TECHNICAL REPORT 403-15

A COMPUTER PROCEDURE FOR GENERATING VISIBLE-LINE DRAWINGS OF SOLIDS BOUNDED BY QUADRIC SURFACES

Peter Hoon

December 1970

DEPARTMENT of ELECTRICAL ENGINEERING
SCHOOL of ENGINEERING and SCIENCE
NEW YORK UNIVERSITY
UNIVERSITY HEIGHTS BRONX, NEW YORK 10453

TECHNICAL REPORT 403-15

A COMPUTER PROCEDURE FOR GENERATING VISIBLE-LINE DRAWINGS OF SOLIDS BOUNDED BY QUADRIC SURFACES

Peter Woon

December 1970

Prepared By
NEW YORK UNIVERSITY
SCHOOL OF ENGINEERING AND SCIENCE
Department of Electrical Engineering
University Heights
Bronx, New York 10453

For
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Under
GRANT NUMBER AF-AFOSR-70-1854

Security Classification

DOCUMENT CONTROL DATA – R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

New York University
Department of Electrical Engineering
University Heights, Bronx, New York 10453

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

A COMPUTER PROCEDURE FOR GENERATING VISIBLE-LINE DRAWINGS OF SOLIDS BOUNDED BY QUADRIC SURFACES

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Scientific Interim

5. AUTHOR(S) *(First name, middle initial, last name)*

Peter Woon

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1970	82 + ix	31

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AFOSR 70-1854	TR 403-15
b. PROJECT NO. 9769	
c. 61102F	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d. 681304	AFOSR 71-2013

10. DISTRIBUTION STATEMENT

1. This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
THIS OTHER	Air Force Office of Scientific Rsch (NM) 1400 Wilson Blvd Arlington, Virginia 22209

13. ABSTRACT

A computer procedure for generating line drawings of solid objects bounded by quadric surfaces is described. It embodies an efficient solution to the "hidden-line" problem, that is, the problem of determining which parts of an opaque object are invisible when the object is viewed from a given vantage point. A view of a quadric object is represented by the orthographic projection of surface boundaries visible from the given vantage point. The procedure is implemented in a FORTRAN program and can be extended to do perspective drawing and shading; it may be used as the basis of a system for interactive computer-aided design of solids on a CRT display console.

DD FORM 1 NOV 68 1473

Security Classification

ACKNOWLEDGMENT

The research described in this report was sponsored by the Air Force Office of Scientific Research, Air Force System Command, USAF, under grant [redacted]-AFOSR-70-1854, Professor H. Freeman, Principal Investigator.

ABSTRACT

A computer procedure for generating line drawings of solid objects bounded by quadric surfaces has been developed. It embodies an efficient solution to the "hidden-line" problem, that is, the problem of determining which parts of an opaque object are invisible when the object is viewed from a given vantage point. The major intended area of application is in computer-aided design of machine-made objects.

A general method of specifying a "quadric object" - an object bounded by quadric surfaces - is presented. A quadric object is characterized in terms of generalized, view-dependent definitions of "vertices", "edges" and "faces". Such a characterization has made it possible to develop a hidden-line determination technique that is analogous to one that has been successfully applied to polyhedra.

A view of a quadric object is represented by the orthographic projection of surface boundaries visible from the given vantage point. The invisibility of a point is measured quantitatively by the number of surfaces hiding it. Point-by-point visibility determination can be avoided by finding the points at which visibility changes; the projections of these points are the intersections of the projections of surface boundaries. For conics, the equations of their projections may be derived and solved to find the

intersections, but piecewise-linear techniques are necessary for handling curves represented by higher-degree equations.

The procedure is implemented in a FORTRAN program, which can efficiently generate high-quality line drawings of quadric objects of fairly complicated shapes. The procedure can be extended to do perspective drawing and shading, and may be used as the basis of a system for interactive computer-aided design of solids on a CRT display console.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF SYMBOLS

S	Any quadric object
S_r	r^{th} component surface of S
Q	Vantage point
π	Picture plane
$0\text{-}X''Y''Z''$	Cartesian frame of reference with respect to which an object is specified
$q_r''(x'',y'',z'')=0$ or $q_r''=0$	Equation of S_r referred to $0\text{-}X''Y''Z''$
$w_p''(x'',y'',z'')=0$ or $w_p''=0$	p^{th} auxiliary bounding surface
$0\text{-}XYZ$	Frame of reference for a specified view
$0\text{-}YZ$	Frame of reference in π
$q_r(x,y,z)=0$ or $q_r=0$	Equation of S_r referred to $0\text{-}XYZ$
φ, θ, ψ	Angles of rotation from $0\text{-}X''Y''Z''$ to $0\text{-}XYZ$
R	Matrix of rotation from $0\text{-}X''Y''Z''$ to $0\text{-}XYZ$
λ_i, μ_i, ν_i, $i=1,2,3$	Direction cosines of OX, OY, OZ with respect to $0\text{-}X''Y''Z''$
P_k	k^{th} point whose coordinates with respect to $0\text{-}XYZ$ are x_k, y_k, z_k
P_k'	Projection of P_k in π
P_kP_h	Straight line joining P_k and P_h
$\overrightarrow{P_kP_h}$	Vector extending from P_k to P_h
$\lvert\overrightarrow{P_kP_h}\rvert$	Magnitude of $\overrightarrow{P_kP_h}$

