since the tablet was relatively small, especially in comparison with the one used by Sutherland, it appeared that any method for inputting three-dimensional data from the tablet, without putting the burdens invoked by the two-view method (used by Sutherland and Parke) upon the user, would require that all information about a three-dimensional point be contained in one two-dimensional view. This is accomplished by a slicing technique. One may visualize this technique as physically slicing the object at various intervals along one of the three axes. The coordinate, r, of the axis along which the slicing takes place (usually horizontal slicing) is input on a scale across the tablet. For a particular slice that information need only be input once and applies to all points input on that slice. Once this information has been input the various points along the surface of the object are input as the coordinates touched on the tablet with a third coordinate of r.

As a simple example, consider the block house in Fig. 5. It can be horizontally sliced as shown in Fig. 6. The data describing this house would be input in three slices consisting of the foundation, the ceiling and the pinnacle of the roof. The actual points necessary for this are shown in Fig. 7. These points, along with the height coordinate (also shown in Fig. 7), adequately describe the surface of this house, if the surface polygons are correctly formed connecting the points of each successive pair of slices.

It would seem appropriate to consider whether an object can be fully described by a series of slices and, if so, how many slices are necessary. Consider for the purposes of this discussion that slicing

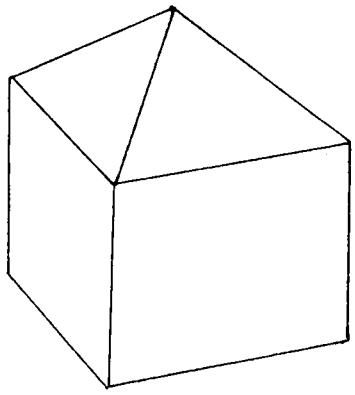


Fig. 5. Block house.

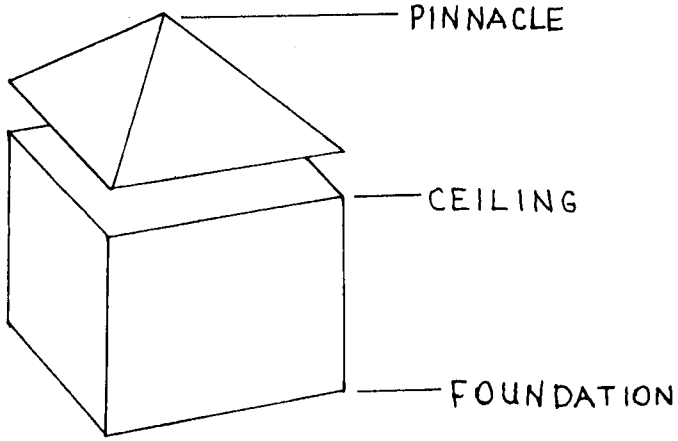
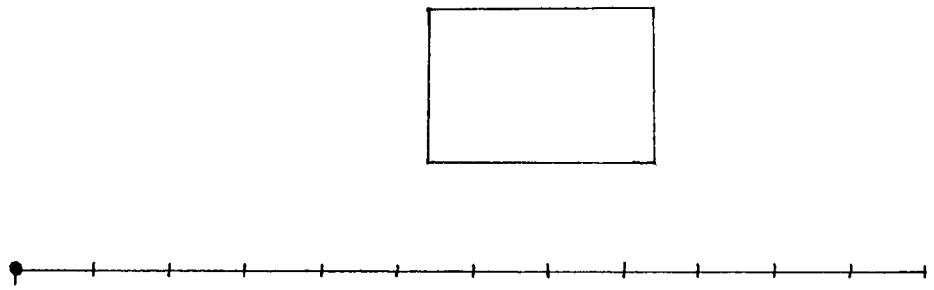
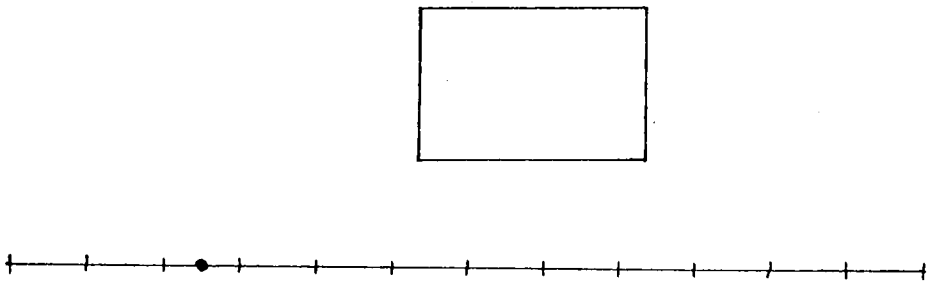


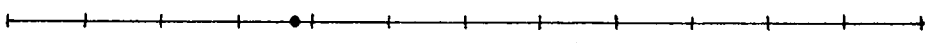
Fig. 6. Sliced block house.



(a) Foundation with input mode of POLYGON.



(b) Ceiling with input mode of LINE.



(c) Pinnacle with input mode of POINT.

Fig. 7. Data on block house as it would be input.

will be made perpendicular to the vertical or Z-axis passing through the object's center. Hence, the surface of the object can be considered a function of Z. For the continuous case (i.e., $\Delta Z \rightarrow 0$) it is intuitively obvious that slicing at each $Z_i = Z_{i-1} + \Delta Z$ completely describes the object. However, with discrete ΔZ this is not so obvious. However, since the object's surface can be expressed as a function of Z, the Sampling Theorem implies that the object can be fully described if slices are taken at least twice as frequently as the greatest frequency of "wiggles" on the object's surface. Thus, the Sampling Theorem establishes a lower bound on the number of slices necessary to fully describe an object. Admittedly, this is a theoretical lower bound and many practical examples require fewer slices to describe the object fully.

One might also consider how readily available slice data is. It can be obtained in several ways. One method which was used for the example in Figs. 8 and 9 is actually to slice the object and input points around the surface of the object. However, this technique is limited because it destroys the object. Another possibility is to use orthogonal projections of the object. The actual slice information is located on the plan and the Z-coordinates from each slice can be obtained from the elevation view. A third technique for obtaining the slice data utilized computer trans-axial tomography, an X-ray technique which actually produces pictures of objects at various distances along one axis. Thus, there are a variety of data-gathering techniques which make the slice method viable as an input method.

Hence, the slicing technique described above produces two distinct, positive effects: (1) it allows three-dimensional information to be

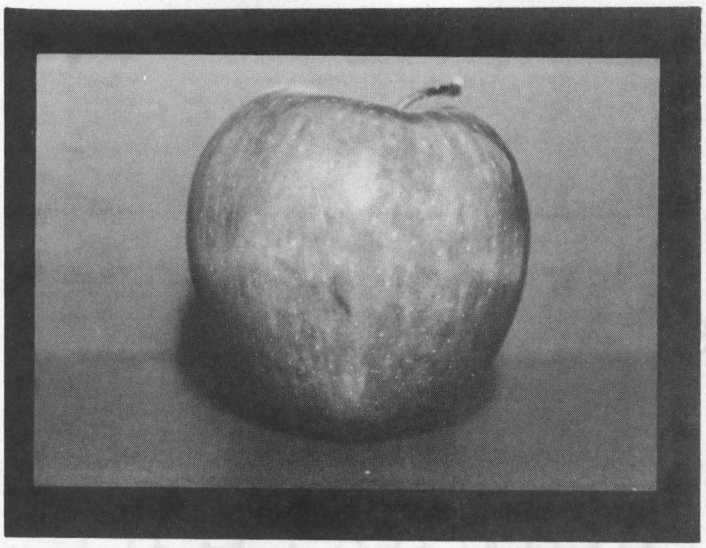


Fig. 8. Real apple before slicing.

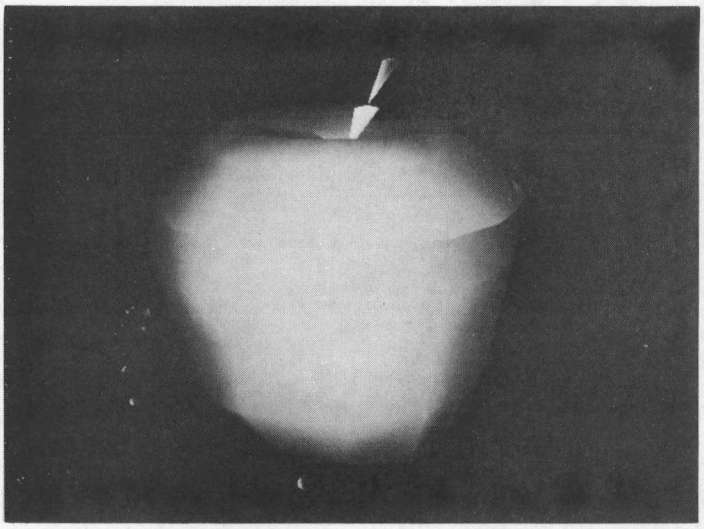


Fig. 9. View of shaded apple.

described in one two-dimensional view, and (2) it removes the burden from the user of keeping track of points in multiple views of the object. These effects permit the use of the small tablet.

However, this technique also has some minor drawbacks. The user must be accurate in giving the height coordinate for each slice or the resulting picture could be misshapen in the vertical direction. Also, the slices must be aligned properly on the tablet or the resulting picture could be vertically skewed. Hence, there is also a burden on the user with the slicing technique, but it would appear to be easier to bear than the one from the two-view method, especially in a small-tablet environment.

Description of User Controls and Commands

The means for interactively controlling the program largely consists of selecting a function from a two-tiered "menu" of functions which is shown in Fig. 10. The first thing that is done when the program begins is to reserve the location for the menu on the tablet. The location could be at any place on the useable area of the tablet; however, the upper portion is recommended because it cannot be displayed on the Tektronix scope. The location of the menu is reserved by touching six points in the horizontal direction, corresponding to the six intersections in the grid along the top edge of the menu, and three points in the vertical direction, corresponding to the intersections along the left-hand side of the grid. The user can use a printed plastic overlay or a drawing of the menu. If the user is unfamiliar with the design of the menu, the program will display it on the screen at the user's request.

INPUT POINT	SAVE POLYGON	LIST	SCALE	STOP ADD
SHOW LINE	HELP EOP	ROTATE	CHANGE DELETE	RECALL MOVE

Fig. 10. Instruction menu.

As mentioned in the description of the menu and illustrated in Fig. 10, the menu has two tiers: the upper one, consisting of the commands written above the line in each box, are the major commands and are to be given on hearing a loud beep; the second tier of commands, written below the lines in each box, are subcommands given at various times in the process of executing two of the major commands. All of these commands will be described in the following paragraphs.

The INPUT command calls upon a series of subroutines which allows the user to input slices of the object and which forms the polygons which make up the skin of the object. These subroutines form the major portion of the system because they establish the internal description of the object. Used in the input process are the four subcommands, POINT, LINE, POLYGON and END OF PLANE, which appear on the left-hand side of the menu. The purpose of these subcommands is to indicate to the program the input mode (i.e., how the data points touched next are to be connected in the internal description of the object). The POINT subcommand implies that the following point(s) have no spatial relation to any other points in

that slice and, hence, it will not be connected to any other point via a line node. An example in which this might occur is if the plane being input is the top of a pyramid. The LINE subcommand implies that a line will exist between any two consecutive points until a new input mode is given. This series of lines is stopped by touching a new input mode, which may also be LINE. The POLYGON subcommand is used for objects which have a horizontal surface, such as the base of a pyramid. However, due to the nature of MOVIE.BYU the polygons are restricted to quadrilaterals. It should be noted that five data points must be entered to form a quidrilateral, with the last point being approximately the same as the first. The END OF PLANE subcommand serves two functions: (1) to indicate that a slice has been completely entered so that the appropriate subroutines may form the surface polygons between the slice and the last one entered, and (2) to signal the completion of an object's description. The latter function is invoked when the END OF PLANE mode is entered as the first input mode in a slice.

The SHOW function is significant in that it allows the user to view the object he has created before he sees the shaded object. This ability is significant because errors in the description of the object may be detected before the shaded picture is created and, hence, these errors may be corrected in an earlier stage of picture development. The actions caused by the invocation of the SHOW function are the application of any rotations or scalings and the displaying of the object with no hidden lines removed on the TEKTRONIX 4010.

The SAVE function creates and loads a file with the description of the object in a format suitable for use by the MOVIE.BYU software

package. The name for the file must be supplied by the user. In order to make sure that the name chosen for the file doesn't duplicate the name of a file previously created, a LIST command is available as an aid to list all files previously created by the system. Another aid to the user is available through the HELP command which prints on the screen a brief description of all commands.

The ROTATE, SCALE, AND CHANGE commands are used in conjunction with the SHOW command to make limited alterations in an object after it has been input. The ROTATE command permits a rotation around any axis through a given number of degrees. This permits different aspects of the picture to be examined and to a small degree facilitates determination of front and back surfaces. The SCALE function reduces or expands the size of the object for better viewing. These two functions may be combined so that a scaled object may be rotated and vice versa.

The CHANGE command can be used to make three different types of changes in an object after it has been completely formulated through the INPUT command. These may be requested by indicating one of the subcommands on the right side of the menu. The DELETE subcommand will delete both of the polygons surrounding the line whose endpoints the user indicated. This allows the user to eliminate extraneous polygons created by the heuristic procedure for forming polygons. The ADD subcommand permits the addition of polygons which the heuristic failed to form. The MOVE subcommand is called upon if the user believes that by changing the position of a point the surface of the object will be more closely approximated.

The final two commands are RECALL and STOP. RECALL retrieves either the original description of an object which has been changed by rotations or scalings, or the description of the last object created by the program. The latter is useful if the user is unsatisfied with the picture of the object as produced by MOVIE.BYU. The STOP instruction simply calls for an orderly exiting of the program.

All of the above commands are described in greater detail in the User's Guide in Appendix B.

A Description of a Typical Input Session

The first thing that occurs during an input session is that TOODEE-THREEDEE asks the user whether he is familiar with the tablet. If not, the menu is displayed on the screen for his reference. Next the user is asked to reserve a location on the tablet for the menu, which is done as previously described. When this is completed, a loud beep will be emitted by the terminal, signaling that the program is ready for a command. For defining a new object the appropriate command would be INPUT.

The INPUT command produces a soft beep and the following message on the screen: < INPUT THE Z - COORDINATE >. The Z - coordinate for any cross section of the object is actually the X-coordinate of the point touched on the tablet following the above request. Once the Z-coordinate of the plane has been established the following message appears on the screen: < INPUT THE PICTURE COORDINATE OR CHANGE THE INPUT MODE >. On the first slice of the object the user will usually use an input mode of

either POINT or POLYGON, depending on whether the object rests on a flat surface or a single point. The last message will not reappear until the user is ready for a new slice to be input. At any time during the input of a slice the input mode may be changed. When the user is finished with a slice, he touches the EOP subinstruction and the terminal displays an orthogonal projection of the slice just input and asks if it is correct. If the user is not satisfied he may either completely redraw the slice or move one or more of the points of the projection until he is satisfied. When he is completely happy with that slice, the program will again ask for a new Z-coordinate if this was the first slice; or if it wasn't, the program will form the surface polygons between this slice and the most recent one and ask for a new slice. This process is repeated until all slices have been input. At this point the program will ask for a new Z-coordinate which the user must supply. Then, when a new input mode is requested, a subcommand of EOP signals the program that the user is finished with the object.

When the object has been input, additional parts or objects in the scene may be added by recalling the INPUT command. When all the objects or parts have been input, the user may view the scene with no hidden lines removed. He may rotate or scale the scene to determine whether it has been created according to his desires. If he is not satisfied, he may use the CHANGE command to delete or add polygons or to move points. To do this, the user must indicate to the program which points he is referring to. This is done by touching the tablet at the approximate location of the point on the screen. The program

then draws a square around the points within a small radius of the digitized point one at a time. If the program does not find the point the user wanted, the user is requested to try again until he does find the proper point. Then the requested changes may be made.

When the user is satisfied with the picture as displayed on the TEKTRONIX 4010, he can save the picture description file for use with MOVIE.BYU. Presently, because there is no interface between the DEC system-10 and the PDP 11/40, the shaded pictures created by MOVIE.BYU must be transferred on magnetic tape in order to be displayed on the TEKTRONIX 632 Picture Monitor. This is done by executing SETUP.FOR and PACK.MAC (26) and submitting PICTUR.CTL as a batch job. When the tape is ready, it is put on the PDP 11/40 and displayed using DSP3 (27).

If after viewing the shaded picture the user still is not satisfied, he may recall the picture with TWODEE-THREEDEE and make further changes. At the present time the display of shaded pictures is a long process because of the tape transfer. However, an interface between the two machines is currently being written and the entire process will be speeded up considerably when it is finished.

The Heuristic for Forming the Surface Polygons

Obviously, the most integral part of the program is that portion which forms the polygons connecting two slices together. If the polygons are not correctly linked, there may be holes in the surface of the object where they should not appear or there may be extra surfaces due to extra polygons. The capabilities for making corrections

in the polygonal representation are made available in case the user doesn't apply the guidelines for inputting data given in Appendix B.

The heuristic quite simply tries to form triangles between the points of the consecutive slices. It begins by determining which of the two slices has the most points and this one becomes the dominant slice. For each point, A, in the dominant slice, the closest point to it in the other slice, B, is found and a line is formed between them if one is not already there. Then for each of the two possible points, C, which lie in the dominant plane and are connected to A with a line, their nearest neighboring point, D, in the other plane is found. If points B and D are the same, a single triangular polygon in formed. If they are not the same, the program determines if there is a line between B and D. If so, two triangular polygons are formed, which are ACD and ABD. If no line exists between B and D, the next closest point, D´, to C is found and B and D´ are compared as before. This process is repeated for all points in the dominant plane at all times making sure not to duplicate any polygons.

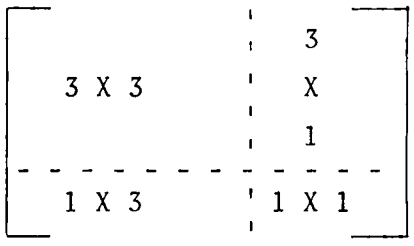
The Viewing Process

The author's system displays a perspective line drawing of the object with no hidden-lines removed when the user calls upon the SHOW command. The basic steps in this process are: (1) perform any rotations and scalings, (2) determine the centroid of the object and a viewpoint at a distance of 400 units away from the centroid, (3) move the origin of the coordinate system to the viewpoint, (4) rotate the axes until the centroid of the object is on the Y-axis, (5) transform

the object into the clipping coordinate system and perform the clipping, and (6) perform the perspective transformation. When these operations have been completed, the lines are displayed by utilizing the MOVABS and DRWABS commands of the TCS package (MOVe ABSolute and DRaW ABSolute).

The majority of the above operations are performed by matrix multiplications of the homogeneous coordinates of the vertices of the polygons by 4 x 4 transformation matrices. The homogeneous coordinates of a point [X Y Z] in 3-space can be formed by simply adding a fourth coordinate of one, giving [X Y Z 1]. The use of homogeneous coordinates permits all geometric transformations, including perspective, to be performed by matrix multiplication, thus simplifying the mathematics involved. In order to return to 3-space coordinates from a point [X Y Z W] in homogeneous coordinates, one merely divides each coordinate by W, giving [X/W Y/W Z/W 1], and the 3-space coordinates are [X/W Y/W Z/W] (12).

The general 4 x 4 transformation matrix which is used to perform the operations on the matrix of homogenous vertices can be divided into four submatrices as shown below.



Each of these submatrices performs a distinct function on the vertex matrix. The 3 X 3 matrix produces a linear transformation (i.e., one

which transforms an initial linear combination of vectors into the same linear combination of transformed vectors) in the form of rotations, scalings and shearings. The 1 X 3 row matrix produces translations and the 3 X 1 column matrix produces perspective transformations. Finally, the 1 X 1 single element produces overall sealing. Thus, one 4 X 4 matrix can perform all these transformations in one multiplication and that single matrix can be computed by successive multiplications of matrices which perform only one transformation (28).

When the user calls upon either the ROTATE or SCALE commands, a matrix is formed which produces the requested rotation or scaling and this is multiplied by a rotation-scaling matrix, RS, which keeps track of all rotations and scaling which have been requested by the user up to this point in time. This permits the user to perform several rotations and/or scalings before viewing the object. The matrix of homogeneous points are then multiplied by RS before the centroid and viewpoint are calculated. The translation of the origin to the viewpoint (VP) is performed by a matrix V which has the coordinates of the viewpoint in the 1 X 3 row vector.

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ VP_x & VP_y & VP_z & 1 \end{bmatrix}$$

The axes are rotated so that the centroid lies on the Y-axis by a combination of two rotations, one about the Z-axis and a second about

the X-axis. Then the clipping coordinate system is produced by the following matrix:

$$C = \begin{bmatrix} 2.0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and the clipping is performed by the Sutherland-Cohen 3-D clipping routine given by Newman and Sproul (12, pp. 249-254). This insures that no lines appear which are behind the viewpoint or outside of the viewport. The perspective transformation is produced by the following matrix which also reverses the effect of the transformation to clipping coordinates,

$$P = \begin{bmatrix} 511.5 & 0.0 & 0.0 & 0.0 \\ 511.5 & 0.0 & 390.0 & 1.0 \\ 0.0 & 0.0 & 390.0 & 0.0 \\ 0.0 & -1.0 & 0.0 & 0.0 \end{bmatrix}.$$

As mentioned previously, if the picture description file is saved, the object can also be viewed using MOVIE.BYU which provides some additional capabilities. Besides the ability to rotate and scale the object, which is available in the author's system, the object can be translated a specified distance from the observer, rotated by parts, drawn with contour lines and viewed with hidden-lines removed on the

TEKTRONIX 4010 scope. For display on the TEKTRONIX 632 Picture Monitor, the object can be displayed with smooth, flat or uniform shading with hidden-surfaces removed in addition to the previously stated capabilities. This gives the shaded graphics output for which the input program was designed.

Results

Five objects were chosen as examples of the effectiveness of the combination of TWODEE-THREEDEE and MOVIE.BYU. The five objects, a ball, an apple, a landscape, a castle and a body organ, represent several of the possible applications for three-dimensional graphics. They will be discussed in the order of their creation.

After verification that MOVIE.BYU had been sufficiently implemented on the DEC system-10 at the University of Tennessee, the first objects to be input were of general interest, a ball and an apple. The data for the ball was prepared for the input process by drawing six concentric circles and dividing the inner three into eight arcs and the outer three into sixteen arcs. Then the data was input as thirteen vertically equidistant cross sections, beginning with the center of the circles as the first slice and moving to the next larger circle with each successive slice until the outermost was reached. Then the smaller circles were again input in order of decreasing size until the center was input as the last slice. With the smooth shading option in effect, MOVIE.BYU produced Fig. 11 from this data.

The data for the second object, an apple, was somewhat more difficult to obtain. After an unsuccessful attempt at freehand generation of the

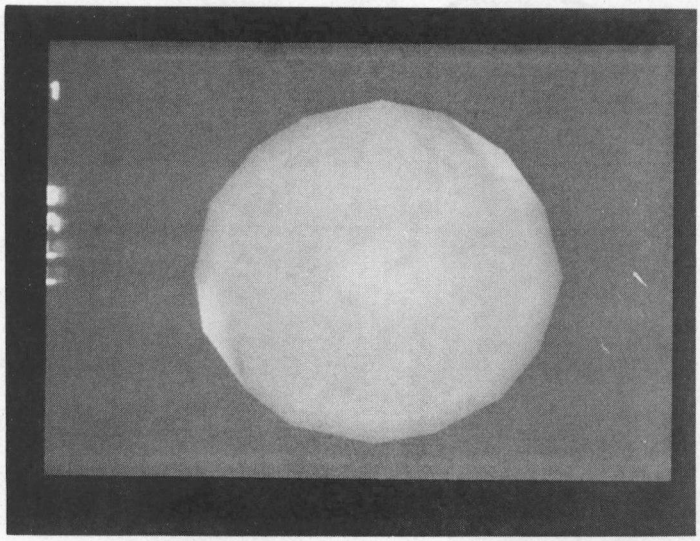


Fig. 11. Ball

sliced data (Fig. 12), an apple, shown in its original state in Fig. 13, was actually sliced. Each slice was placed on the tablet and various points along the edge of the slice were input. This produced more favorable results as evidenced in Figs. 8 and 9, page 33.

One possible application which seemed well suited to the method of input was to generate a three-dimensional landscape from a contour map. Because of the exceedingly large amount of data available on the map and due to the limitations on the quantity of data in the program, only every other contour line was input from the map in Fig. 14. Fig. 15 is an artist's conception of the landscape. Fig. 16 is the final result utilizing flat-shading. The smooth shading technique appeared inappropriate for the example as evidenced by Fig. 17.

A further application of three-dimensional graphics is in the area of architecture. A castle, which was used in Sutherland's paper (20, p. 461), was used as example of this application. The information required was merely a drawing of the plan view and the elevation view (Fig. 18). In this case, to avoid extraneous polygons, like those created in the landscape, each wall or turret was created as a separate object. The results can be seen in Figs. 19, 20 and 21.

The final example is a body organ, thus illustrating possible uses of the system in medicine. The slices used as input were created automatically from a computed tomography scanner. The resulting pictures of the object are in Figs. 22 and 23.

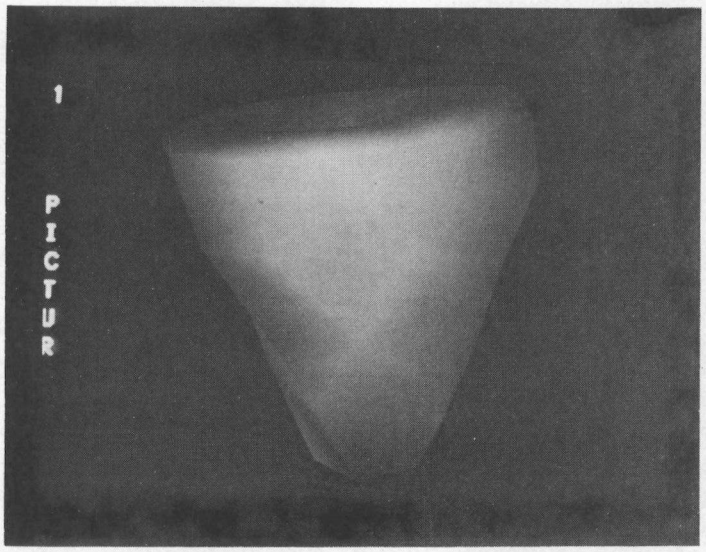


Fig. 12. First attempt at shaded apple.

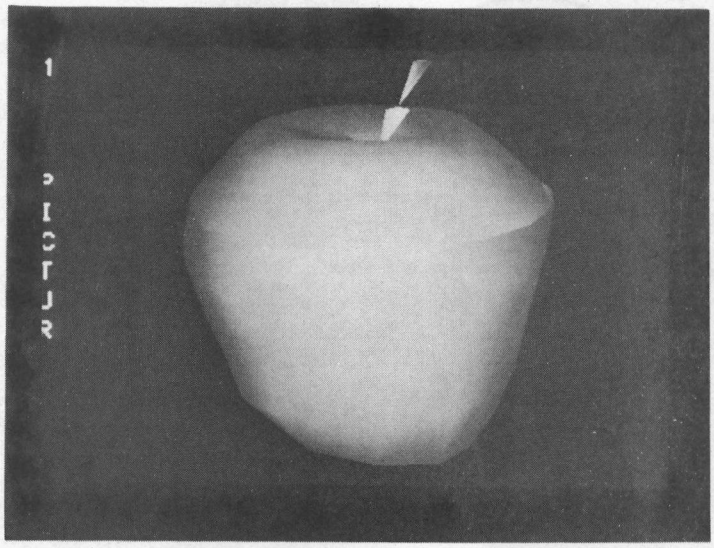


Fig. 13. Second view of shaded apple.

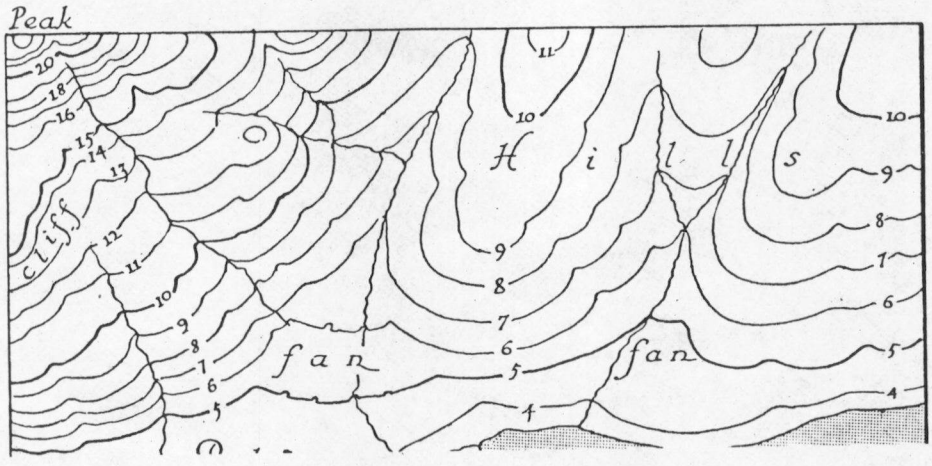


Fig. 14. Contour map.

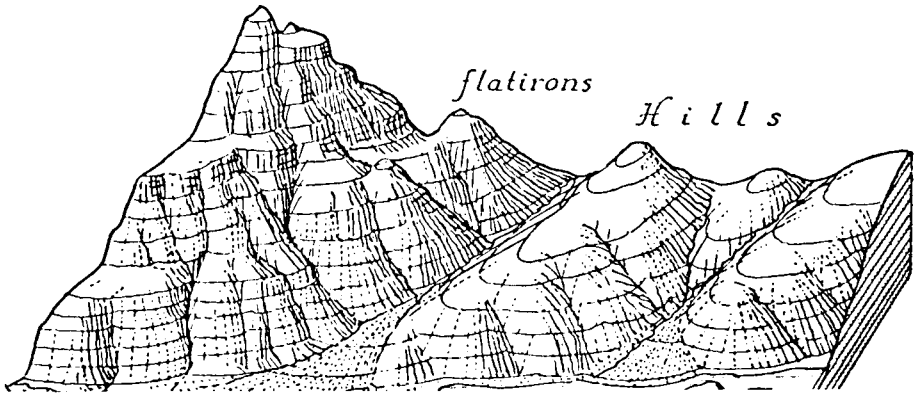


Fig. 15. Artist's conception of the landscape.

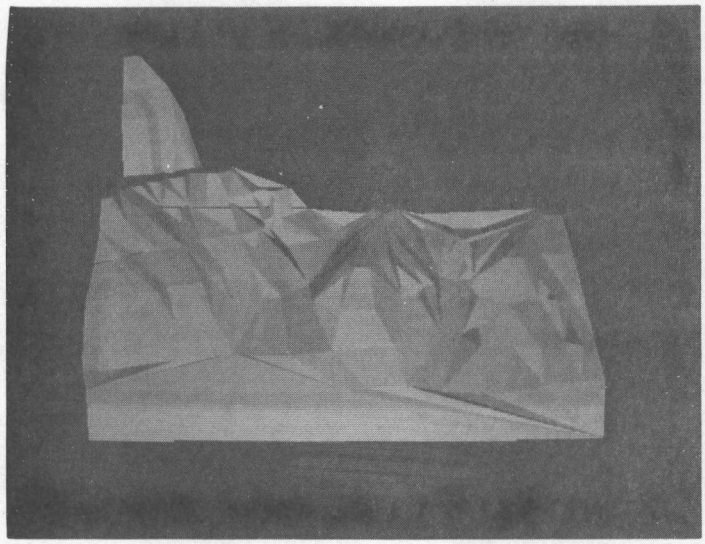


Fig. 16. View of shaded landscape with extra polygons removed.

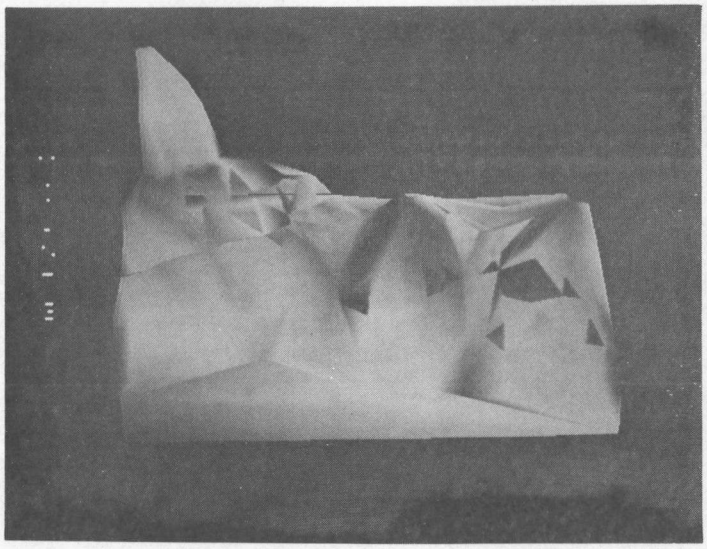


Fig. 17. View of shaded landscape with smooth shading.

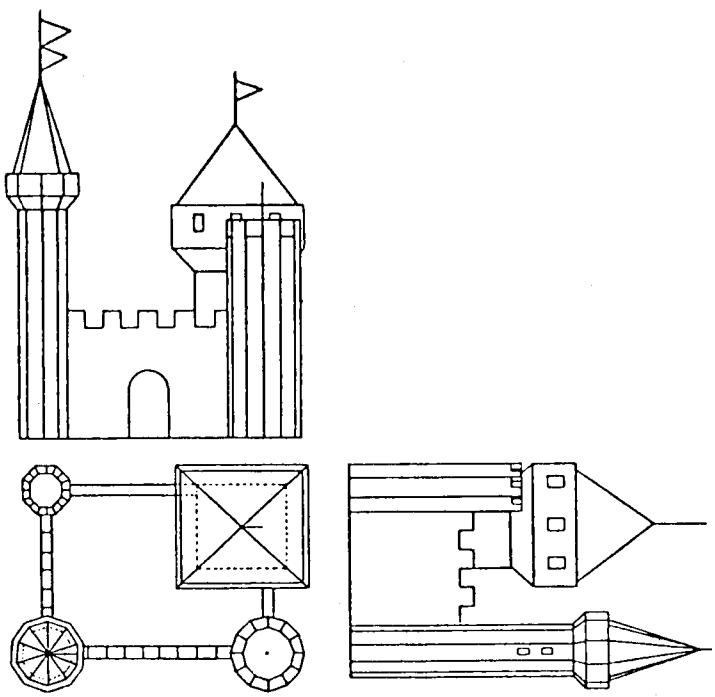


Fig. 18. Engineering drawing of Sutherland's Castle used as three-dimensional input.

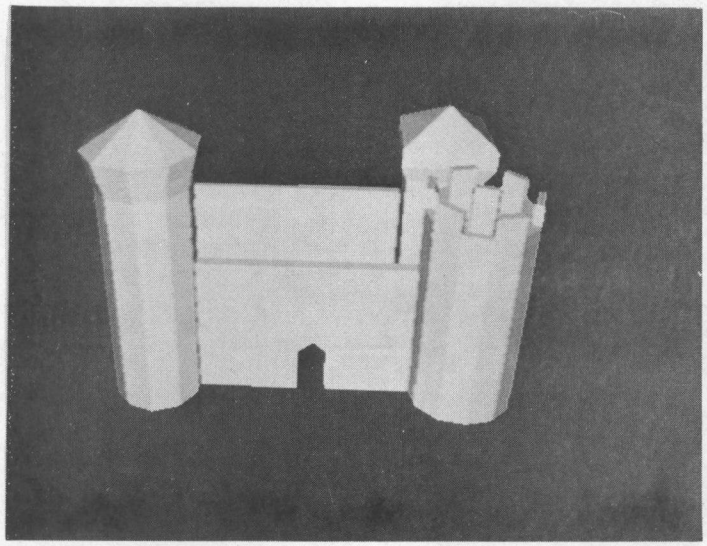


Fig. 19. View of shaded castle.