$\vec{u}_x$, $\vec{u}_y$, $\vec{u}_z$	Unit vectors with the directions of the positive X-, Y- and Z-axes respectively
$q_r(P_k)$	Value of $q_r(x,y,z)$ at (x_k,y_k,z_k)
$\vec{n}_r(P_k)$	Outward unit normal vector on S_r at P_k
grad $q_r(P_k)$	Gradient of $q_r(x,y,z)$ at P_k (q_r may be regarded as a scalar function of the vector extending from O to (x,y,z))
$I_{r,s}$	Intersection of S_r and S_s
L_r	Limb of S_r
V_i	i^{th} vertex
V_i'	Projection of V_i onto π
$E_{i,j}$	Edge between V_i and V_j
$E_{i,j}^{\alpha}$	The α^{th} edge between V_i and V_j
$E_{i,j}'$	Projection of $E_{i,j}$ onto π
F_m	m^{th} face
F_m'	Projection of F_m onto π
$\Delta_r(P_k)$	Orientation of P_k on S_r with respect to Q
$\Omega(P_k)$	Order of invisibility of P_k
W_τ	τ^{th} visibility transition point on an edge

I. INTRODUCTION

1.1 Statement of the Problem

In computer graphics, techniques for generating pictorial representations of three-dimensional objects are of fundamental importance. In most of the existing computer drawing algorithms, objects are represented by combinations of planar surfaces only; curved surfaces must be approximated by numerous small polygons. Progress in the development of nonplanar-surface algorithms has been slow primarily because of the computational difficulties presented by nonlinear equations.

The simplest of the curved surfaces are the quadric surfaces. In a Cartesian coordinate system, a quadric surface is the locus of the general second-degree equation of the form

$$q(x,y,z) = a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5yz + a_6zx + a_7x + a_8y + a_9z + a_0 = 0, \qquad (1.1)$$

where the a_i are real numbers. Quadric surfaces are so simple and familiar that they can be readily conceived by a designer. An object of fairly complex shape can be represented by the combination of a few quadric surfaces. Most of the mechanical parts in commercial machines have simple quadric surfaces.

The phenomenon that light is stopped by opaque matter plays a major part in our visual experience. If a picture of a real object is to reflect this reality, those portions of the object that are hidden by itself from an observer at a given vantage point should be omitted from the picture. Visibility determination is generally regarded as the most formidable problem in computer rendering of solid objects.

The objective of the research reported in this thesis was to develop an efficient computer procedure for making a line drawing (see Sec. 2.3) of any view of an object bounded by quadric surfaces, with the hidden parts of the object omitted from the drawing. The major application in mind was computer-aided design of common machine-made objects.

1.2 Basic Definitions

A surface is the locus of a point in three-space whose Cartesian coordinates always satisfy one equation. A bounded surface is a finite, connected portion of a surface delimited by intersections with other surfaces. A self-bounded surface is a surface on which a path of finite length can be traced between two arbitrary points on that surface, e.g., a sphere or an ellipsoid. An object (a model of a real object, in fact) is either a self-bounded surface or a finite set of bounded surfaces every one of which is bounded by one or more of the other surfaces in the set. Any one of the

bounded surfaces of an object is called a component surface of the object. A single-compartment object is an object that divides the three-space into a totally enclosed interior region and an exterior region extending to infinity. A continuous path can be traversed between two arbitrary points in the same region without intersecting any one of the component surfaces of the object, but any path between two points in different regions must intersect at least one of the component surfaces. To an observer situated outside a single-compartment object, the object appears to be "solid", and one side of every one of its component surfaces is never visible. A quadric object is a single-compartment object, every component surface of which is a bounded quadric surface. Let P_k be any point on an object S, and Q be the vantage point in the exterior region. P_k is said to be invisible or hidden if the straight line QP_k pierces at least one of the component surfaces of S in at least one point between Q and P_k. (This definition implies that all of the component surfaces of S are opaque). A hidden curve segment or hidden line is a curve segment all points on which are hidden.

1.3 Work Done by Others

In his work on the reconstruction of a solid object from a photograph of the object, Roberts [22] solved the hidden-line problem for objects constructed from cubes, wedges

and hexagonal prisms. Using a computer to make perspective movies, Zajac [31] also had a method for eliminating hidden lines for restricted cases of polyhedra. The first general and efficient computer algorithms for making line-drawings of opaque polyhedra were developed by Appel [1] and Loutrel [19]; the former went on to develop a technique for shading a line-drawing of a polyhedron and exhibiting the shadows cast by the polyhedron [2]. Galimberti and Montanari [14] also developed a method similar to Loutrel's.