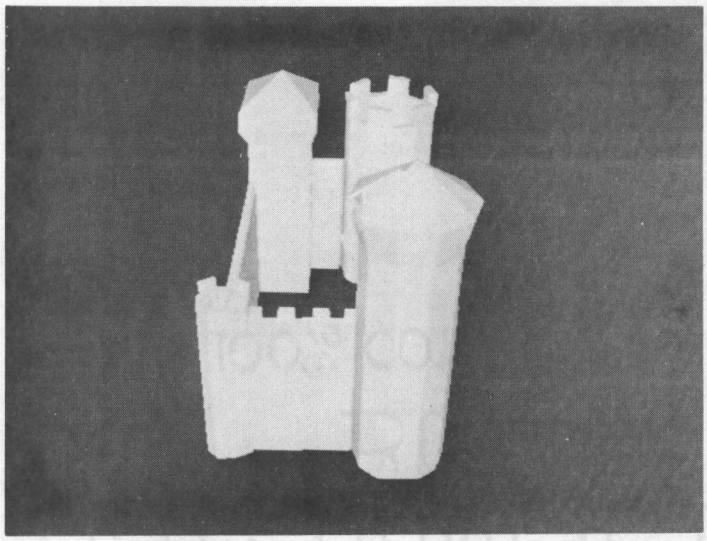


Fig. 20. Second view of shaded castle.

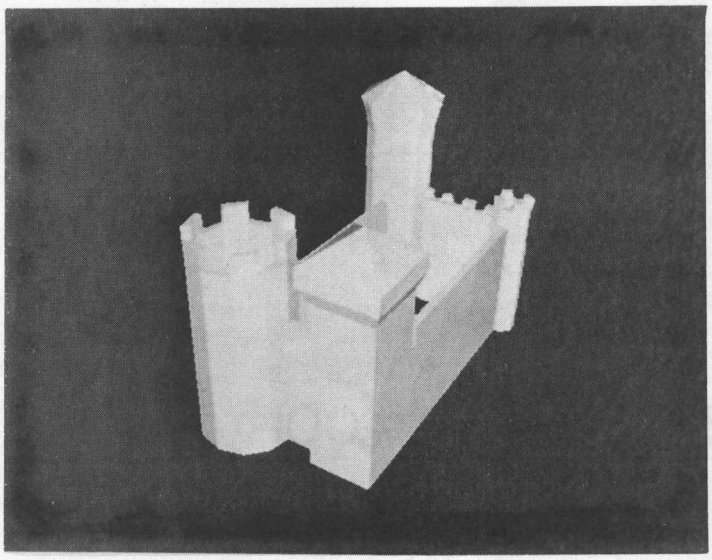


Fig. 21. Third view of shaded castle.

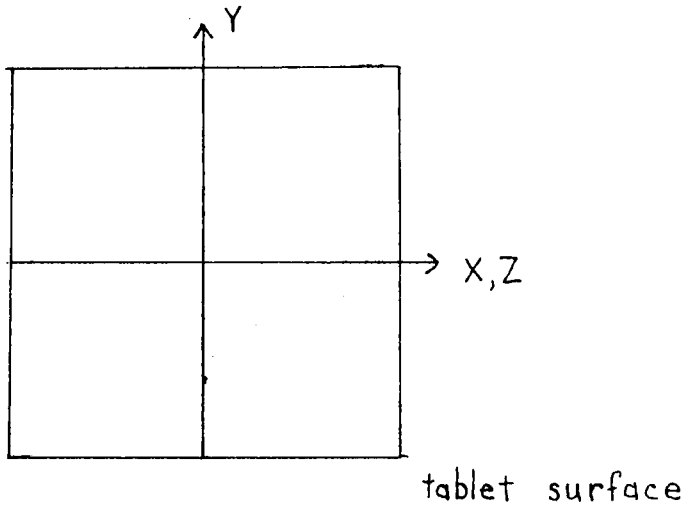


Fig. 22. Object coordinate system.

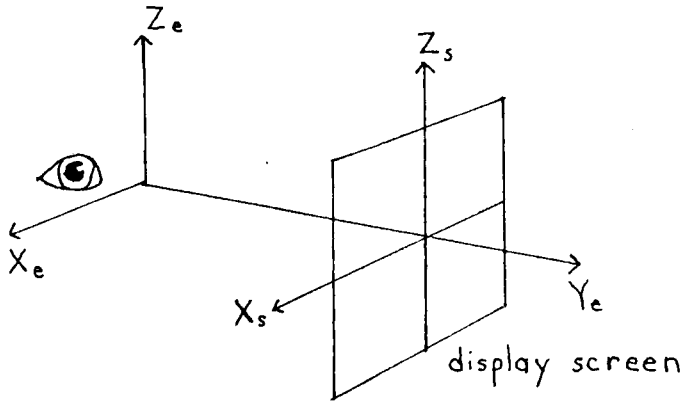


Fig. 23. Screen coordinate system.

IV. CONCLUSIONS

Previously in this thesis two goals were mentioned as the motivations behind TWODEE-THREEDEE. The first of these goals was that the system should obtain the data necessary for describing a three-dimensional object without requiring special hardware devices such as the large digitizers and picture-processing equipment used by other researchers. This goal has effectively been met in TWODEE-THREEDEE since it requires only a small tablet digitizer and display device. There was one difficulty which was caused by the use of this equipment. The problem occurred after the data had been input and the user realized that some corrections were needed in the description of the object. The more complicated the object was, the greater the number of lines involved in the description. The large quantity of lines made it a difficult and tedious process to indicate to the system which points needed correcting. This was especially true if a number of lines were concentrated in a small area, as was the case with the landscape (Figs. 15, 16 and 17, pp. 50-51). This difficulty could possibly be alleviated by the addition to the display device of either light pen capabilities or movable crosshairs. Either of these techniques would lend more accuracy than the present method for locating a point which involves repeatedly pointing to the approximate location on the tablet of the point on the screen until the system locates the desired point.

Another aid in ascertaining the correctness of the data description might have been to display the object without hidden lines. This also

could help in locating points in the description which need to be altered since not as many lines would appear on the screen to interfere with the user's perception of the object. However, it was felt that this could duplicate an important feature of MOVIE.BYU.

The second goal which motivated the creation of this system was to reduce the user's attention to the formation of the surface polygons. To some extent this goal has been met, especially when the object is relatively simple, like the ball. However, the more complicated the object is, the more attention the user must pay to the sequence of steps with which he inputs the data. This was amply demonstrated by the example of the castle. If each wall or turret had not been input as individual objects there would have been a multitude of extra polygons formed and a large amount of time would have been required to make the necessary corrections. Hence, in the present system the user must at times keep in mind how the polygons are formed as he inputs the object.

There is one particular case in which the user must exercise good planning on how to input an object. This case involves objects which have multiple vertical promontories. The scene must be divided so that each promontory is a single object. Otherwise, the tops of the promontories might be joined by a polygon when they should not be.

One topic for further research might be the development of additional heuristics for the creation of surface polygons between planes. A suggestion might be one which creates the surface polygons for the entire object at one time, after all points have been entered, instead of formulating them between each pair of slices. Such a

heuristic, having an overall view of the object, would probably eradicate the need for planning in the case mentioned above.

Another possible area of research would involve attacking the input from an application viewpoint. TWODEE-THREEDEE is biased toward engineering and scientific applications where sliced data or data which can be transformed into sliced data is more readily available. However, such data is not available to the general user who might want to apply shaded graphics as an art form. Hence, the development of an input system which could be utilized very easily by such a general user might be a topic for further study.

Finally, more research into the methods for internally describing a three-dimensional object would be appropriate. Possibly with a more suitable description of an object the difficulties encountered so far in developing input systems could be substantially alleviated.

LIST OF REFERENCES

LIST OF REFERENCES

1. Coons, S. A. Surfaces for Computer-Aided Design of Space Forms. M.I.T. Project MAC, MAC-TR-41, June 1967.

2. Forrest, A. R. Curves and Surfaces for Computer-Aided Design. Cambridge Computer-Aided Design Group Thesis, July 1968.

3. Armit, A. P. and Forrest, A. R. "Interactive Surface Design," Advanced Computer Graphics: Economics Techniques and Applications (eds.: R. D. Parslow, R. E. Green) Phenum Press, London, pp. 1179-1201, 1971.

4. Armit, A. P. "A Multipatch Design System for Coons' Patches," IEE International Conference on Computer Aided Design (Conf. Publ. 51) Southampton, U. K., April 1969, pp. 152-161.

5. Clark, James H. "Designing Surfaces in 3-D," Communications of the ACM 19, 8, 454-460 (Aug. 1976).

6. Sutherland, I. C. "A Head-Mounted Three-Dimensional Display," Proc. AFIPS 1968 Fall Joint Comput. Conf., Vol. 33, AFIPS Press, Montvale, N. J., pp. 757-764.

7. Newell, Martin E. "Man Machine Communication in Three Dimensions," Computer Aided Geometric Design (ed.: R. E. Barnhill, R. F. Riesenfeld) Academic Press, New York, 1976, pp. 303-315.

8. Braid, I. E. "The Synthesis of Solids Bounded by Many Faces," Communications of the ACM 18, 4, 209-216 (April 1975).

9. Goldstein, R. A. and Nagel, R. "3-D Visual Simulation," Simulation 16, 1, pp. 25-31 (Jan. 1971).

10. Bui Tuong Phong. "Illumination for Computer Generated Pictures," Communications of the ACM 18, 6, pp. 311-317 (June 1975).

11. Roberts, L. G. "Machine Perception of Three Dimensional Solids," Optical and Electro-Optical Information Processing (ed.: Tipper, et al.) MIT Press, 1964, p. 159.

12. Newman, W. M. and Sproull, R. F. Principles of Interactive Computer Graphics. McGraw-Hill Book Company, New York, 1973, pp. 283-332.

13. Loutrel, P. P. "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," IEEE Transactions on Computers EC-19, 3, 205, March 1970.

14. Matsushita, Y. "Hidden-line Elimination for a Rotating Object," Communications of the ACM 15, 4, 245, April 1972.

15. Galimberti, R. and Montanari, U. "An Algorithm for Hidden-Line Elimination," Communications of the ACM 12, 4, 206, April 1969.

16. Rommey, G. W., Watkins, G. S., Evans, D. C. "Real Time Display of Computer Generated Half-Tone Perspective Pictures," Proc. 1968 IFIP Congress (ed.: Morrell, A. J. H.) North Holland Pub. Co., 973, 1969.

17. Bouknight, W. J. "A Procedure for Generation of Three-dimensional Half-toned Computer Graphics Representations," Communications of the ACM 13, 9, 527, Sept. 1970.

18. Sutherland, I. E., Sproull, R. F. and Schumacker, R. A. "A Characterization of Ten Hidden-Surface Algorithms," ACM Computing Surveys, 6, 1, pp. 1-55, 1974.

19. Gouraud, Henri. "Continous Shading of Curved Surfaces," IEEE Transactions on Computer C-20, 6, pp. 623-629, June 1971.

20. Sutherland, Ivan E. "Three-Dimensional Data Input by Tablet," Proc. of the IEEE, 62, 4, pp. 453-461, April 1974.

21. Chandrasekaran, B. and Parent, R. E. "Molding Computer Clay-Steps Toward a Computer Graphics Sculptor's Studio," Pattern Recognition and Artificial Intelligence (ed.: Chen, C. H.) Academic Press, Inc., New York, 86-107, 1976.

22. Negroponte, N. "Recent Advances in Sketch Recognition," Proc. National Computer Conference, pp. 663-675, 1973.

23. Parke, Frederic I. "Measuring Three-Dimensional Surfaces With a Two-Dimensional Data Tablet," Comput. & Graphics, 1, pp. 5-7, 1975.

24. Plot-10: Graphics Tablet Utility Routines User's Manual. Information Display Division, Tektronix, Inc., Beaverton, Ore., Doc. No. 062-1666-00, 1974.

25. Plot-10: Terminal Control System User's Manual. Information Display Division, Tektronix, Inc., Beaverton, Ore., Doc. No. 062-1494-00, 1974.

26. Private communication with Richard Cloutier, U. of Tennessee.

27. Private communication with Mike Casey, U. of Tennessee.

28. Rogers, D. F. and Adams, J. A. Mathematical Elements for Computer Graphics. McGraw-Hill Book Company, New York, 1976.

29. Raisz, Erwin. General Cartography. McGraw-Hill Book Company, Inc., New York, 1948.

30. Christiansen, Hank. Technical Report, Brigham Young University, Provo, Utah, 1976.

31. Christiansen, Hank and Stephenson, Mike. MOVIE.BYU: A General Purpose Computer Graphics Display System. Dec. 1976, Brigham Young University.

APPENDICES

APPENDIX A

DESCRIPTION OF MOVIE.BYU

This appendix describes a compatible set of Fortran computer programs called MOVIE.BYU, which facilitate the display and animation of any model composed of triangles and quadrilaterals. The major portion of the system consists of a program which generates line drawings and continuous tone shaded images. However, it also includes programs which are used to organize and modify user data files, to process three-dimensional model definitions so as to expose interior surfaces, and to generate two- and three-dimensional alphanumeric characters for title purposes.

An interactive session with the display program begins with the request for three files: (1) a geometry file containing the nodal coordinates, an element connectivity array which describes how the nodes are connected into triangles and quadrilaterals, and a definition of how the scene is divided into parts, (2) a displacement file, and (3) a special function file which can contain such scalar functions as stress, temperature, or pressure. The user is then capable through key-word commands to:

1. Select from the available line drawing or continuous tone display devices,
2. Manipulate the rigid model,
3. Specify data organization and control operation of the hidden line or surface algorithm,
4. Control the display of vector and scalar functions,
5. Control the specification of color and call upon the various shading techniques,

6. Specify simple harmonic motion for the production of animated sequences,
7. Read a new set of object description files, and
8. Survey a list of available commands and exit the program.

These capabilities are made available through the following 29 commands:

Center - Automatic field definition and translation

Color - Specification of part colors and color fringe definitions

Contours - Specify parameters for contours on line drawings

Device - Selection of output device

Difuse - Specification of ambient light conditions

Distance - Specification of user distance from object

Draw - Display without hidden line or surface removal

Exit - Controlled exit from the program

Explode - Specification of local motion patterns by parts

Fast - Selection of poor man's hidden surface procedures

Field - Specification of the frustrum of vision

Flat - Invokes "flat" element shading

Fringe - Specifications related to displacement and special function fringing

Help - Types a list of available commands

Linear - Allows linear interpolation between two sets of displacement and special function files

Movie - Specification of harmonic vibration and "smooth" animation sequences

Parts - Scene content selection

Read - Read new geometry, displacement and/or special function files

Restore - Initializes rotation and translation conditions

Rotate - Specification of rotations about any translated axis

Scale - Specification of scale factors for displacements

Scope - Specification of display parameters

Smooth - Invokes "smooth" element shading (Gouraud's algorithm)

Summary - Prints summary of geometry, displacement and special function files

Translate - Shifts reference origin

Uniform - Invokes "uniform" element shading

View - Display with hidden line or surface removal (Watkin's algorithm)

Warp - Geometry modification to allow viewing of special functions

The data editing routines permit the user to start new geometry, displacement or special function files or to read them and modify the files by the deletion or addition of elements, or the modification of special values. Also, the modification routines permit the creation of symmetrical additions to the model. Another available option is to help organize the order of the nodes which define the element corners so the outward normals may be detected by the display program.

The entire MOVIE.BYU system is divided into seven files. They are:

1. USER.DOC This file contains the user documentation including helps for machine implementation on 32-bit or larger machines and 16-bit machines.

2. MOVIE.FOR This file contains the main driver and many of the subroutines for the display program.

3. HIDDEN.FOR This file contains all subroutines for the Watkin's Algorithm for hidden line and surface removal.

4. DEVICE.FOR This file contains the subroutines which access the available display devices. The program permits use of the HP Plotter, Calcomp Plotter, Tektronix Scope and Comtal Monitor.

5. UTILITY.FOR This file contains the driver and subroutines which comprise the data handling program.

6. SECTION.FOR This file contains the main driver and subroutines which comprise the program for editing hexahedron elements for three dimensional objects.

7. TITLE.FOR This file contains the main driver and subroutines for the program which generates two and three dimensional alphanumeric characters.

The present implementation of MOVIE.BYU on The University of Tennessee campus is on the DEC system-10. Only the display program (files 2, 3, and 4) has been tested and made operational. The DEVICE.FOR file has been modified so that when the Comtal monitor is requested a file is generated on the DEC system-10 for latter transfer by magnetic tape to the PDP 11/40. At present, this permits only the transfer of black and white pictures (30,31).

APPENDIX B

USER'S GUIDE AND PROGRAM LISTING

USER'S GUIDE TO TWODEE-THREEDEE

All text enclosed in < > is printed on the screen. It is assumed that the user is familiar with the DEC system-10 and its commands.

The opening sequence of questions is as follows:

< DO YOU NEED TO SEE A PICTURE OF THE MENU? >

YES → The menu will be drawn on the screen so that the user will know what it looks like. This is necessary so that he can reserve the area on the tablet for the menu.

NO → The user is familiar with the menu.

< TOUCH EACH INTERSECTION ALONG THE TOP OF THE MENU. >

The user should touch six points on the tablet in a roughly horizontal line. This is done as part of the process for reserving an area of the tablet for the menu. It is suggested that this area be in an upper portion of the usable tablet space. The six points correspond to the intersections of lines along the top of the menu. (See Fig. 10, page 35.)

< TOUCH EACH INTERSECTION ALONG THE LEFT EDGE OF THE MENU. >

The user should input the three points which correspond to the three intersections of lines along the left edge of the menu. (See Fig. 10, page 35.) The area for the menu consists of the intersection of the space bounded by vertical lines through the six points along the top of the menu with the space bounded by the horizontal lines through the three points along the left edge. These lines form a grid pattern with each block in the grid reserved for an instruction.

When the menu area has been reserved, a loud, long beep will be emitted by the Tektronix terminal. This signals that TWODEE-THREEDEE is ready for an instruction. An instruction is given by pressing the tablet pen in the block set aside for the desired instructions. In Fig. 10, page 35, the commands are those above the line in each block. Those below the line are subcommands which will be described later.

The Commands

The INPUT command is used to call all routines needed to form the data description of the object. TWODEE-THREEDEE responds to this command with:

< INPUT THE Z-COORDINATE. >

This is a request for the user to indicate the Z-coordinate for the slice which will be input next. The user supplies the Z-coordinate by touching a point on the tablet. The X-coordinate of that point will be the Z-coordinate for the slice. The object coordinate system is, therefore, as shown in Fig. 22, page 55. A short beep will be emitted immediately after the above message signaling that TWODEE-THREEDEE is ready to accept the Z-coordinate.

< INPUT THE PICTURE COORDINATES OR CHANGE THE INPUT MODE. >

At all times one of four possible input modes is in effect to indicate how the data points are to be connected in the data structure. These input modes are the four subcommands on the left side of the menu: POINT, LINE, POLYGON and EOP. Hence, immediately after the last message appears, always indicate which mode you want by touching the appropriate block in the menu.

The POINT mode indicates that all points digitized until the next change of mode will be regarded as isolated points, not connected to any other points on that slice. This mode is particularly useful when the object has a peak, such as the pinnacle of the roof in Fig. 5, page 30.

The LINE node indicates that all points digitized until the next change of mode will be connected by lines in the order they were input. For example, by entering the four points in Fig. 7 (b), page 31, and repeating the initial point, the four lines making up the rectangle will be formed in the data structure. This input mode is the one most often used, particularly in the interior slices of the object.

The POLYGON subcommand is used to form the next four or five points into surface polygons, either triangles or quadrilaterals. The first point entered must always be the last point entered. A polygon differs from lines drawn between three or four points in that the area inside the polygon will be considered a surface and will be shaded in the final shaded graphics picture.

The EOP subcommand signals that a slice is finished. At this time, TWODEE-THREEDEE forms the surface polygons between this slice and the last one. The EOP subcommand also indicates when the user is finished inputting the points for an object. This occurs when the EOP mode is the first input mode of a slice.

When the user signals that the slice is complete with an EOP command, the plan view of the slice just finished will be drawn on the screen and the following question will appear: < IS THIS CORRECT? > If the user responds YES, the surface polygons will be formed and

< INPUT THE Z-COORDINATE > will appear. (If the user is finished with the object, any Z-coordinate may be input.) If the user responds NO, the program will respond with < DO YOU WANT TO REDRAW THIS PLANE? > If the user responds YES to this question, the program will request a new input mode and the last slice will be wiped out. If the user responds NO, the program will respond with < INDICATE POINT TO BE CHANGED. > The user must then point with the pen to the approximate location on the tablet corresponding to the location on the screen of the point which needs to be moved. TWODEE-THREEDEE will enclose in a small square the point it believes the user is indicating and ask < IS THE POINT ENCLOSED WITH THE SQUARE THE ONE YOU ARE INDICATING? > If the user responds NO, it will draw a square around another point and repeat the question. This will continue until the user responds YES or all points within a small distance of the approximated point's coordinates are exhausted. Then the program will write < ***POINT NOT FOUND. TRY AGAIN.*** > After this is done, the plan view will be redrawn and user requested to verify its correctness.

At this time the message < INPUT THE Z-COORDINATE. > will reappear. At any time during the input of one slice the input mode may be changed. In fact, the same mode may be used again. For POLYGON this indicates that a new polygon is being formed. If LINE mode is used and the previous mode was LINE, then no line segment will be formed between the point input before LINE mode was reentered and the point after LINE mode was reentered.

It is possible to input multiple objects in a scene by issuing multiple INPUT commands during one input session. This capability is

especially helpful when the object being entered has several vertical promontories. If possible, one should input each peak as a single object. However, if this is not possible, input that part of the object up to the highest slice which is common to all promontories as one object, and each peak rising above the saddle point as separate objects. One should keep in mind that no surface polygons are formed between slices of different objects. Also, the user should study the heuristic given in Section III for forming the surface polygons. By understanding how it works, fewer mistakes will be made by the user's improper input of slices.

The SHOW command is used to display the object on the terminal screen with no hidden lines removed. When the SHOW command is given, all scalings and rotations which have been accumulated from the SCALE and ROTATE commands are applied to the object. When the object is displayed, the coordinate system is not the screen coordinate system used in Newman and Sproul. The screen coordinate system (shown in Fig. 23, page 55) was chosen because one normally thinks of the Z-axis as the vertical direction. Hence, it appears more logical to use the coordinate system of Fig. 23 (page 55) because a side of the object is toward the viewer; whereas, with Newman and Sproul's system, one would always view the bottom of the object. (Note: MOVIE.BYU uses the same screen coordinate system as Newman and Sproul.)

The SAVE command creates a file which contains the object description. This allows the user to utilize the capabilities of MOVIE.BYU on the data description of the object formed using TWODEE-THREEDEE. The program asks the user to < INPUT A TEN-CHARACTER OR LESS

WORD TO SERVE AS FILE NAME. > This must be a unique name that has not already been used.

The LIST command will write on the screen the names of all files which have already been saved. This function is available in order to meet the requirements of the SAVE function.

The HELP command prints a list of all commands and subcommands available with TWODEE-THREEDEE along with a brief description. The main purpose for this instruction is to list the commands, not to explain them.

The ROTATE command allows the user to rotate the object about any axis. The program will respond to this command with < AXIS, ANGLE IN DEGREES > (A1,F3.0). The user should respond with either X, Y or Z and the number of degrees to be rotated using the FORTRAN formats shown in the message. This rotation and all other rotations or scalings accumulated since the object was input or the last RECALL command will be applied to the object when the SHOW command is used.

The SCALE command applies overall scaling to the object. TWODEE-THREEDEE responds to this command with < INPUT THE DESIRED SCALE FACTOR > F3.0. The user should input a number using the supplied FORTRAN format, with any number less than one reducing the size of the object and any number greater than one enlarging it.

The CHANGE command permits certain limited changes in the data description if the user is not satisfied. These changes are requested by using the subcommands DELETE, ADD and MOVE on the right side of the menu. TWODEE-THREEDEE responds to the CHANGE command with a short beep signaling it is ready for a subcommand. Note that with each of the

subcommands if a point is to be indicated to the system, the process discussed under the INPUT command is used.

The DELETE subcommand allows the user to remove polygons from the object description in pairs. This is accomplished by indicating the endpoints of a line segment. The two polygons of which that segment is a part are removed from the data description.

The ADD subcommand permits the addition of a triangular polygon to the object description. The program asks the user to < INDICATE THE POINTS IN THE POLYGON. > The user indicates each of the three points by the method described under the INPUT command.

The MOVE instruction permits the user to change the location of a point. When the program asks < WHICH POINT? >, the user indicates one as explained before. Then the user in response to < SHOW NEW POSITION. > should touch the pen to a location on the tablet corresponding to the desired location for the point on the screen. Note that only the X and Y coordinates of the point will be changed.

Any instance of the CHANGE command permits only one subcommand. Hence, after the subcommand is completed, the next beep will be a request for a command.

The STOP command causes an orderly exiting of the program.

The RECALL command has two uses. If the input command has not been executed, the data description recalled is that of the object created the last time the program was run. This feature permits the user to look at his object with hidden lines removed under MOVIE.BYU and then return to TWODEE-THREEDEE to make changes if necessary. If the INPUT

command has been used, the data description recalled will be that object just input. At the same time all accumulated rotations and scalings will be wiped out.

Creation of the Shaded Picture

In order to create a shaded picture, several things must be done. The description must be saved by TWODEE-THREEDEE. When MOVIE.BYU is executed (EX MOVØØ2.FOR,MOVØØ3.FOR,MOVØØ4.FOR,/SEARCH FOR: TCS), the device must be COMT and the VIEW command used. This creates a description file of the shaded picture called PICTUR.PIC. Next, the user should execute SETUP.FOR and PACK.MAC. These programs create a batch job called PICTUR.CTL which should be submitted in order to transfer the shaded picture file to tape. To run SETUP.FOR and PACK.MAC, always answer PICTUR as the file to be transferred and L as the size of the picture (512 × 512 pixels). The tape number depends on which tape you have given to the computing center.

The tape should be returned to the Image and Pattern Analysis Laboratory where it is mounted on the PDP 11/40 for display using DSP3.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THE SUBROUTINE HOMOG TAKES THE THREE-DIMENSIONAL COORDI-
C       NATES OF A PICTURE AND ADDS A FOURTH DIMENSION TO FORM
C       HOMOGENEOUS COORDINATES.  THE VALUE OF THE NEW COORDI-
C       NATE IS 1 IN EACH CASE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE HOMOG(PICTUR,NOPTS)
        REAL PICTUR(300,4)
        DO 5 I=1,NOPTS
5         PICTUR(I,4)=1.0
        RETURN
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THE SUBROUTINE MAMULT MULTIPLIES MATRX1 BY MATRX2 AND
C       STORES THE RESULT IN MATRX1.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE MAMULT(MATRX1,MATRX2,NOPTS)
        REAL MATRX2(4,4),ARRAY(4),MATRX1(NOPTS,4)
        DO 15 I=1,NOPTS
          DO  5 M=1,4
            ARRAY(M)=0.0
            DO 5 J=1,4
             IF (MATRX1(I,J) .EQ. 0.0 .OR. MATRX2(J,M) .EQ. 0.0)
1               GO TO 5
              ARRAY(M)=ARRAY(M)+MATRX1(I,J)*MATRX2(J,M)
5           CONTINUE
            DO 10 J=1,4
10            MATRX1(I,J)=ARRAY(J)
15      CONTINUE
20      RETURN
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THE SUBROUTINE ROTATE CREATES A MATRIX WHICH CAUSES A
C       ROTATION ABOUT ONE OF THE AXES.  THE ANGLE OF THE ROTA-
C       TION CAN EITHER BE SUPPLIED TO THE SUBROUTINE OR IF RAD
C       (THE ANGLE IN RADIANS) IS 0.0, A SUBROUTINE SINCOS MAY
C       BE CALLED TO DETERMINE THE SIN AND COS OF THE
C       ANGLE FROM TWO GIVEN POINTS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE ROTATE(PICTUR,IAXIS,POINT1,POINT2,NOPTS,MFLAG
1   ,RAD)
C
C       SUBROUTINES CALLED:
C         SINCOS-DETERMINE SINE AND COSINE OF ANGLE
C         IDENT-FORM IDENTITY MATRIX
C         MAMULT-MULTIPLIES TWO MATRICES
C         MOVE-TRANSFERS ONE MATRIX INTO ANOTHER
C
C       VARIABLES USED:
C         ROTMAT-ROTATION MATRIX
C         PICTUR-POINTS FORMING OBJECT
C         IAXIS-AXIS OF ROTATION
C         NOPTS-NUMBER OF POINTS IN PICTUR
```

```
06200   C          MFLAG- 1=> MULTIPLYING ROTMAT AND PICTUR
06300   C               - 0=> MOVING ROTMAT INTO PICTUR
06400   C          COSTHE-COSINE OF ANGLE OF ROTATION
06500   C          SINTHE-SINE OF ANGLE OF ROTATION
06600   C          POINT1,POINT2-POINTS WHICH DETERMINE ANGLE
06700   C          RAD-ANGLE IN RADIANS
06800            DIMENSION ROTMAT(4,4),PICTUR(NOPTS,4),POINT1(3),POINT2(3
06900           1  )
07000            IF (RAD .EQ. 0.0) CALL SINCOS(POINT1,POINT2,SINTHE,COSTH
07100           1E,IAXIS)
07200            IF (RAD .EQ. 0.0) GO TO 5
07300            COSTHE=COS(RAD)
07400            SINTHE=SIN(RAD)
07500   5        CALL IDENT(ROTMAT)
07600            DO 10 I=1,3
07700              IF (I .EQ. IAXIS) GO TO 10
07800              ROTMAT(I,I)=COSTHE
07900   10         CONTINUE
08000            IF (IAXIS .NE. 1) GO TO 15
08100            ROTMAT(2,3)=-SINTHE
08200            ROTMAT(3,2)=SINTHE
08300   15       IF (IAXIS .NE. 2) GO TO 20
08400            ROTMAT(1,3)=SINTHE
08500            ROTMAT(3,1)=-SINTHE
08600   20       IF (IAXIS .NE. 3) GO TO 25
08700            ROTMAT(1,2)=-SINTHE
08800            ROTMAT(2,1)=SINTHE
08900   25       IF (MFLAG .EQ. 1)CALL MAMULT(PICTUR,ROTMAT,NOPTS)
09000            IF (MFLAG .EQ. 0) CALL MOVE(PICTUR,ROTMAT,NOPTS)
09100            RETURN
09200            END
09300
09400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
09500   C
09600   C        SINCOS DETERMINES THE SINE AND COSINE OF THE ANGLE OF
09700   C        ROTATION IF THE ROTATION IS ABOUT EITHER THE X OR Y AXIS
09800   C        .  THESE TRIGONOMETRIC VALUES ARE DETERMINED BY:
09900   C        SIN(THETA)= LENGTH OF OPPOSITE SIDE/LENGTH OF HYPOTENUSE
10000   C        COS(THETA)= LENGTH OF ADJACENT SIDE/LENGTH OF HYPOTENUSE
10100   C
10200   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
10300            SUBROUTINE SINCOS (POINT1,POINT2,SINTHE,COSTHE,IAXIS)
10400            DIMENSION POINT1(3),POINT2(3)
10500            REAL Z1(3),Z2(3)
10600            DIST(X,Y)=SQRT(X**2+Y**2)
10700            DIFF(X,Y)=X-Y
10800            DO 25 I=1,3
10900              Z1(I)=POINT1(I)
11000              Z2(I)=POINT2(I)
11100   25       CONTINUE
11200            DISBET=DIST(DIFF(Z1(1),Z2(1)),DIFF(Z1(2),Z2(2)))
11300            IF (IAXIS-2) 5,10,15
11400   15       COSTHE=-DIFF(Z1(2),Z2(2))/DISBET
11500            SINTHE=DIFF(Z1(1),Z2(1))/DISBET
11600            RETURN
11700   10       SINTHE=DIFF(Z1(1),Z2(1))/DISBET
11800            COSTHE=DIFF(Z1(2),Z2(2))/DISBET
11900            RETURN
12000   5        DISTBE=DIST(DIFF(Z1(3),Z2(3)),DISBET)
12100            SINTHE=-DIFF(Z1(3),Z2(3))/DISTBE
12200            COSTHE=DISBET/DISTBE
```

```
12300           RETURN
12400           END
12500
12600   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
12700   C
12800   C       THE SUBROUTINE TRANSL CREATES A MATRIX FOR TRANSLATING
12900   C       THE COORDINATE SYSTEM SO THAT POINT MAPS INTO POINT2.
13000   C
13100   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
13200           SUBROUTINE TRANSL (PICTUR,POINT,NOPTS,MFLAG,POINT2)
13300   C
13400   C       SUBROUTINES CALLED:
13500   C         IDENT-FORM IDENTITY MATRIX
13600   C         MATMULT-MULTIPLIES TWO MATRICES
13700   C         MOVE-TRANSFERS ONE MATRIX INTO ANOTHER
13800   C       VARIABLES USED:
13900   C         PICTUR-MATRIX CONTAINING OBJECT COORDINATES
14000   C         POINT-COORDINATES OF NEW ORIGIN
14100   C         NOPTS-NUMBER OF POINTS IN PICTUR
14200   C         MFLAG-SAME AS IN ROTATE
14300   C         POINT2-OLD ORIGIN
14400   C
14500           REAL MATRIX(4,4),PICTUR(NOPTS,4),POINT(3),POINT2(3)
14600           CALL IDENT(MATRIX)
14700           DO 5 I=1,3
14800   5         MATRIX(4,I)=POINT2(I)-POINT(I)
14900           IF (MFLAG .EQ. 1) CALL MAMULT(PICTUR,MATRIX,NOPTS)
15000           IF (MFLAG .EQ. 0) CALL MOVE(PICTUR,MATRIX,NOPTS)
15100           RETURN
15200           END
15300
15400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
15500   C
15600   C       THE SUBROUTINE IDENT CREATES THE IDENTITY MATRIX.
15700   C
15800   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
15900           SUBROUTINE IDENT(MATRIX)
16000           REAL MATRIX(4,4)
16100           DO 5 I=1,4
16200             DO 5  J=1,4
16300               MATRIX(I,J)=0.0
16400               IF (I .EQ. J) MATRIX(I,J)=1.0
16500   5       CONTINUE
16600           RETURN
16700           END
16800
16900   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
17000   C
17100   C       THIS SECTION CREATES THE VIEWING TRANSFORMATION AND
17200   C       CHANGES THE OBJECT COORDINATES TO EYE COORDINATES.  IN
17300   C       THE EYE COORDINATE SYSTEM THE ORIGIN IS LOCATED AT THE
17400   C       VIEWPOINT AND THE Z-AXIS IS DIRECTED TOWARDS A SECOND
17500   C       POINT.
17600   C
17700   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
17800           SUBROUTINE SHOW(POINT,POINT2,NOLINE,NOPTS,VNORML)
17900   C
18000   C       SUBROUTINES CALLED:
18100   C         HOMOG-CREATES FOURTH COORDINATE
18200   C         TRANSL-TRANSLATES ORIGIN
18300   C         ROTATE-CAUSES ROTATION
```