Luh and Krolak [20] first used simple quadric surfaces to model and draw machine parts. Weiss [28] developed a complete procedure for drawing any combination of bounded quadric surfaces. These three researchers adopted a point-by-point solution to the hidden-line problem: closely-spaced points on surface boundaries are computed and a visibility test is performed on each point against all the bounded surfaces. Davis et. al. [9] used a method called "combinatorial geometry" to describe and display objects constructed by combining a number of simple geometric shapes such as cylinders, ellipsoids and rectangular parallelepipeds. Comba [6] dealt with the problem of detecting intersections between convex quadric objects. A by-product of his work is a method for determining hidden lines, which, though mathematically elegant, is less efficient than Weiss' method in general.

Coons [7,8] has developed a very powerful mathematical technique for specifying and displaying on a CRT console "free-form" surfaces such as those found on the bodies of airplanes, ships, etc. In Coons' method a complicated surface is constructed by smoothly piecing together surface "patches" specified by boundary curves; the surface is rendered by the perspective projection of a set of parametrically defined curves on the surface. No attempt has been made to solve the hidden-line problem for "Coons' surfaces", which are essentially sixth-order surfaces. Higher-order surfaces have also been treated by Kubert et. al. [18] for the purpose of visualizing any continuous, single-valued function of two variables. A portion of the surface defined by such a function is rendered by the perspective projection of two orthogonal families of curves on the surface. Determination of the curve segments hidden by the surface itself is made only after the surface has been approximated by planar triangles whose vertices are the points of intersection between the two families of curves.

At present, research is most active on programming and hardware techniques for generating shaded or "half-tone" pictures of solids on a CRT display screen. By means of raster scanning and hardware aids, a photograph-like picture of a planar-surfaced structure is displayed in the form of a

fine grid of light spots of varied intensities. Such a method is not, however, directly applicable to making line-drawings of objects with curved surfaces. This trend of research was initiated by Wylie et. al. [30], and successive improvements on their techniques have been made by Warnock [25,26], Bouknight [3] and Watkins [27]. Kelley [17] extended Bouknight's method to do shadowing.

The researchers at the General Electric Electronics Research Laboratory took a total-hardware approach [4,11]. A large array of special-purpose circuitry (85,000 logic gates in 52,000 integrated circuits) was used to build a simulator that can display a moving, planar-surfaced spacecraft in color; hidden parts are removed as quickly as the vehicle turns.

Watkins [27] made an appropriate classification of the existing hidden-line algorithms. According to his classification, the algorithms belong to either of two major categories: the "path-of-edges" category and the "sample-space" category. In the former category, various methods are used to trace along the edges (surface boundaries) of objects and determine which portions of the edges are invisible; this type of algorithm is represented by the work of Roberts, Appel, Loutrel and Weiss. In the latter category, the hidden-line problem is solved at discrete points on a two-dimensional

picture grid; this type of algorithm is represented by the work of Wylie et. al., Davis et. al., Warnock, Bouknight and Watkins. The procedure described in this report belongs to the path-of-edges category.

II. OBJECT SPECIFICATION AND RENDERING

2.1 Mathematical Specification of a Quadric Object

A polyhedron can be conveniently specified in terms of its vertices, edges and faces, but to specify a quadric object unambiguously is not so simple. In one form or another, a complete specification must contain the equivalent of the following information:

1. The equations of the surfaces from which the component surfaces of the object are formed. They will be referred to as the "equations of the component surfaces".
2. The polarity of each component surface, that is, an indication as to which of the two sides of the surface is on the outside of the object. (Note that from the definition of a quadric object, the inner side is never visible from any point outside the object).
3. The bounds of each component surface, that is, a Boolean combination of inequalities specifying exactly how the component surface is bounded. Each inequality expresses the bounding effect of another component surface or that of an auxiliary bounding surface which is a transparent surface introduced for the sole purpose of eliminating ambiguities.

In more precise terms, a quadric object is specified as follows:

An object S is composed of N component surfaces denoted by S_i, $i = 1,2,\ldots,N$. Each S_i is a portion of a surface represented by an equation of the form of (1.1):

$$q_i(x,y,z) = 0. \tag{2.1}$$

The polarity of S_r is given implicitly by the equation above such that grad q_r at every point on S_r is a normal vector pointing into the exterior region. Every point on S_r satisfies not only (2.1) but also a specified Boolean combination of inequalities each of which is either of the form

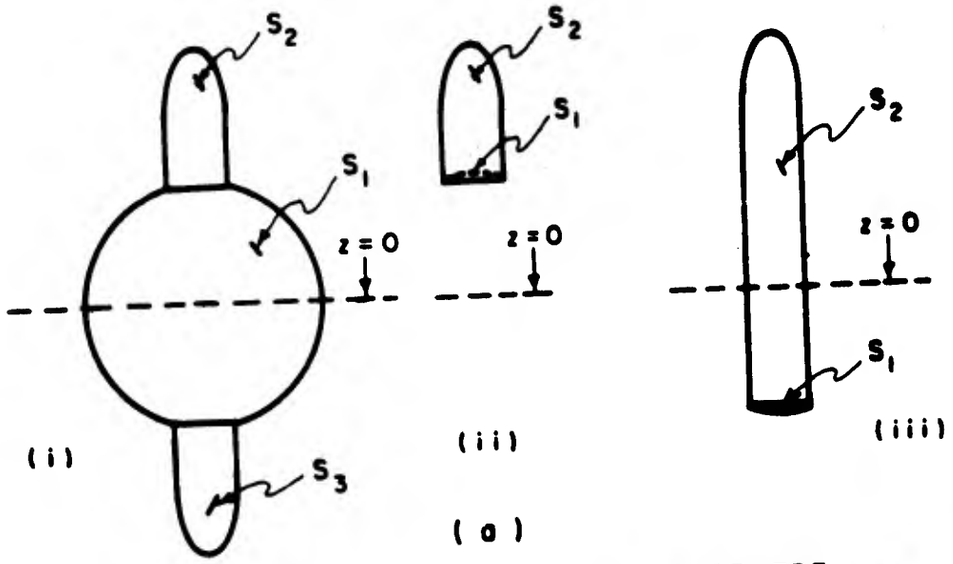
$$q_s(x,y,z) \ \sigma \ 0, \tag{2.2}$$

where $q_s(x,y,z) = 0$ is the equation of S_s bounding S_r, and σ denotes one of the relational operators $\leq$, $\geq$; or of the form

$$w_p(x,y,z) \ \sigma \ 0, \tag{2.3}$$

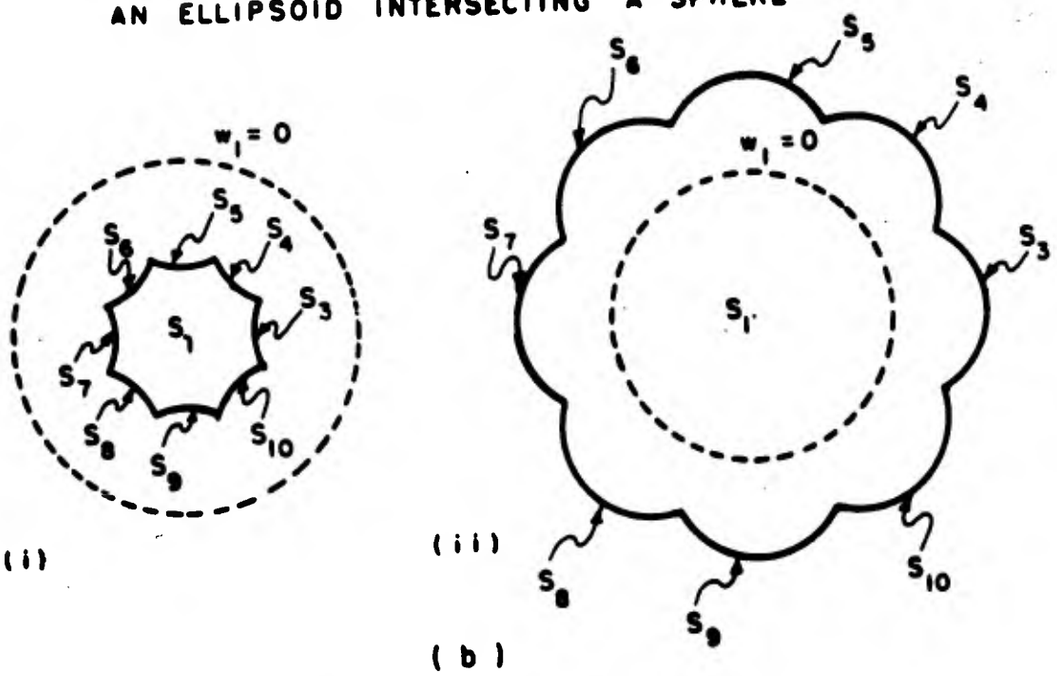
where $w_p(x,y,z) = 0$ is the equation of an auxiliary bounding surface bounding S_r.

In Fig. 1, two sets of simple quadric objects are shown in orthographic projection (see Sec. 2.2). The objects in Fig. 1a are all formed by a sphere intersecting an ellipsoid of revolution, and they are viewed in such a direction that the curves of intersection appear as straight lines. The object in Fig. 1a(i) is specified as follows:



(a)

AN ELLIPSOID INTERSECTING A SPHERE



(b)

CYLINDERS AND PLANES INTERSECTING ONE ANOTHER

FIGURE 1

EXAMPLES TO ILLUSTRATE OBJECT SPECIFICATION

$$q_1 = x^2 + y^2 + z^2 - 9;$$
$$q_2 = 36x^2 + 36y^2 + z^2 - 36;$$
$$q_3 = q_2;$$
$$w_1 = z;$$
$$S_1: \quad q_1 = 0 \quad \wedge \quad q_2 \geq 0;$$
$$S_2: \quad q_2 = 0 \quad \wedge \quad q_1 \geq 0 \quad \wedge \quad w_1 \geq 0;$$
$$S_3: \quad q_3 = 0 \quad \wedge \quad q_1 \geq 0 \quad \wedge \quad w_1 \leq 0.$$

Very small changes in the preceding specification lead to a quite different object shown in Fig. 1a(ii):

$$q_1 = -x^2 - y^2 - z^2 + 9;$$
$$q_2 = 36x^2 + 36y^2 + z^2 - 36;$$
$$w_1 = z;$$
$$S_1: \quad q_1 = 0 \quad \wedge \quad q_2 \leq 0 \quad \wedge \quad w_1 \geq 0;$$
$$S_2: \quad q_2 = 0 \quad \wedge \quad q_1 \leq 0 \quad \wedge \quad w_1 \geq 0.$$

Note that the polarity of S_1 is changed because the inner (concave) side of the sphere is now required to be the outside of the object.