```
18400   C          CLIPCO-TRANSFORMS INTO CLIPPING COORDINATE SYSTEM
18500   C          MAMULT-MULTIPLIES TWO MATRICES
18600   C          CLIP-REMOVES LINES OUTSIDE THE SCREEN
18700   C          PRSPEC-PERFORMS PERSPECTIVE TRANSFORMATION
18800   C          DISPLA-DRAWS OBJECT ON SCREEN
18900   C
19000   C        VARIABLES USED:
19100   C          PICTUR-MATRIX CONTAINING OBJECT COORDINATES
19200   C          LINES-INDICATES WHICH POINTS FORM LINES AND WHICH
19300   C                POLYGONS THE LINES ARE IN
19400   C          MATRIX-TRANSFORMATION MATRIX
19500   C          NOLINE-NUMBER OF LINES
19600   C
19700           REAL POINT(3),POINT2(3),MATRIX(4,4),PICTUR(300,4),POINT3
19800           1  (3),MAT(4,4)
19900           COMMON /A/MATRIX /DESCRP/POLYGN,LINES,PICTUR
20000           COMMON /TRANS/ MAT
20100           INTEGER POLYGN(600,10),LINES(1000,5)
20200           DATA POINT3/3*0.0/
20300           MFLAG=0
20400           CALL TRANSL(MATRIX,POINT,4,MFLAG,POINT3)
20500           MFLAG=1
20600           CALL ROTATE(MATRIX,3,POINT,POINT2,4,MFLAG,0.0)
20700   92      CALL ROTATE(MATRIX,1,POINT,POINT2,4,MFLAG,0.0)
20800           CALL CLIPCO(MATRIX,MFLAG)
20900   100     CALL MAMULT(PICTUR,MATRIX,300)
21000           CALL CLIP(PICTUR,LINES,NOLINE,NOPTS)
21100           CALL PRSPEC(PICTUR,NOPTS,300)
21200           CALL DISPLA(LINES,NOLINE,PICTUR,1,3)
21300           RETURN
21400           END
21500
21600   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
21700   C
21800   C        THE SUBROUTINE SCALE CREATES A MATRIX WHICH CHANGES THE
21900   C        SCALE OF THE COORDINATE SYSTEM ALONG ANY ONE, OR ALL OF
22000   C        THE AXES.
22100   C
22200   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
22300           SUBROUTINE SCALE (MATRIX,SX,SY,SZ,MFLAG)
22400           REAL MATRIX(300,4),MATR(4,4)
22500           CALL IDENT(MATR)
22600           MATR(1,1)=SX
22700           MATR(2,2)=SY
22800           MATR(3,3)=SZ
22900           IF (MFLAG .EQ. 1) CALL MAMULT(MATRIX,MATR,100)
23000           IF (MFLAG .EQ. 0) CALL MOVE(MATRIX,MATR,4)
23100           RETURN
23200           END
23300
23400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
23500   C
23600   C        THE SUBROUTINE  MOVE PUTS ONE MATRIX IN ANOTHER.
23700   C
23800   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
23900           SUBROUTINE MOVE (MATRIX,MATR,NOPTS)
24000           REAL MATRIX(NOPTS,4),MATR(4,4)
24100           DO 5 I=1,NOPTS
24200             DO 5 J=1,4
24300   5           MATRIX(I,J)=MATR(I,J)
24400           RETURN
```

```
24500           END
24600
24700   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
24800   C
24900   C       THE SUBROUTINE CLIP DELETES THAT PORTION OF THE PICTURE
25000   C       WHICH WILL NOT FIT IN THE VIEWING PYRAMID BECAUSE IT IS
25100   C       EITHER BEHIND THE VIEWPOINT OR OUTSIDE THE FIELD OF
25200   C       VISION.
25300   C
25400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
25500           SUBROUTINE CLIP(POINT,LINES,NOLINE,NOPTS)
25600   C
25700   C       SUBROUTINES CALLED:
25800   C         CODE-ESTABLISHES A CODE FOR EACH LINE
25900   C         OR-LOGICAL OR OVER FOUR LOGICAL CONSTANTS
26000   C       VARIABLES USED:
26100   C         POINT-MATRIX CONTAINING OBJECT COORDINATES
26200   C         KODE-ARRAY CONTAINING CODE FOR BOTH ENDPOINTS OF A
26300   C              LINE
26400   C         PTNO1-FIRST ENDPOINT
26500   C         PTNO2-OTHER ENDPOINT
26600   C         ITEMP-USED TO EXCHANGE POINTS
26700   C         LTEMP-USED TO EXCHANGE CODES
26800   C         T-FACTOR USED TO MOVE ENDPOINTS
26900   C
27000           LOGICAL KODE(3,4),LOR,LTEMP
27100           REAL POINT(300,4)
27200           INTEGER PTNO1, PTNO2,LINES(1000,5)
27300           LOR(I)=KODE(I,1).OR.KODE(I,2).OR.KODE(I,3).OR.KODE(I,4)
27400           DO 5 I=1,NOLINE
27500   C
27600   C       CALCULATE CODES FOR BOTH ENDPOINTS OF THE LINE
27700   C
27800             LINES(I,5)=0
27900             PTNO1=LINES(I,3)
28000             PTNO2=LINES(I,4)
28100   7         CALL CODE(KODE,PTNO1,PTNO2,POINT,NOPTS)
28200   C
28300   C       IF BOTH CODES ARE ZERO, THE LINE IS COMPLETELY VISIBLE
28400   C       GO TO NEXT LINE.
28500   C
28600             IF ((.NOT. LOR(1)) .AND. (.NOT. LOR(2))) GO TO 5
28700   C
28800   C       COMPUTE THE LOGICAL AND OF THE CODES.  IF IT IS NOT
28900   C       ZERO, THEN NO PART OF THE LINE IS VISIBLE--SET A FLAG
29000   C       ON THE LINE SO AS NOT TO DISPLAY IT.
29100   C
29200             DO 10 J=1,4
29300   10          KODE(3,J)=KODE(1,J) .AND. KODE(2,J)
29400             IF (.NOT. LOR(3)) GO TO 15
29500             LINES(I,5)=1
29600             GO TO 5
29700   C
29800   C       IF THE FIRST POINT IS ON THE SCREEN (CODE EQUALS
29900   C       ZERO), EXCHANGE POINTS 1 AND 2.
30000   C
30100   15        IF (.NOT. LOR(1)) GO TO 20
30200             ITEMP=PTNO1
30300             PTNO1=PTNO2
30400             PTNO2=ITEMP
30500             DO 25 J=1,4
```

```
30600                 LTEMP=KODE(1,J)
30700                 KODE(1,J)=KODE(2,J)
30800    25         KODE(2,J)=LTEMP
30900    C
31000    C         PUSH THE FIRST POINT TOWARD THE SECOND POINT UNTIL IT
31100    C         INTERSECTS A LEGITIMATE BOUNDARY OF THE VIEWING PYRAMID.
31200    C         IF KODE(1,4) IS TRUE, PUSH TOWARD THE BOUNDARY X=-Y.
31300    C
31400    20         IF (.NOT. KODE(1,4)) GO TO 30
31500                T=(POINT(PTNO1,2)+POINT(PTNO1,1))/((POINT(PTNO1,1)-
31600               1    POINT(PTNO2,1))-(POINT(PTNO2,2)-POINT(PTNO1,2)))
31700                POINT(PTNO1,2)=T*(POINT(PTNO2,2)-POINT(PTNO1,2))+
31800               1   POINT(PTNO1,2)
31900                POINT(PTNO1,1)=-POINT(PTNO1,2)
32000                POINT(PTNO1,3)=T*(POINT(PTNO2,3)-POINT(PTNO1,3))+
32100               1   POINT(PTNO1,3)
32200                GO TO 7
32300    C
32400    C         IF KODE(1,3) IS TRUE, PUSH TOWARD THE BOUNDARY X=Y
32500    C
32600    30           IF (.NOT. KODE(1,3)) GO TO 35
32700                T=(POINT(PTNO1,2)-POINT(PTNO1,1))/((POINT(PTNO2,1)-
32800               1   POINT(PTNO1,1))-(POINT(PTNO2,2)-POINT(PTNO1,2)))
32900                POINT(PTNO1,2)=T*(POINT(PTNO2,2)-POINT(PTNO1,2))+
33000               1   POINT(PTNO1,2)
33100                POINT(PTNO1,1)=POINT(PTNO1,2)
33200                POINT(PTNO1,3)=T*(POINT(PTNO2,3)-POINT(PTNO1,3))+
33300               1   POINT(PTNO1,3)
33400                GO TO 7
33500    C
33600    C         IF KODE(1,2) IS TRUE, PUSH TOWARD THE BOUNDARY Z=-Y.
33700    C
33800    35           IF (.NOT. KODE(1,2)) GO TO 40
33900                T=(POINT(PTNO1,2)+POINT(PTNO1,3))/((POINT(PTNO1,3)-
34000               1   POINT(PTNO2,3))-(POINT(PTNO2,2)-POINT(PTNO1,2)))
34100                POINT(PTNO1,2)=T*(POINT(PTNO2,2)-POINT(PTNO1,2))+
34200               1   POINT(PTNO1,2)
34300                POINT(PTNO1,1)=T*(POINT(PTNO2,1)-POINT(PTNO1,1))+
34400               1   POINT(PTNO1,1)
34500                POINT(PTNO1,3)=-POINT(PTNO1,2)
34600                GO TO 7
34700    C
34800    C         IF KODE(1,1) IS TRUE, PUSH TOWARD THE BOUNDARY Z=Y.
34900    C
35000    40         T=(POINT(PTNO1,2)-POINT(PTNO1,3))/((POINT(PTNO2,3)-
35100               1   POINT(PTNO1,3))-(POINT(PTNO2,2)-POINT(PTNO1,2)))
35200                POINT(PTNO1,2)=T*(POINT(PTNO2,2)-POINT(PTNO1,2))+
35300               1   POINT(PTNO1,2)
35400                POINT(PTNO1,1)=T*(POINT(PTNO2,1)-POINT(PTNO1,1))+
35500               1   POINT(PTNO1,1)
35600                POINT(PTNO1,3)=POINT(PTNO1,2)
35700                GO TO 7
35800    5           CONTINUE
35900              RETURN
36000              END
36100
36200    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
36300    C
36400    C         CLIPCO TRANSFORMS THE POINTS FROM THE EYE COORDINATE
36500    C         SYSTEM TO THE CLIPPING COORDINATE SYSTEM.
36600    C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CLIPCO(MATRIX,MFLAG)
      REAL MATRIX(4,4),MAT(4,4)
      DATA SX/2.0/,SZ/2.5/
C
C     CREATE THE TRANSFORMATION MATRIX, MAT.
C
      CALL IDENT(MAT)
      MAT(1,1)=SX
      MAT(3,3)=SZ
      IF (MFLAG .EQ. 1) CALL MAMULT(MATRIX,MAT,4)
      RETURN
      END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THE SUBROUTINE CODE ESTABLISHES A CODE FOR EACH END-
C     POINT OF THE LINE WHICH THE SUBROUTINE CLIP IS WORKING
C     ON.  THE CODE FOR THE FIRST ENDPOINT IS IN THE FIRST ROW
C     OF THE ARRAY KODE AND THE CODE FOR THE SECOND END-
C     POINT IS IN THE SECOND ROW.  THE THIRD ROW IS USED FOR
C     THE LOGICAL AND OF THE FIRST TWO ROWS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CODE(KODE,NO1,NO2,POINTS,NOPTS)
C
C     VARIABLES USED:
C       KODE-CONTAINS CODE OF BOTH LINE ENDPOINTS
C       NO1-FIRST ENDPOINT
C       NO2-OTHER ENDPOINT
C       POINTS-MATRIX CONTAINING OBJECT COORDINATES
C       NOPTS-NUMBER OF POINTS
C       NO-EITHER NO1 OR NO2
      LOGICAL KODE(3,4)
      REAL POINTS(300,4)
      NO = NO1
      DO 5 I=1,4
        DO 5 J=1,3
5         KODE(J,I)=.FALSE.
      DO 10 I=1,2
C
C     THE CODES FOR EACH POINT ARE ESTABLISHED IN THE FOLLOW-
C     ING WAY:
C        IF Z > Y, THEN CODE=1000
C        IF Z < -Y, THEN CODE=0100
C        IF X > Y, THEN CODE=0010
C        IF X < -Y,THEN CODE=0001
C     WHERE X, Y, Z ARE THE X, Y, Z COORDINATES FOR THE
C     POINT AND 1=>TRUE AND 0=>FALSE.
C
        IF (POINTS(NO,3) .GT. POINTS(NO,2)) KODE(I,1)=.TRUE.
        IF (POINTS(NO,3) .LT. -POINTS(NO,2)) KODE(I,2)=.TRUE.
        IF (POINTS(NO,1) .GT. POINTS(NO,2)) KODE(I,3)=.TRUE.
        IF (POINTS(NO,1) .LT. -POINTS(NO,2)) KODE(I,4)=.TRUE.
        NO=NO2
10    CONTINUE
      RETURN
      END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
C       THE SUBROUTINE PRSPEC CREATES A PERSPECTIVE TRANS-
C       FORMATION AND MULTIPLIES IT TIMES THE PICTURE ELEMENTS.
C       THIS PREPARES THE PICTURE FOR DISPLAY.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE PRSPEC(POINTS,NOPTS,NUM)
C
C       SUBROUTINES CALLED:
C         MAMULT-MULTAPLIES TWO MATRICES
C       VARIABLES USED:
C         POINTS-COORDINATES OF OBJECT POINTS
C         MATRIX-PERSPECTIVE TRANSFORMATION MATRIX
C         VSX-WIDTH OF SCREEN ALONG X AXIS
C         VCX-CENTER OF SCREEN ALONG X AXIS
C         VSZ-WIDTH OF SCREEN ALONG Z AXIS
C         VCZ-CENTER OF SCREEN ALONG Z AXIS
C
        REAL MATRIX(4,4),POINTS(NUM,4)
        DATA VSX,VCX,VCZ,VSZ/511.5,511.5,390.,390./
C
C       CREATE THE MATRIX
C
        DO 5 I=1,4
          DO 5 J=1,4
5       MATRIX(I,J)=0.0
        MATRIX(1,1)=VSX
        MATRIX(2,1)=VCX
        MATRIX(4,2)=-1.0
        MATRIX(2,3)=VCZ
        MATRIX(3,3)=VSZ
        MATRIX(2,4)=1.0
C
C       MULTIPLY THE MATRICES
C
        CALL MAMULT(POINTS,MATRIX,NUM)
C
C       RETURN TO NORMAL HOMOGENEOUS COORDINATES.
C
        DO 10 I=1,NOPTS
          DO 10 J=1,4
10          POINTS(I,J)=POINTS(I,J)/POINTS(I,4)
        RETURN
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DRAW ALL LINES WHICH HAVE NOT BEEN DELETED BY CLIP.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE DISPLA(LINES,NOLINE,POINTS,NUM,SUB)
C
C       VARIABLES USED:
C         LINES-WHICH POINTS FORM LINES
C         NUM-FIRST LINE TO BE DISPLAYED
C         SUB-EITHER Z OR Y COORDINATE IS USED
C         NOLINE-NUMBER OF LINES PRESENT
C
        INTEGER PTNO1,PTNO2,SUB,LINES(1000,5)
        REAL POINTS(300,4)
        CALL NEWPAG
        CALL DWINDO(0.0,1023.,0.0,780.)
```

```
48900    1        DO 5 I=NUM,NOLINE
49000               IF (LINES(I,5) .EQ. 1.0) GO TO 5
49100               PTNO1=LINES(I,3)
49200               PTNO2=LINES(I,4)
49300               CALL MOVEA(POINTS(PTNO1,1),POINTS(PTNO1,SUB))
49400               CALL DRAWA(POINTS(PTNO2,1),POINTS(PTNO2,SUB))
49500    5        CONTINUE
49600             RETURN
49700             END
```

```
00100   CCCC
```

```
06200           CALL RESET
06300           L=I
06400           IF (ANSWER .EQ. 'Y') GO TO 100
06500           L=L+1
06600           GO TO 15
06700   C
06800   C       HAVE THE USER TRY AGAIN IF HE WASN'T CLOSE ENOUGH.
06900   C
07000   20      CALL ANMODE
07100           CALL LINEF
07200           WRITE(5,35)
07300   35      FORMAT(' ***POINT NOT FOUND. TRY AGAIN.***')
07400           FOUND=0
07500           CALL RESET
07600   100     RETURN
07700           END
07800
07900   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
08000   C
08100   C       THE SUBROUTINE INSTR ALLOWS THE USER TO POINT TO A
08200   C       LOCATION ON A MENU AND WILL THEN DETERMINE THE CODE
08300   C       WHICH CORRESPONDS TO THAT LOCATION.  WHEN THIS
08400   C       CODE IS RETURNED TO THE MAIN PROGRAM, AN INSTRUCTION
08500   C       CORRESPONDING TO THAT CODE WILL BE EXECUTED.
08600   C
08700   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
08800           SUBROUTINE INSTR(ORDER,FOUND,BIT,IX,IY)
08900   C
09000   C       VARIABLES USED:
09100   C         LOCX-X COORDINATES OF MENU GRID
09200   C         LOCY-Y COORDINATES OF MENU GRID
09300   C         INST-INSTRUCTIONS
09400   C         ORDER-INSTRUCTION CHOSEN FROM MENU
09500   C         FOUND- 0=> INVALID INSTRUCTION
09600   C                1=> VALID INSTRUCTION
09700   C         BIT- 0=> CALL FROM NPLANE
09800   C              1=> CALL FROM MAIN PROGRAM
09900   C         IX-X COORDINATE OF INSTRUCTION POINT
10000   C         IY-Y COORDINATE OF INSTRUCTION POINT
10100   C
10200           INTEGER LOCX(10),LOCY(10),INST(2,6),ORDER,FOUND
10300           COMMON /MENUIN/LOCX,LOCY,N,M
10400           DATA INST/1,2,3,4,5,6,7,8,9,10,11,12/
10500   C
10600   C       INITIALIZE THE TABLET AND GET THE LOCATION IN THE
10700   C       MENU INDICATED BY THE USER.
10800   C
10900           IF (BIT .EQ. 0.00) GO TO 40
11000   20      CALL ONEPNT(IX,IY)
11100           CALL ANMODE
11200           CALL RECOVR
11300   C
11400   C       DETERMINE IN WHICH OF THE REGIONS ALONG THE X-AXIS
11500   C       OF THE MENU THE POINT IS LOCATED.
11600   C
11700   40      DO 5 I=1,N
11800             IF(IX .GE. LOCX(I) .AND. IX .LE. LOCX(I+1)) GO TO 15
11900   5         CONTINUE
12000   C
12100   C       IF THE POINT IS NOT IN ONE OF THE REGIONS OF THE MENU,
12200   C       TELL THE USER TO INPUT A NEW LOCATION.
```