The object in Fig. 1a(iii) is specified by:

$$q_1 = x^2 + y^2 + z^2 - 9;$$
$$q_2 = 36x^2 + 36y^2 + z^2 - 36;$$
$$w_1 = z;$$
$$S_1: \quad q_1 = 0 \quad \wedge \quad q_2 \leq 0 \quad \wedge \quad w_1 \leq 0$$
$$S_2: \quad q_2 = 0 \quad \wedge \quad (w_1 \geq 0 \quad \vee \quad q_1 \leq 0)$$

Figure 1b shows the "end-on" views of two objects each of which is formed by the intersection of two parallel planes (S_1 and S_2) with a ring of eight circular cylinders (S_i, i=3,4, ...,10). The axes of symmetry of the cylinders are parallel to the direction of view and lie on a cylinder $w_1 = 0$ which is an auxiliary bounding surface with such a polarity that $w_1 > 0$ outside the cylinder. The two planes are perpendicular to the direction of view. Omitting the details, let the equations of the parallel planes be $q_1 = 0$ and $q_2 = 0$ and the equations of the eight cylinders be $q_3 = 0$, $q_4 = 0$, ..., $q_{10} = 0$. The object shown in Fig. 1b(i) is specified as follows, where the q_i, i=3,4, ..., 10, are required to be less than zero outside the cylinders $q_i = 0$:

$$S_1: \quad q_1 = 0 \wedge w_1 \leq 0 \wedge q_3 \leq 0 \wedge q_4 \leq 0 \wedge \ldots \wedge q_{10} \leq 0;$$

$$S_2: \quad q_2 = 0 \wedge w_1 \leq 0 \wedge q_3 \leq 0 \wedge q_4 \leq 0 \wedge \ldots \wedge q_{10} \leq 0;$$

$$S_3: \quad q_3 = 0 \wedge w_1 \leq 0 \wedge q_1 \leq 0 \wedge q_2 \leq 0 \wedge q_{10} \leq 0 \wedge q_4 \leq 0;$$

$$S_4: \quad q_4 = 0 \wedge w_1 \leq 0 \wedge q_1 \leq 0 \wedge q_2 \leq 0 \wedge q_3 \leq 0 \wedge q_5 \leq 0;$$

$$\vdots$$

$$S_{10}: \quad q_{10} = 0 \wedge w_1 \leq 0 \wedge q_1 \leq 0 \wedge q_2 \leq 0 \wedge q_9 \leq 0 \wedge q_3 \leq 0.$$

A very different object (Fig. 1b(ii)) results from the following specification in which the q_i, i=3,4, ..., 10, are now required to be greater than zero outside the cylinders $q_i = 0$:

S_1: $q_1 = 0 \wedge (w_1 \leqq 0 \vee q_3 \leqq 0 \vee q_4 \leqq 0 \vee \ldots \vee q_{10} \leqq 0)$;

S_2: $q_2 = 0 \wedge (w_1 \leqq 0 \vee q_3 \leqq 0 \vee q_4 \leqq 0 \vee \ldots \vee q_{10} \leqq 0)$;

S_3: $q_3 = 0 \wedge w_1 \geqq 0 \wedge q_1 \leqq 0 \wedge q_2 \leqq 0 \wedge q_{10} \geqq 0 \wedge q_4 \geqq 0$;

S_4: $q_4 = 0 \wedge w_1 \geqq 0 \wedge q_1 \leqq 0 \wedge q_2 \leqq 0 \wedge q_3 \geqq 0 \wedge q_5 \geqq 0$;

$\vdots$

S_{10}: $q_{10} = 0 \wedge w_1 \geqq 0\ q_1 \leqq 0 \wedge q_2 \leqq 0 \wedge q_9 \geqq 0 \wedge q_3 \geqq 0$.

Even from these simple examples one can see that finding suitable auxiliary bounding surfaces for specifying a complicated object could be quite tricky. The difficulty is due to our use of equations that represent whole surfaces to describe bounded surfaces. However, the method just described gives the user considerable flexibility in modeling real objects. A language such as that designed by McIlroy [21] for using Weiss' programs [28] simplifies the task of specifying the equations of surfaces, but it cannot help the user in finding the right bounds. A common approach to the design of a system for modeling three-dimensional objects is to enable the user to construct an object by relating and transforming some basic geometric components [5,9]. A most effective means to help a user specify an object would be an interactive system that allows the user to manipulate and assemble geometric components on a CRT display console. Such a system is suggested in Sec. 8.4.

2.2 Projection and Frames of Reference

Perspective projection maps an arbitrary point P in space to a point P' on a picture plane π such that all lines PP' intersect in a common point Q which corresponds to the vantage point of an observer. Let O' be a point on π such that $O'Q \perp \pi$. O'Q is called the line of sight. Orthographic projection may be defined as a special case of perspective projection with Q at infinity, and with all lines PP' parallel to the line of sight. Orthographic projection is uniquely specified by the direction of the line of sight, which can be expressed in terms of direction cosines referred to a Cartesian frame of reference. Although perspective projection best represents objects as we see them, orthographic projection offers the advantage that distances along parallel lines in an orthographic view can be measured with a constant scale. In other words, orthographic projection shows the true relative dimensions of parallel lines on an object. The procedure presented in this report is based on orthographic projection, but it can be readily adapted to perspective projection as described in Sec. 8.1.

Let O-X"Y"Z" be the Cartesian frame of reference with respect to which an object is defined. Using OQ as the line of sight, an orthographic view of the object can be specified by an azimuth angle φ and an elevation angle θ of the vantage

point Q, and by an angle of twist ψ about OQ. Consider another frame of reference O-XYZ which initially coincides with O-X"Y"Z" and which moves with Q. For a specified view, O-XYZ is successively rotated through angles φ, θ, ψ about the Z, Y and X axes, respectively. We choose the YZ-plane to be the picture plane and O-YZ to be the picture reference axes (see Fig. 2). Thus the X-axis always coincides with OQ. The coordinates of a point with respect to O-X"Y"Z" and to O-XYZ are related by the rotational transformation

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix}, \tag{2.4}$$

where

$$R = \begin{bmatrix} \cos\varphi\cos\theta & \sin\varphi\cos\theta & \sin\theta \\ -\cos\varphi\sin\theta\sin\psi - \sin\varphi\cos\psi & -\sin\varphi\sin\theta\sin\psi + \cos\varphi\cos\psi & \cos\theta\sin\psi \\ -\cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi & -\sin\varphi\sin\theta\cos\psi - \cos\varphi\sin\psi & \cos\theta\cos\psi \end{bmatrix} \tag{2.5}$$

The projection of a point (x",y",z") is simply (y,z). To transform the equation of a surface referred to O-X"Y"Z" into an equation referred to O-XYZ, the inverse of (2.4) is required:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = R^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R^{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.6}$$