```
12300   C
12400   35      FOUND=0
12500           IF (BIT .EQ. 0) GO TO 100
12600           CALL ANMODE
12700           WRITE(5,10)
12800   10      FORMAT(' **INVALID INSTRUCTION--TRY AGAIN.**')
12900           CALL RECOVR
13000           GO TO 20
13100   C
13200   C       DETERMINE IN WHICH REGION OF THE MENU ALONG THE Y-AXIS
13300   C       THE POINT IS LOCATED.
13400   C
13500   15      DO 25 J=1,M
13600             IF (IY .LE. LOCY(J) .AND. IY .GE. LOCY(J+1)) GO TO 30
13700   25        CONTINUE
13800           GO TO 35
13900   C
14000   C       THE CODE FOR THE INSTRUCTION IS THEN LOCATED IN THE
14100   C       ARRAY INST IN THE SAME LOCATION AS THE BLOCK FOR THAT
14200   C       INSTRUCTION IN THE MENU.
14300   C
14400   30      ORDER=INST(J,I)
14500           FOUND=1
14600   100     RETURN
14700           END
14800
14900   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
15000   C
15100   C       THE SUBROUTINE MENU CREATES THE ARRAYS, LOCX & LOCY,
15200   C       WHICH CONTAIN THE LOCATIONS OF THE INTERSECTIONS OF
15300   C       THE GRID WHICH FORMS THE MENU.
15400   C
15500   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
15600           SUBROUTINE MENU
15700   C
15800   C       VARIABLES USED:
15900   C        I-NUMBER OF X COORDINATES IN MENU GRID
16000   C        J-NUMBER OF Y COORDINATES IN MENU GRID
16100   C        IH-CODE FOR POSITION OF PEN
16200   C
16300           INTEGER LOCX(10),LOCY(10),I,J,IH,IY
16400           COMMON /MENUIN/LOCX,LOCY,I,J
16500   C
16600   C       HAVE THE USER TOUCH ALL THE INTERSECTIONS ALONG
16700   C       THE TOP OF THE MENU GRID.
16800   C
16900           CALL TAHINT(0,0,0)
17000           CALL ANMODE
17100           WRITE(5,1)
17200   1       FORMAT(' <DO YOU NEED TO SEE A PICTURE OF THE MENU?>')
17300           READ(5,2) IANS
17400   2       FORMAT(A1)
17500           IF (IANS .EQ. 'N') GO TO 4
17600           WRITE(5,35)
17700           WRITE(5,40)
17800           WRITE(5,40)
17900           WRITE(5,50)
18000           WRITE(5,40)
18100           WRITE(5,55)
18200           WRITE(5,40)
18300           WRITE(5,65)
```

```
18400            WRITE(5,40)
18500            WRITE(5,35)
18600            WRITE(5,40)
18700            WRITE(5,40)
18800            WRITE(5,60)
18900            WRITE(5,40)
19000            WRITE(5,55)
19100            WRITE(5,40)
19200            WRITE(5,70)
19300            WRITE(5,40)
19400            WRITE(5,35)
19500    4       WRITE(5,5)
19600    5       FORMAT(' <TOUCH EACH INTERSECTION ALONG THE TOP OF THE '
19700            1 ,'MENU.>')
19800            CALL RECOVR
19900            DO 25 I=1,6
20000              CALL ONEPNT(LOCX(I),IY)
20100    25      CONTINUE
20200            I=I-2
20300    C
20400    C       HAVE THE USER INPUT THE INTERSECTIONS ALONG THE
20500    C       LEFT SIDE OF THE MENU GRID.
20600    C
20700            CALL LINEF
20800            CALL LINEF
20900            CALL ANMODE
21000            WRITE(5,15)
21100    15      FORMAT(' <TOUCH EACH INTERSECTION ALONG THE LEFT EDGE',
21200            1 ' OF THE MENU.>')
21300            CALL RECOVR
21400            DO 30 J=1,3
21500              CALL ONEPNT(IX,LOCY(J))
21600    30      CONTINUE
21700            J=J-2
21800    35      FORMAT(' ','X',5(9('*'),'X'))
21900    40      FORMAT(' ','*',5(9X,'*'))
22000    50      FORMAT('+',3X,'INPUT',5X,'SAVE',6X,'LIST',6X,'SCALE',5X,
22100            1  'STOP')
22200    55      FORMAT('+',3X,5('-----',5X))
22300    60      FORMAT('+',3X,'SHOW',6X,'HELP',5X,'ROTATE',4X,'CHANGE',
22400            1  4X,'RECALL')
22500    65      FORMAT('+',3X,'POINT',4X,'POLYGON',25X,'ADD')
22600    70      FORMAT('+',3X,'LINE',6X,'EDP',16X,'DELETE',4X,'MOVE')
22700            RETURN
22800            END
22900
23000    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
23100    C
23200    C       THE SUBROUTINE SAVE ALLOWS THE USER TO SAVE ANY
23300    C       PICTURE WITH WHICH HE IS SATISFIED.
23400    C
23500    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
23600            SUBROUTINE SAVE
23700            INTEGER PTNUM,POLCNT,POLYGN(600,10),LINES(1000,5),NPLT(20
23800            1  ,2)
23900            DOUBLE PRECISION FILEN(30)
24000            REAL POINTS(300,4)
24100            COMMON /STORAG/FILEN,MANY/DESCRP/POLYGN,LINES,POINTS
24200            1  /PNTRS/PTNUM,LINCNT,POLCNT,ZMID,NP /PARTS/ NPLT
24300    C
24400    C       RETRIEVE THE LIST OF FILE NAMES.
```

```
24500   C
24600           CALL NEWPAG
24700           CALL GET
24800           MANY=MANY+1
24900   C
25000   C       HAVE THE USER INPUT THE NAME OF THE FILE UNDER WHICH
25100   C       THE PICTURE WILL BE STORED.
25200   C
25300           CALL ANMODE
25400           WRITE(5,5)
25500   5       FORMAT(' <INPUT A TEN-CHARACTER OR LESS WORD TO SERVE',
25600           1  ' AS A FILE NAME.>')
25700           READ(5,10) FILEN(MANY)
25800   10      FORMAT(A10)
25900           CALL RESET
26000   C
26100   C       STORE THE PICTURE DESCRIPTION AND RETURN THE LIST OF
26200   C       FILE NAMES TO STORAGE.
26300   C
26400           OPEN(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
26500           NEND=PTNUM-1
26600           READ(21,65)((POINTS(I,J),J=1,3),I=1,NEND)
26700   65      FORMAT(//(6E12.5))
26800           CLOSE(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
26900           OPEN(UNIT=23,ACCESS='SEQOUT',FILE=FILEN(MANY))
27000           WRITE(23,15) NP,NEND,POLCNT
27100           WRITE(23,15) ((NPLT(I,J),J=1,2),I=1,NP)
27200           WRITE(23,20) ((POINTS(I,J),J=1,3),I=1,NEND)
27300           WRITE(23,15) ((POLYGN(I,J),J=1,4),I=1,POLCNT)
27400   15      FORMAT(20I4)
27500   20      FORMAT(6E12.5)
27600           CLOSE(UNIT=23,ACCESS='SEQOUT',FILE=FILEN(MANY))
27700           OPEN(UNIT=22,DEVICE='DSK',ACCESS='SEQOUT')
27800           WRITE(22,25) MANY,(FILEN(I),I=1,MANY)
27900   25      FORMAT(' ',I3/(' ',5A10))
28000           CLOSE(UNIT=22,DEVICE='DSK',ACCESS='SEQOUT')
28100           RETURN
28200           END
28300
28400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
28500   C
28600   C       THE SUBROUTINE GET RETRIEVES THE LIST OF PICTURE
28700   C       FILE NAMES FROM STORAGE.
28800   C
28900   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
29000           SUBROUTINE GET
29100           DOUBLE PRECISION FILEN(30)
29200           COMMON /STORAG/FILEN,MANY
29300           OPEN(UNIT=22,DEVICE='DSK',ACCESS='SEQIN')
29400           READ(22,10) MANY
29500           READ(22,5) (FILEN(I),I=1,MANY)
29600   10      FORMAT(1X,I3)
29700   5       FORMAT(1X,5A10)
29800           CLOSE(UNIT=22,DEVICE='DSK',ACCESS='SEQIN')
29900           RETURN
30000           END
30100
30200   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
30300   C
30400   C       THE SUBROUTINE LIST PROVIDES THE USER WITH THE LIST OF
30500   C       PICTURE FILE NAMES SO THAT HE MAY CHOOSE A UNIQUE NAME
```

```
30600   C       FOR A NEW FILE.
30700   C
30800   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
30900           SUBROUTINE LIST
31000           DOUBLE PRECISION FILEN(30)
31100           COMMON /STORAG/FILEN,MANY
31200   C
31300   C       RETRIEVE THE LIST OF FILE NAMES AND PRINT THEM.
31400   C
31500           CALL NEWPAG
31600           CALL GET
31700           CALL ANMODE
31800           WRITE(5,5) (FILEN(I),I=1,MANY)
31900   5       FORMAT(' ',A10)
32000           CALL RESET
32100           RETURN
32200           END
32300
32400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
32500   C
32600   C       THE SUBROUTINE HELP PRINTS THE HELP FILE.
32700   C
32800   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
32900           SUBROUTINE HELP
33000           INTEGER WORDS(12)
33100           OPEN(UNIT=21,FILE='HELP.DAT',ACCESS='SEQIN')
33200           DO 5 I=1,20
33300             READ(21,10) (WORDS(J),J=1,12)
33400             WRITE(5,10) (WORDS(J),J=1,12)
33500   5       CONTINUE
33600   10      FORMAT(12A5)
33700           CLOSE(UNIT=21,FILE='HELP.DAT',ACCESS='SEQIN')
33800           RETURN
33900           END
34000
34100   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
34200   C
34300   C       THE MAIN PROGRAM REQUESTS THE INSTRUCTION FROM THE USER
34400   C       AND THEN CONTROLS WHICH SUBROUTINES ARE CALLED TO
34500   C       EXECUTE THAT INSTRUCTION.
34600   C
34700   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
34800   C
34900   C       SUBROUTINES CALLED:
35000   C         MENU-SETS UP MENU
35100   C         INSTR-TAKES INSTRUCTION FROM MENU
35200   C         NPLANE-INPUTS OBJECT POINTS ONE PLANE AT A TIME
35300   C         FIRST1-GETS ORIGINAL POINTS FROM DISK
35400   C         SHOW-DISPLAYS LINE DRAWING
35500   C         SAVE-PUTS PICTURE DESCRIPTION ON DISK
35600   C         RECALL-RECALLS A PICTURE DESCRIPTION FROM DISK
35700   C         LIST-GIVES A LIST OF ALL SAVED PICTURES
35800   C         HOMOG-ADDS FOURTH COORDINATE
35900   C         CENTR-FINDS CENTROID OF OBJECT
36000   C         ROTATE-ROTATES OBJECT
36100   C         TRANSL-TRANSLATES OBJECT
36200   C         SCALE-SCALES OBJECT
36300   C
36400   C       VARIABLES USED:
36500   C         PTNUM-NUMBER OF POINTS
36600   C         POLCNT-NUMBER OF POLYGONS
```

```
36700    C          POLYGN-POINTS WHICH MAKE UP POLYGONS
36800    C          LINES-POINTS WHICH ARE ENDPOINTS OF LINES
36900    C          VIEWPT-VIEWPOINT FROM WHICH OBJECT IS OBSERVED
37000    C          CENTER-POINT WHICH IS LOOKED TOWARDS
37100    C          FILEN-LIST OF FILE NAMES FOR PICTURE DESCRIPTIONS
37200    C          MANY-NUMBER OF STORED PICTURE DESCRIPTIONS
37300    C          LINCNT-NUMBER OF LINES
37400    C          ZMID-MIDDLE OF OBJECT ALONG Z AXIS
37500    C          DEGREE-ANGLE OF ROTATION IN DEGREES
37600    C          RAD-ANGLE OF ROTATION IN RADIANS
37700    C          FACT-SCALING FACTOR
37800    C
37900              INTEGER PTNUM,POLCNT,POLYGN(600,10),LINES(1000,5)
38000             1  ,LOCX(10),LOCY(10),ORDER,N,M,NPLT(20,2)
38100              REAL MATS(4,4),MATR(4,4),MAT(4,4)
38200              DOUBLE PRECISION FILEN(30)
38300              REAL POINTS(300,4),VIEWPT(3),CENTER(3)
38400              COMMON /STORAG/ FILEN,MANY /DESCRP/ POLYGN,LINES,POINTS
38500             1   /PNTRS/ PTNUM,LINCNT,POLCNT,ZMID,NP /MENUIN/
38600             2  LOCX,LOCY,N,M /PARTS/NPLT
38700              COMMON /TRANS/ MAT
38800              DATA VIEWPT(1),VIEWPT(2),CENTER(1),CENTER(2)/512.0,1000.
38900             1  ,512.0,-100.0/, NP/0/
39000              DATA MAT/1.0,4*0.0,1.0,4*0.0,1.0,4*0.0,1.0/
39100              CALL NEWPAG
39200              CALL INITT(30)
39300              CALL MENU
39400              CALL IDENT(MATR)
39500              CALL IDENT(MATS)
39600    15        CALL BELL
39700              CALL BELL
39800              CALL INSTR(ORDER,FOUND,1.,IX,IY)
39900              GO TO (1,2,3,4,5,6,7,8,9,10),ORDER
40000    1         CALL NPLANE
40100              GO TO 15
40200    2         VIEWPT(3)=ZMID
40300              OPEN(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
40400              NEND=PTNUM-1
40500              READ(21,55) ((POINTS(I,J),J=1,3),I=1,NEND)
40600    55        FORMAT(//(6E12.5))
40700              CLOSE(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
40800              CALL HOMOG(POINTS,PTNUM-1)
40900              CALL MAMULT(POINTS,MAT,300)
41000              CALL CENTR(POINTS,CENTER,PTNUM)
41100              VIEWPT(1)=CENTER(1)
41200              YMAX=0.0
41300              DO 25 I=1,PTNUM
41400                IF (POINTS(I,2) .LE. YMAX) GO TO 25
41500                YMAX=POINTS(I,2)
41600    25        CONTINUE
41700              VIEWPT(2)=YMAX+800.0
41800              CALL SHOW(VIEWPT,CENTER,LINCNT-1,PTNUM-1,VNORML)
41900              GO TO 15
42000    3         CALL SAVE
42100              GO TO 15
42200    4         CALL HELP
42300              GO TO 15
42400    5         CALL LIST
42500              GO TO 15
42600    6         CALL ANMODE
42700              CALL NEWPAG
```

```
42800             WRITE(5,29)
42900    29       FORMAT(' <AXIS,ANGLE IN DEGREES>(A1,F3.0)')
43000             READ(5,30) IAXIS,DEGREE
43100    30       FORMAT(A1,F3.0)
43200             CALL RECOVR
43300             IF (IAXIS .EQ. 'X') KAXIS=1
43400             IF (IAXIS .EQ. 'Y') KAXIS=2
43500             IF (IAXIS .EQ. 'Z') KAXIS=3
43600             RAD=DEGREE/57.2958
43700             CALL ROTATE(MATR,KAXIS,VIEWPT,CENTER,4,0,RAD)
43800             CALL MAMULT(MAT,MATR,4)
43900             GO TO 15
44000    7        CALL ANMODE
44100             CALL NEWPAG
44200             WRITE(5,35)
44300    35       FORMAT(' <INPUT THE DESIRED SCALE FACTOR.>(F3.0)')
44400             READ(5,50) FACT
44500    50       FORMAT(F3.0)
44600             CALL RECOVR
44700             SY=FACT
44800             SZ=FACT
44900             CALL SCALE(MATS,FACT,SY,SZ,0)
45000             CALL MAMULT(MAT,MATS,4)
45100             GO TO 15
45200    8        CALL CHANGE
45300             GO TO 15
45400    9        STOP
45500    10       CALL FIRST1(POINTS,PTNUM)
45600             CALL IDENT(MAT)
45700             GO TO 15
45800             END
45900
46000    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
46100    C
46200    C        GET PICTURE COORDINATES FROM SAVED FILE.
46300    C
46400    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
46500             SUBROUTINE FIRST1
46600             REAL POINTS(300,4)
46700             INTEGER PNTCNT,POLYGN(600,10),LINES(1000,5),
46800            1  POLCNT,NPLT(20,2)
46900             COMMON /DESCRP/ POLYGN,LINES,POINTS
47000             COMMON /PNTRS/ PNTCNT,LINCNT,POLCNT,ZMID,NP
47100             COMMON /PARTS/ NPLT
47200             OPEN(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
47300             READ(21,100) NP,NEND,POLCNT,LINCNT
47400             READ(21,100) ((NPLT(I,J),J=1,2),I=1,NP)
47500             READ(21,90)((POINTS(I,J),J=1,3),I=1,NEND)
47600             READ(21,100)  ((POLYGN(I,J),J=1,4),I=1,POLCNT)
47700             READ(21,100) ((LINES(I,J),J=1,5),I=1,LINCNT)
47800    90       FORMAT(6E12.5)
47900    100      FORMAT(20I4)
48000             CLOSE(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
48100             PNTCNT=NEND+1
48200             RETURN
48300             END
48400
48500    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
48600    C
48700    C        THIS SUBROUTINE FINDS THE CENTROID OF THE OBJECT SO
48800    C        THAT WHEN THE OBJECT IS ROTATED OR SCALED, IT REMAINS
```