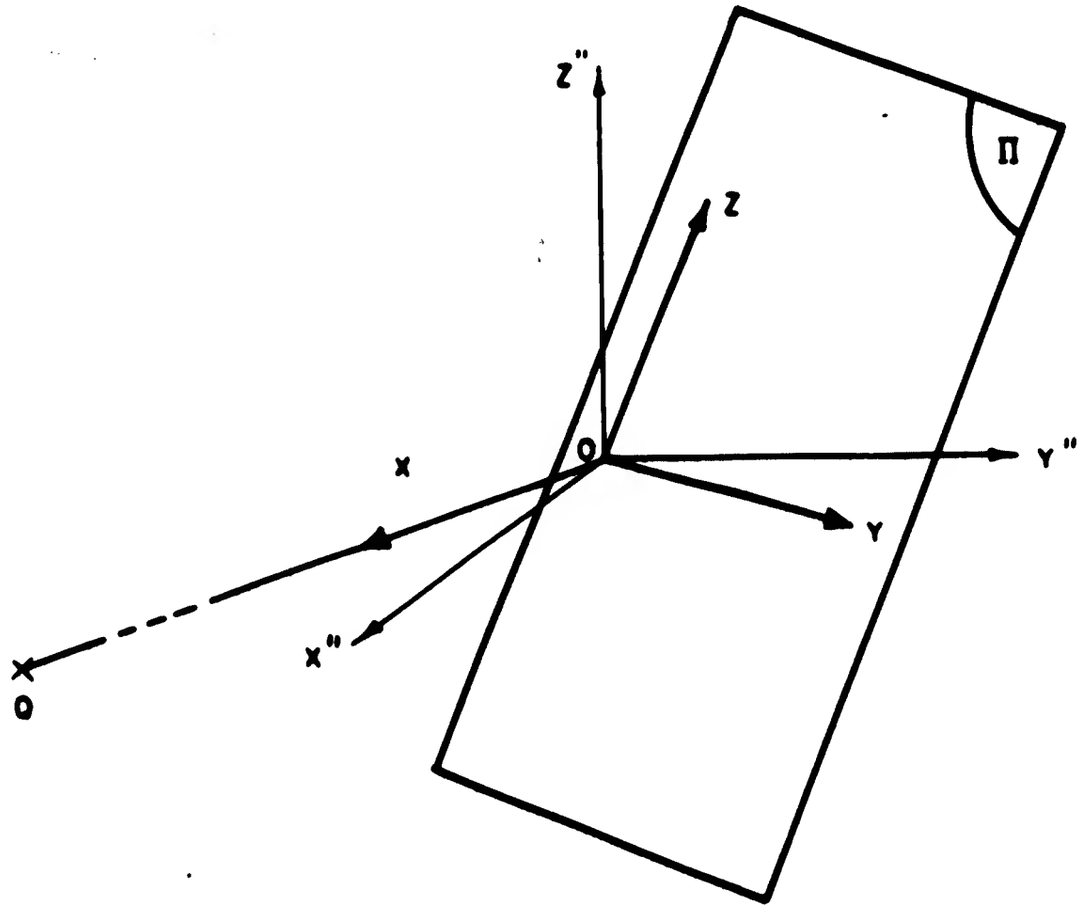


FIGURE 2

FRAMES OF REFERENCE AND THE PICTURE PLANE
FOR ORTHOGRAPHIC PROJECTION

where R^{-1} and R^T denote the inverse and transpose of R, respectively. Let q"(x",y",z") = 0 be the equation of a surface referred to 0-X"Y"Z". By substituting (2.6) into this equation the transformed equation q(x,y,z) = 0 is obtained.*

Although it is more natural to specify a direction of view by φ, θ and γ, most mathematical relations in analytical geometry are expressed in terms of direction cosines. An equivalent form of R is:

$$R = \begin{bmatrix} \lambda_1 & \mu_1 & \nu_1 \\ \lambda_2 & \mu_2 & \nu_2 \\ \lambda_3 & \mu_3 & \nu_3 \end{bmatrix} \qquad (2.7)$$

where λ_1, μ_1, ν_1; λ_2, μ_2, ν_2; and λ_3, μ_3, ν_3 are respectively the three sets of direction cosines of the X-, Y-, and Z-axes with respect to 0-X"Y"Z". The direction cosines are related to φ, θ and γ simply by equating corresponding elements of the matrices in (2.5) and (2.7). The rotation from 0-X"Y"Z" to 0-XYZ simplifies the task of deriving the formulae for view computation.

2.3 Curves Used for Rendering

Two types of curves on an object convey the essential information about the shape of the object: the view-independent

*From this point on, unprimed symbols will be used in place of primed symbols (q",x",y",z", etc.) except where distinction is necessary.

curves of intersection between component surfaces and the view-dependent curves that are the apparent ("natural") boundaries of the component surfaces. These two types of curves are usually used to represent an object in a line drawing.

Surface Intersection

The intersection of two surfaces is the locus of the point whose coordinates satisfy the equations of the two surfaces. The intersection of two quadric surfaces may consist of a single continuous section or two disjoint sections each of which is continuous. Each continuous section of an intersection may be a closed curve called a circuit, or an unbounded open curve. The intersection of two quadric surfaces is said to be planar if it (or each of its disjoint sections) lies entirely in one plane; the intersection is nonplanar (or twisted) if no section of it lies entirely in one plane. We are interested only in those segments of a surface intersection that actually exist on a given object, that is, those segments on which the coordinates of all points satisfy not only the equations but also the bounds of the two intersecting component surfaces. The intersection of S_r and S_s is denoted by $I_{r,s}$. $I_{r,s}$ is said to be protrusive (recessive) if $q_s(P_k) \leq 0$ ($q_s(P_k) \geq 0$) for all P_k on S_r in the immediate neighborhood of $I_{r,s}$, and $q_r(P_h) \leq 0$ ($q_r(P_h) \geq 0$) for all P_h on S_s in the immediate neighborhood of $I_{r,s}$. For example, $I_{3,4}$

in Fig. 1b(i) is a protrusive intersection, whereas $I_{3,4}$ in Fig. 1b(ii) is a recessive intersection. The object specification scheme described in Sec. 2.1 does not allow for any surface intersection that is partially protrusive and partially recessive.

Limb

For a given Q, the apparent boundary of a surface is the locus of the point P on the surface such that PQ is always tangent to the surface. Following the suggestion by Comba [6], we shall call this locus the limb of the surface - a term used by astronomers. The limb of S_r is denoted by L_r. Every point on L_r must satisfy $q_r=0$, the bounds of S_r and the equation

$$\vec{u} \cdot \text{grad } q_r = 0, \tag{2.8}$$

where $\vec{u} = \vec{PQ}/|\vec{PQ}|$. For a quadric surface, (2.8) is the equation of a plane, called the polar plane of Q with respect to S_r. (We shall refer to it as the "polar plane of S_r", Q being implied). Hence L_r of a quadric surface is always a conic. For orthographic projection as defined, $\vec{u} = \vec{u}_x$, a unit vector in the positive direction of the X-axis, and thus (2.8) becomes

$$2a_1x + a_4y + a_6z + a_7 = 0. \tag{2.9}$$

If $a_1 = a_4 = a_6 = 0$, no L_r exists for the given Q.