```
48900    C        IN THE SAME PLACE.
49000    C
49100    CCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THE SUBROUTINE NPLANE IS THE BASIC INPUT SUBROUTINE FOR
C       A SLICE OF THE OBJECT ALONG THE Z-AXIS.  POINTS, LINES
C       OR POLYGONS MAY BE INPUT AND THE USER INDICATES THIS BY
C       USING THE MENU.  THE DEFAULT INPUT MODE IS LINE INPUT.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE NPLANE
C
C       SUBROUTINES CALLED:
C         INSTR-GET INSTRUCTION
C         DISPLA-SHOW CROSS-SECTION OF THIS PLANE
C         FIXUP-MAKE A CHANGE IF NECESSARY
C         CONECT-FORMS SURFACES BETWEEN PLANES
C
        INTEGER ORDER,PNTCNT,ENDPNT,POLCNT,FOUND ,END,PNTNUM,
        1  PNT,POLYPL(600),POLYGN(600,10),POLY,LINES(1000,5),
        2  PLANE(30,2),FIRST,NPLT(20,2)
        REAL POINTS(300,4)
        LOGICAL IGO
        COMMON /DESCRP/POLYGN,LINES,POINTS /MENUIN/LOCS,LOCY,IA,
        1  IB/PNTRS/ PNTCNT,LINCNT,POLCNT,ZMID,NP/PARTS/NPLT
        2  /FIX/ PLANE,FIRST,IANS,IPOL
        DATA ENDPNT,POLCNT,PNTCNT,LINCNT,NOPLAN,ORDER/2*0,3*1,2/
C
C       INPUT THE Z-COORDINATE FOR THIS SLICE.  IT IS THE X-
C       COORDINATE OF THE POINT YOU INPUT.
C
        CALL NEWPAG
        Z=0
        IGO=.FALSE.
        NP=NP+1
        NPLT(NP,1)=POLCNT+1
        CALL INITT(30)
        CALL TABINT(0,0,0)
30      CALL ANMODE
        WRITE(5,40)
40      FORMAT(' <INPUT THE Z-COORDINATE.>')
        CALL RECOVR
        CALL BELL
        FIRST=LINCNT
        IPOL=POLCNT
        CALL ONEPNT(IZ,IY)
        PLANE(NOPLAN,1)=PNTCNT
        Z=Z+IZ
        CALL LINEF
        CALL LINEF
C
C       TELL THE USER TO INPUT A POINT, EITHER FROM THE MENU OR
C       THE PICTURE, AND ACCEPT THE POINT.
C
42      CALL ANMODE
        WRITE(5,45)
45      FORMAT(' <INPUT THE PICTURE COORDINATES OR CHANGE THE ',
        1  'INPUT MODE.>')
        CALL RECOVR
5       CALL ONEPNT(IX,IY)
C
C       SEE IF AN INSTRUCTION FROM THE MENU HAS BEEN INPUT.
C
```

```
06100           CALL INSTR(ORDER,FOUND,0.0,IX,IY)
06200     C
06300     C     SEE IF THE USER IS BEGINNING A NEW SET OF LINES.
06400     C
06500           IF (FOUND .EQ. 1 .AND. ORDER .GE. 2) ENDPNT=0
06600     C
06700     C     TWO CONSECUTIVE END THE PLANE INSTRUCTIONS MEANS THE
06800     C     INPUT OF THE PICTURE IS COMPLETED.  IF END=1 THIS HAS
06900     C     OCCURRED.
07000     C
07100           IF (END .EQ. 1 .AND. ORDER .EQ. 4) GO TO 100
07200     C
07300     C     SEE IF THE INSTRUCTION WAS TO END THE PLANE.
07400     C
07500           IF (ORDER .EQ. 4) GO TO 25
07600     C
07700     C     INCREASE THE POLYGON COUNT IF A POLYGON INSTRUCTION HAS
07800     C     JUST BEEN INPUT.
07900     C
08000           IF (FOUND .EQ. 1 .AND. ORDER .EQ. 3) POLCNT=POLCNT+1
08100     C
08200     C     IF AN INSTRUCTION HAS JUST BEEN INPUT RETURN TO GET A
08300     C     POINT.
08400     C
08500           IF (FOUND .EQ. 1) GO TO 5
08600     C
08700     C     DETERMINE IF THE POINT JUST INPUT HAS BEEN INPUT BEFORE.
08800     C     IF IT HAS, THEN THE OLD LOCATION IN THE ARRAY POINTS
08900     C     WILL BE USED IN LINES.  IF IT HASN'T, THE POINT WILL BE
09000     C     PLACED IN THE ARRAY POINTS AND THAT LOCATION WILL BE
09100     C     USED IN THE LINE NODE.
09200     C
09300           END=0
09400           K=PLANE(NOPLAN,1)
09500           PNTNUM=PNTCNT
09600     C
09700     C     SEARCH FOR THE POINT.
09800     C
09900           DO 10 I=K,PNTCNT
10000             IF (DIST((IX-POINTS(I,1)),(IY-POINTS(I,2))) .GT
10100     1    .3) GO TO 10
10200     C
10300     C     THE POINT IS FOUND.
10400     C
10500           PNTNUM=I
10600           GO TO 15
10700     10    CONTINUE
10800     C
10900     C     THE POINT IS NOT FOUND.  INSERT IT INTO POINTS.
11000     C
11100           POINTS(PNTCNT,1)=IX
11200           POINTS(PNTCNT,2)=IY
11300           POINTS(PNTCNT,3)=IZ
11400           PNTCNT=PNTCNT+1
11500     C
11600     C     IF THE INSTRUCTION IS POINT, GO BACK FOR A NEW POINT.
11700     C
11800     15    IF(ORDER .EQ. 1) GO TO 5
11900     C
12000     C     SEE IF THERE IS ONE ENDPOINT OF A LINE ALREADY THERE.
12100     C
```

```
12200           IF (ENDPNT .NE. 0) GO TO 20
12300     C
12400     C     IF NOT, LET THE NEW POINT BE AN ENDPOINT(PNT) AND GET
12500     C     ANOTHER POINT.
12600     C
12700           PNT=PNTNUM
12800           ENDPNT=1
12900           GO TO 5
13000     C
13100     C     CREATE A LINE NODE BY PUTTING IN THE LOCATION OF THE
13200     C     ENDPOINTS.
13300     C
13400     20    LINES(LINCNT,3)=PNT
13500           LINES(LINCNT,4)=PNTNUM
13600           PNT=PNTNUM
13700           LINCNT=LINCNT+1
13800     C
13900     C     IF THE INSTRUCTION IS LINE, GET A NEW POINT.
14000     C
14100           IF (ORDER .EQ. 2) GO TO 5
14200     C
14300     C     POLYPL KEEPS TRACK OF WHERE TO PLACE THE LOCATION OF THE
14400     C     NEXT POINT IN THE POLYGON.
14500     C
14600           POLYPL(POLCNT)=POLYPL(POLCNT)+1
14700           IF (POLYPL(POLCNT) .NE. 1) GO TO 35
14800     C
14900     C     GET THE FIRST POINT IN THE POLYGON FROM THE LINE NODE.
15000     C
15100           POLYGN(POLCNT,1)=LINES(LINCNT-1,3)
15200           POLYPL(POLCNT)=POLYPL(POLCNT)+1
15300     C
15400     C     PUT ANOTHER POINT INTO THE POLYGON ARRAY.
15500     C
15600     35    POLY=POLYPL(POLCNT)
15700           POLYGN(POLCNT,POLY)=PNTNUM
15800     C
15900     C     PUT IN THE LINE NODE WHICH POLYGON THE LINE IS PART OF.
16000     C
16100           LINES(LINCNT-1,1)=POLCNT
16200           GO TO 5
16300     C
16400     C     ON AN END THE PLANE INSTRUCTION, DISPLAY THE INPUT FROM
16500     C     THAT PLANE AND ASK FOR VERIFICATION.
16600     C
16700     25    CALL DISPLA(LINES,LINCNT-1,POINTS,FIRST,2)
16800           CALL HOME
16900           CALL ANMODE
17000           WRITE(5,50)
17100     50    FORMAT(' <IS THIS CORRECT?>')
17200           CALL RECOVR
17300           READ(5,60) ANSWER
17400     60    FORMAT(A1)
17500     C
17600     C     IF IT ISN'T CORRECT, MAKE CHANGES.
17700     C
17800           IF (ANSWER .EQ. 'Y') GO TO 55
17900           CALL FIXUP
18000           IF (IANS .EQ. 'Y') GO TO 42
18100           GO TO 25
18200     55    PLANE(NOPLAN,2)=PNTCNT-1
```

```
18300   C
18400   C       IF TWO PLANES HAVE BEEN INPUT, CONNECT THE POINTS
18500   C       BETWEEN PLANES INTO POLYGONS.
18600   C
18700           IF (IGO) CALL CONECT(NOPLAN,PLANE)
18800           CALL NEWPAG
18900           NOPLAN=NOPLAN+1
19000           IGO=.TRUE.
19100           END=1
19200           GO TO 30
19300   100     OPEN(UNIT=21,DEVICE='DSK',ACCESS='SEQOUT')
19400           ZMID=Z/NOPLAN
19500           NPLT(NP,2)=POLCNT
19600           NEND=PNTCNT-1
19700           WRITE(21,95) NP,NEND,POLCNT,LINCNT
19800           WRITE(21,95) ((NPLT(I,J),J=1,2),I=1,NP)
19900           WRITE(21,90)((POINTS(I,J),J=1,3),I=1,NEND)
20000           WRITE(21,95) ((POLYGN(I,J),J=1,4),I=1,POLCNT)
20100           WRITE(21,95) ((LINES(I,J),J=1,5),I=1,LINCNT)
20200   95      FORMAT(20I4)
20300   90      FORMAT(6E12.5)
20400           CLOSE(UNIT=21,DEVICE='DSK',ACCESS='SEQOUT')
20500           RETURN
20600           END
20700
20800   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
20900   C
21000   C       THE SUBROUTINE CONECT CONNECTS THE POINTS OF TWO PLANES
21100   C       TOGETHER SO THAT THEY MAY FORM THE POLYGONS WHICH
21200   C       MAKE UP THE SKIN OF THE OBJECT.  THIS IS DONE BY DRAWING
21300   C       A LINE FROM A POINT A OF ONE PLANE TO THE NEAREST POINT
21400   C       B OF THE OTHER PLANE.  THEN IF THERE EXISTS A POINT C
21500   C       WHICH LIES IN THE FIRST PLANE AND IS THE OTHER ENDPOINT
21600   C       OF A LINE FROM A, A LINE IS DRAWN FROM C TO THE NEAREST
21700   C       POINT D OF THE OTHER PLANE WHICH IS THE ENDPOINT OF A
21800   C       LINE FROM B.  THEN THE POLYGONS ABD AND ACD ARE FORMED.
21900   C       IF THERE EXISTS NO C, THEN A POINT BP, THE SECOND CLOS-
22000   C       EST POINT TO A, IS FOUND.  IF THERE'S A LINE FROM B TO
22100   C       BP, THE POLYGON ABBP IS FORMED.  THIS IS DONE FOR
22200   C       EVERY POINT IN THE PLANE.
22300   C
22400   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
22500           SUBROUTINE CONECT(NOPLAN,PLANE)
22600   C
22700   C       SUBROUTINES CALLED:
22800   C         CLOSE-FINDS CLOSEST POINT
22900   C         FLINE-DETERMINES IF A LINE NODE HAS ALREADY BEEN
23000   C             FORMED
23100   C         FORML-FORMS A LINE NODE
23200   C         ALREDY-DETERMINES IF A PAIR OF POLYGONS HAS ALREADY
23300   C               BEEN FORMED
23400   C         GOFISH-DETERMINES IF ONE POLYGON HAS ALREADY BEEN
23500   C               FORMED
23600   C         FORMP-FORMS A POLYGON
23700   C
23800           INTEGER PLANE(30,2),PLANE1,PLANE2,LINES(1000,5),
23900           1  POLCNT,POLYGN(600,10)
24000           REAL POINTS(300,4)
24100           COMMON /DESCRP/POLYGN,LINES,POINTS /PNTRS/PNTCNT,LINCNT,
24200           1  POLCNT,ZMID,NP
24300   C
```

```
24400    C        DETERMINE WHICH OF THE TWO PLANES HAS THE MOST POINTS
24500    C        AND CALL THAT PLANE PLANE1 AND THE NUMBER OF POINTS
24600    C        NDIFF1.
24700    C
24800             LPLANE=POLCNT+1
24900             M=PLANE(NOPLAN-1,2)-PLANE(NOPLAN-1,1)+1
25000             N=PLANE(NOPLAN,2)-PLANE(NOPLAN,1)+1
25100             PLANE1=NOPLAN
25200             PLANE2=NOPLAN-1
25300             NDIFF1=N
25400             NDIFF2=M
25500             IF(M .LE. N) GO TO 5
25600             PLANE1=PLANE2
25700             PLANE2=PLANE2+1
25800             NDIFF1=M
25900             NDIFF2=N
26000    C
26100    C        THE DO LOOP FORMS POLYGONS USING EVERY POINT ON PLANE1.
26200    C
26300    5        DO 10 I=1,NDIFF1
26400    C
26500    C        IPNT1 IS THE FIRST POINT INPUT ON PLANE1.  AND IPNTA
26600    C        IS THE POINT TO BE USED IN THE POLYGON.
26700    C
26800             IPNT1=PLANE(PLANE1,1)
26900             IPNTA=IPNT1+I-1
27000    C
27100    C        DETERMINE WHICH POINT ON PLANE2 IS CLOSEST TO A AND
27200    C        FORM A LINE BETWEEN THEM IF THERE ISN'T ONE ALREADY.
27300    C
27400             AMIN=0
27500             CALL CLOSE(PLANE,PLANE2,IPNTA,IPNTB,AMIN)
27600             N=1
27700             M=0
27800             CALL FLINE(LINCNT,PLANE,IPNTA,M,IPNTB,ANS,MPNT,LINE1,
27900             1  PLANE1,N)
28000             IF (ANS .EQ. 0) CALL FORML(IPNTA,IPNTB,LINE1,LINCNT,
28100             1  LINES)
28200             NEXT1=1
28300             DO 35 K=1,2
28400             M=1
28500    C
28600    C        DETERMINE WHICH, IF ANY, POINT IN PLANE1 HAS A LINE
28700    C        BETWEEN IT AND A.
28800    C
28900    20       CALL FLINE(LINCNT,PLANE,IPNTA,M,0,ANS,IPNTC,LINE2,PLANE1
29000             1  ,NEXT1)
29100    C
29200    C        IF NO POINT C WAS FOUND FIND THE SECOND CLOSEST POINT TO
29300    C        A.
29400    C
29500             IF (LINE2 .EQ. 0) GO TO 30
29600    C
29700    C        IF C WAS FOUND, FIND THE CLOSEST POINT D IN THE OTHER
29800    C        PLANE WHICH IS AN ENDPOINT OF A LINE TO POINT B.
29900    C
30000             AMIN=0
30100    15       CALL CLOSE(PLANE,PLANE2,IPNTC,IPNTD,AMIN)
30200             N=0
30300             CALL FLINE(LINCNT,PLANE,IPNTC,N,IPNTD,JANS,MPNT,LINE4,
30400             1  PLANE2,IK)
```

```
30500    C
30600    C       IF IPNTA AND IPNTD ARE THE SAME FORM ONLY ONE TRIANGU-
30700    C       LAR POLYGON.
30800    C
30900            IF (IPNTB .EQ. IPNTD) GO TO 50
31000            N=1
31100            CALL FLINE(LINCNT,PLANE,IPNTB,0,IPNTD,ANS,MPNT,LINE3,
31200           1  PLANE2,N)
31300            IF(IPNTD .EQ. 0) GO TO 10
31400            IF (ANS .EQ. 0) GO TO 15
31500    C
31600    C       DETERMINE IF THE TWO POLYGONS ARE ALREADY IN THE POLY-
31700    C       GON LIST.  IF SO, FORM THE NEXT POLYGONS.
31800    C
31900            CALL ALREDY(POLCNT,IPNTA,IPNTB,IPNTC,IPNTD,POLYGN,IANS,
32000           1  LPLANE)
32100            IF (IANS .NE. 0) GO TO 35
32200    C
32300    C       CREATE POLYGONS ABD AND ACD.
32400    C
32500    50      N=1
32600            CALL FLINE(LINCNT,PLANE,IPNTA,0,IPNTD,ANS,MPNT,LINE5,
32700           1  PLANE2,N)
32800            IF (ANS .EQ. 0) CALL FORML(IPNTA,IPNTD,LINE5,LINCNT,
32900           1  LINES)
33000            IF (ANS .EQ. 0)CALL FORML(IPNTC,IPNTD,LINE4,LINCNT,
33100           1  LINES)
33200            IF (IPNTB .NE. IPNTD) GO TO 55
33300            CALL GOFISH(IPNTA,IPNTC,IPNTD,POLCNT,POLYGN,KANS)
33400            IF (KANS .EQ. 1) GO TO 35
33500    55      CALL FORMP(POLCNT,IPNTA,IPNTC,IPNTD,LINE2,LINE4,LINE5)
33600            IF (IPNTB .NE. IPNTD) CALL FORMP(POLCNT,IPNTA,IPNTB,
33700           1  IPNTD,LINE1,LINE3,LINE5)
33800    35      CONTINUE
33900            GO TO 10
34000    C
34100    C       IF C WAS NOT FOUND, FIND THE SECOND CLOSEST POINT BP
34200    C       WHICH IS THE ENDPOINT OF A LINE TO B.
34300    C
34400    30      CALL CLOSE(PLANE,PLANE2,IPNTA,IPNTBP,AMIN)
34500            CALL FORML(IPNTA,IPNTBP,LINE2,LINCNT,LINES)
34600            CALL FLINE(LINCNT,PLANE,IPNTB,0,IPNTBP,ANS,MPNT,LINE3,
34700           1  PLANE2,1)
34800    C
34900    C       FORM A POLYGON IF SUCH A LINE EXISTS.
35000    C
35100            IF (ANS .EQ. 1) CALL FORMP(POLCNT,IPNTA,IPNTB,IPNTBP,
35200           1  LINE1,LINE2,LINE3)
35300    10      CONTINUE
35400            RETURN
35500            END
35600
35700    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
35800    C
35900    C       THE SUBROUTINE FORML CREATES A LINE NODE AND RETURNS
36000    C       THE ROW NUMBER OF THAT NODE IN THE ARRAY LINES FOR USE
36100    C       IN FORMING A POLYGON.
36200    C
36300    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
36400            SUBROUTINE FORML(KPNT1,KPNT2,LINE,LINCNT,LINES)
36500            INTEGER LINES(1000,5)
```

```
36600           LINES(LINCNT,3)=KPNT1
36700           LINES(LINCNT,4) =KPNT2
36800           LINE=LINCNT
36900           LINCNT=LINCNT+1
37000           RETURN
37100           END
37200
37300   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
37400   C
37500   C       THE SUBROUTINE FORMP CREATES A POLYGON NODE AND PUTS THE
37600   C       NUMBER OF THE POLYGON IN THE LINE NODES OF EACH OF THE
37700   C       LINES WHICH MAKE UP THE POLYGON.
37800   C
37900   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
38000           SUBROUTINE FORMP(POLCNT,KPNT1,KPNT2,KPNT3,LINE1,
38100           1  LINE2,LINE3)
38200           INTEGER LINES(1000,5),POLYGN(600,10),POLCNT
38300           REAL POINTS(300,4)
38400           COMMON /DESCRP/POLYGN,LINES,POINTS
38500           POLCNT=POLCNT+1
38600           POLYGN(POLCNT,1)=KPNT1
38700           POLYGN(POLCNT,2)=KPNT2
38800           POLYGN(POLCNT,3)=KPNT3
38900           IF (LINES(LINE1,1) .NE. 0) LINES(LINE1,2)=POLCNT
39000           IF (LINES(LINE1,1) .EQ. 0) LINES(LINE1,1)=POLCNT
39100           IF (LINES(LINE2,1) .NE. 0) LINES(LINE2,2)=POLCNT
39200           IF (LINES(LINE2,1) .EQ. 0) LINES(LINE2,1)=POLCNT
39300           IF (LINES(LINE3,1) .NE. 0) LINES(LINE3,2)=POLCNT
39400           IF (LINES(LINE3,1) .EQ. 0) LINES(LINE3,1)=POLCNT
39500           RETURN
39600           END
39700
39800   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
39900   C
40000   C       THE SUBROUTINE FLINE CAN DETERMINE WHETHER THERE IS A
40100   C       LINE IN A PLANE BETWEEN TWO PARTICULAR POINTS OR IF
40200   C       THERE IS A LINE WITH A PARTICULAR ENDPOINT.  IN
40300   C       THE LATTER CASE FLINE CAN ALSO RETURN WHAT THE OTHER
40400   C       ENDPOINT IS.
40500   C
40600   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
40700           SUBROUTINE FLINE(LINCNT,PLANE,KPNT,K,KPNT2,ANS,MPNT,LIN,
40800           1  PLAN,NEXT)
40900           INTEGER PLAN,LINES(1000,5),POLYGN(600,10),PLANE(30,2)
41000           REAL POINTS(300,4)
41100           COMMON /DESCRP/POLYGN,LINES,POINTS
41200           ANS=0
41300           DO 5 I=NEXT,LINCNT
41400   C
41500   C       DOES THIS LINE HAVE AN ENDPOINT OF KPNT?
41600   C
41700           IF (LINES(I,3) .EQ. KPNT) GO TO 10
41800           IF (LINES(I,4) .EQ. KPNT) GO TO 15
41900           GO TO 5
42000   C
42100   C       IF THE INSTRUCTION IS TO SEE IF A LINE EXISTS BETWEEN
42200   C       TWO POINTS, CHECK THE OTHER ENDPOINT.
42300   C
42400   10      IF (K .EQ. 1) GO TO 20
42500           IF (KPNT2 .NE. LINES(I,4)) GO TO 5
42600           ANS=1
```

```
42700            GO TO 40
42800    C
42900    C       CHECK TO SEE IF THE OTHER ENDPOINT OF THE LINE IS ON THE
43000    C       PLANE.
43100    C
43200    20      IF(LINES(I,4) .GE. PLANE(PLAN,1) .AND. LINES(I,4) .LE.
43300            1  PLANE(PLAN,2)) GO TO 25
43400            GO TO 5
43500    C
43600    C       CHECK TO SEE IF THE OTHER ENDPOINT IS THE DESIRED ONE.
43700    C
43800    15      IF (K .EQ. 1) GO TO 30
43900            IF (KPNT2 .NE. LINES(I,3)) GO TO 5
44000            ANS=1
44100            GO TO 40
44200    C
44300    C       CHECK TO SEE IF THE OTHER ENDPOINT IS ON THE PLANE.
44400    C
44500    30      IF (LINES(I,3) .GE. PLANE(PLAN,1) .AND. LINES(I,3) .LE.
44600            1  PLANE(PLAN,2)) GO TO 35
44700    5       CONTINUE
44800            GO TO 100
44900    C
45000    C       MPNT IS THE OTHER ENDPOINT.
45100    C
45200    25      MPNT=LINES(I,4)
45300            GO TO 40
45400    35      MPNT=LINES(I,3)
45500    40      LIN=I
45600            NEXT=I+1
45700    100     RETURN
45800            END
45900
46000    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
46100    C
46200    C       THE SUBROUTINE CLOSE IS USED TO FIND THE POINT WHICH
46300    C       IS CLOSEST TO KPNT.  BY REPEATEDLY CALLING CLOSE WITH-
46400    C       OUT CHANGING THE VALUE OF AMIN IN THE CALLING PROGRAM,
46500    C       THE POINTS WILL BE RETURNED TO THE CALLING PROGRAM IN
46600    C       INCREASING ORDER OF DISTANCE FROM KPNT.
46700    C
46800    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
46900            SUBROUTINE CLOSE(PLANE,NOPLAN,KPNT,KANS,AMIN)
47000            INTEGER PLANE(30,2),LINES(1000,5),POLYGN(600,10)
47100            REAL POINTS(300,4)
47200            COMMON /DESCRP/POLYGN,LINES,POINTS
47300    C
47400    C       SET BEST TO AN INITIALLY HIGH VALUE AND LET J AND K BE
47500    C       THE RANGE OF POINTS WHOSE DISTANCES WILL BE COMPARED.
47600    C
47700            BEST=100000
47800            J=PLANE(NOPLAN,1)
47900            K=PLANE(NOPLAN,2)
48000    C
48100    C       FIND THE POINT CLOSEST TO KPNT YET FURTHER AWAY THAN
48200    C       AMIN.  CALL THE DISTANCE BEST & THE POINT KANS.
48300    C
48400            DO 5 NPT=J,K
48500              A=DIST((POINTS(NPT,1)-POINTS(KPNT,1)),(POINTS(
48600            1     NPT,2)-POINTS(KPNT,2)))
48700              IF (A .GE. BEST .OR. A .LT. AMIN) GO TO 5
```

```
48800              BEST=A
48900              KANS=NPT
49000    5       CONTINUE
49100            AMIN=BEST+0.0001
49200            RETURN
49300            END
49400
49500    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
49600    C
49700    C       THE DISTANCE FUNCTION
49800    C
49900    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
50000            FUNCTION DIST(X,Y)
50100            DIST=SQRT(X**2+Y**2)
50200            RETURN
50300            END
50400
50500    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
50600    C
50700    C       THE SUBROUTINE ALREDY DETERMINES IF A PAIR OF POLYGON
50800    C       HAS ALREADY BEEN INSERTED INTO THE LIST OF POLYGONS.
50900    C
51000    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
51100            SUBROUTINE ALREDY(POLCNT,KPNTA,KPNTB,KPNTC,KPNTD,POLYGN,
51200            1  M,LPLANE)
51300            INTEGER POLYGN(600,10),POLCNT
51400            DO 10 I=LPLANE,POLCNT,1
51500              M=0
51600              IP1=I+1
51700              DO 5 J=I,IP1
51800                DO 5 K=1,3
51900                  IF (KPNTA .EQ. POLYGN(J,K)) M=M+1
52000                  IF (KPNTB .EQ. POLYGN(J,K)) M=M+1
52100                  IF (KPNTC .EQ. POLYGN(J,K)) M=M+1
52200                  IF (KPNTD .EQ. POLYGN(J,K)) M=M+1
52300    5           CONTINUE
52400              IF (M .GE. 6) GO TO 100
52500    10      CONTINUE
52600            M=0
52700    100     RETURN
52800            END
52900
53000    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
53100    C
53200    C       THE SUBROUTINE GOFISH IS CALLED WHEN ONLY ONE TRIANGU-
53300    C       LAR POLYGON IS ABOUT TO BE FORMED.  GOFISH DETERMINES
53400    C       IF THAT POLYGON HAS ALREADY BEEN PLACED IN THE POLYGON
53500    C       LIST.  IF IT HAS, A VALUE OF 1 IS RETURNED IN M.
53600    C
53700    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
53800            SUBROUTINE GOFISH(KPNTA,KPNTC,KPNTD,POLCNT,POLYGN,M)
53900            INTEGER POLCNT,POLYGN(600,10)
54000            DO 5 I=1,POLCNT
54100              IF (KPNTD .NE. POLYGN(I,3)) GO TO 5
54200            IF (KPNTC .EQ. POLYGN(I,1) .AND. KPNTA .EQ. POLYGN(I,2))
54300            1  GO TO 10
54400    5       CONTINUE
54500            M=0
54600            GO TO 100
54700    10      M=1
54800    100     RETURN
```

```
54900           END
55000
55100   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
55200   C
55300   C       THE SUBROUTINE FIXUP ALLOWS FOR CHANGES ON A SINGLE
55400   C       PLANE.
55500   C
55600   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
55700           SUBROUTINE FIXUP
55800   C
55900   C       SUBROUTINE USED:
56000   C         SEARCH-FIND A PARTICULAR POINT
56100   C
56200           DIMENSION LINES(1000,5),POINTS(300,4)
56300           INTEGER POLYGN(600,10),PTNUM,POLCNT,IPNT(2),PLANE(30,2),
56400           1  FIRST
56500           COMMON /DESCRP/ POLYGN,LINES,POINTS /PNTRS/ PTNUM,LINCNT
56600           1  ,POLCNT,ZMID,NP /FIX/ PLANE,FIRST,IANS,IPOL
56700           CALL ANMODE
56800           WRITE(5,1)
56900   1       FORMAT(' <DO YOU WANT TO REDRAW THIS PLANE?>')
57000           CALL RECOVR
57100           READ(5,2) IANS
57200   2       FORMAT(A1)
57300           IF (IANS .NE. 'Y') GO TO 3
57400            PNTCNT=PLANE(NOPLAN,1)
57500           LINCNT=FIRST
57550           POLCNT=IPOL
57600           RETURN
57700   3       CALL LINEF
57800           CALL LINEF
57900           CALL ANMODE
58000           WRITE(5,5)
58100   5       FORMAT(' <INDICATE POINT TO BE CHANGED.>')
58200           CALL RECOVR
58300   10      CALL ONEPNT(IPNT(1),IPNT(2))
58400           CALL SEARCH(IPNT,L,FOUND,PTNUM,2)
58500           IF (FOUND .EQ. 0) GO TO 10
58600           CALL ANMODE
58700           WRITE(5,15)
58800   15      FORMAT(' <INPUT NEW COORDINATES.>')
58900           CALL RECOVR
59000           CALL ONEPNT(IX,IY)
59100           POINTS(L,1)=IX
59200           POINTS(L,2)=IY
59300           RETURN
59400           END
59500
59600   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
59700   C
59800   C       THE SUBROUTINE CHANGE ALLOWS FOR CHANGES IN THE PICTURE
59900   C       AFTER IT HAS BEEN COMPLETED.
60000   C
60100   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
60200           SUBROUTINE CHANGE
60300   C
60400   C       SUBROUTINES CALLED:
60500   C         INSTR-GETS AN INSTRUCTION
60600   C         SEARCH-FINDS  INDICATED POINT
60700   C         FLINE-FINDS A LINE NODE
60800   C
```

```
60900           DIMENSION LINES(1000,5),POINTS(300,4)
61000           LOGICAL REPEAT
61100           INTEGER POLYGN(600,10),PTNUM,POLCNT,IPNT(2),PLAN,PLANE
61200           1   (30,2)
61300           COMMON /DESCRP/ POLYGN,LINES,POINTS
61400           COMMON /PNTRS/ PTNUM,LINCNT,POLCNT,ZMID,NP
61500           COMMON /FIX/PLANE,IRST,IANS
61600           CALL BELL
61700           CALL INSTR(IORD,IFND,1.0,IX,IY)
61800     C
61900     C     THE DELETE SUB-INSTRUCTION.
62000     C
62100           IF (IORD .NE. 8) GO TO 2
62200           REPEAT=.FALSE.
62300           CALL ANMODE
62400           WRITE(5,3)
62500     3     FORMAT(' <TOUCH ENDPOINTS OF LINE ON BEEP.>')
62600           CALL RECOVR
62700           NEXT=1
62800     7     CALL BELL
62900     4     CALL ONEPNT(IPNT(1),IPNT(2))
63000           CALL SEARCH(IPNT,L,FOUND,PTNUM,3)
63100           IF (FOUND .EQ. 0) GO TO 4
63200           IF (REPEAT) GO TO 6
63300           J=L
63400           REPEAT=.TRUE.
63500           GO TO 7
63600     6     CALL FLINE(LINCNT,PLANE,J,0,L,ANS,MPNT,LIN,PLAN,NEXT)
63700           IF (ANS .EQ. 1) GO TO 8
63800           CALL ANMODE
63900           WRITE(5,9)
64000     9     FORMAT(' **LINE DOESN''T EXIST **')
64100           CALL RECOVR
64200     8     REPEAT=.FALSE.
64300           DO 40 LL=1,5
64400     40      LINES(LIN,LL)=0
64500           LIM=LINES(LIN,1)
64600           IF (LIM .EQ. 0) GO TO 11
64700     14    DO 12 I=LIM,POLCNT
64800             DO 12 J=1,4
64900     12        POLYGN(LIM,J)=POLYGN(LIM+1,J)
65000     11    IF (REPEAT) GO TO 2
65100           REPEAT=.TRUE.
65200           LIM=LINES(LIN,2)
65300           GO TO 14
65400     C
65500     C     THE ADD SUB-INSTRUCTION.
65600     C
65700     2     IF (IORD .NE. 9) GO TO 10
65800           POLCNT=POLCNT+1
65900           CALL ANMODE
66000           K=1
66100           WRITE(5,1)
66200     1     FORMAT(' <INDICATE THE  POINTS IN THE POLYGON.>')
66300           CALL RECOVR
66400     5     CALL ONEPNT(IPNT(1),IPNT(2))
66500           CALL SEARCH(IPNT,L,FOUND,PTNUM,3)
66600           IF (FOUND .EQ. 0) GO TO 5
66700           POLYGN(POLCNT,K)=L
66800           K=K+1
66900           IF (K .LE. 3)  GO TO 5
```

```
67000           GO TO 35
67100    C
67200    C      THE MOVE SUB-INSTRUCTION.
67300    C
67400    10     IF (IORD .NE. 10) GO TO 35
67500           CALL ANMODE
67600           WRITE(5,15)
67700    15     FORMAT(' <WHICH POINT?>')
67800           CALL RECOVR
67900    20     CALL ONEPNT(IPNT(1),IPNT(2))
68000           CALL SEARCH(IPNT,L,FOUND,PTNUM,3)
68100           IF (FOUND .EQ. 0) GO TO 20
68200           CALL ANMODE
68300           WRITE(5,25)
68400    25     FORMAT(' <SHOW NEW POSITION.>')
68500           CALL RECOVR
68600           CALL ONEPNT(IPNT(1),IPNT(2))
68700           IX=IPNT(1)-POINTS(L,1)
68800           IY=IPNT(2)-POINTS(L,3)
68900           OPEN(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
69000           NEND=PTNUM-1
69100           READ(21,30) ((POINTS(I,J),J=1,4),I=1,NEND)
69200    30     FORMAT(6E12.5)
69300           CLOSE(UNIT=21,DEVICE='DSK',ACCESS='SEQIN')
69400           POINTS(L,1)=POINTS(L,1)+IX
69500           POINTS(L,2)=POINTS(L,2)+IY
69600    35     RETURN
69700           END
```

VITA

Stephen P. Jolley was born in Lexington, North Carolina, on April 3, 1953. He attended elementary schools in Shelby, North Carolina, from 1959 to 1966. From 1967 to 1968 he attended Godby Junior-Senior High School in Tallahassee, Florida. He graduated from Shelby Senior High School in June 1971. The following September he entered Wake Forest University and graduated Magna cum laude in May 1975 with a Bachelor of Science degree in Mathematics. In September 1975 he received a teaching assistantship from the Computer Science Department of The University of Tennessee and began study toward a Master of Science degree. This degree was received in December 1977.

The author is a member of The University of Tennessee Student Chapter of the Association for Computing Machinery and Phi Beta Kappa. He began employment with Honeywell Information Systems, Inc., in September 1977 in Charlotte, North Carolina.