III. A VIEW-DEPENDENT CHARACTERIZATION OF A QUADRIC OBJECT

A quadric object may be characterized in terms of generalized, view-dependent "vertices", "edges", and "faces", which can be classified according to their orientation with respect to a given vantage point. Such a characterization makes it possible to develop analogues to Loutrel's technique for determining hidden lines on polyhedra [19]. To illustrate the definitions and classification given in this section, a quadric object is shown in Fig. 3 with the hidden lines dashed. The midsection of this object is an ellipsoid of revolution (S_2) which intersects a sphere (S_1) above and a paraboloid of revolution (S_3) below. The paraboloid is cut by two planes, S_4 and S_5, which also intersect each other. The object is specified such that OZ" is the axis of revolution of each of the three quadric surfaces, S_4 is parallel to the X"Y"-plane and S_5 is parallel to the X"Z"-plane. The view angles for the orthographic projection shown in Fig. 3 are: 7°, 19°, 0°.

3.1 Generalized Definitions of Vertices, Edges and Faces

Vertices

There are two types of vertices on a quadric object: real vertices and virtual vertices.

A real vertex is a point of intersection of three or more component surfaces. Its coordinates satisfy the equations

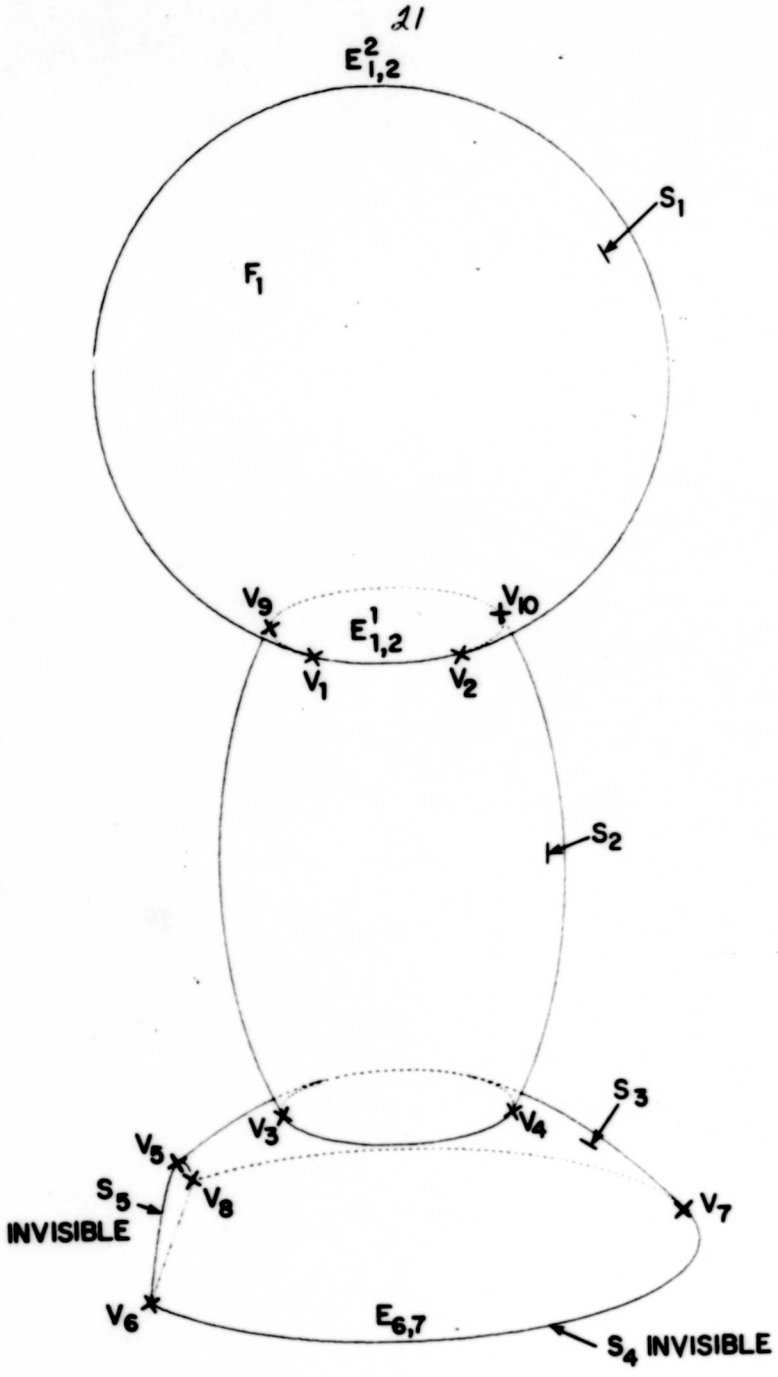


FIGURE 3

CHARACTERIZATION OF A QUADRIC OBJECT

and bounds of all the component surfaces that form it. It is the only view-independent element in our definitions.

A virtual vertex is a point at which the intersection of two component surfaces meets the limb of one of the two component surfaces. In other words, it is a point of intersection of two component surfaces and the polar plane of one of the two component surfaces. Its coordinates satisfy the equation of the polar plane and the equations and bounds of the two component surfaces. It is possible for a virtual vertex to coincide with a real vertex. In Fig. 3, V_6 is a real vertex formed by S_3, S_4 and S_5. V_1 and V_2 are virtual vertices formed by S_1, S_2 and the polar plane of S_1. V_7 is a virtual vertex formed by S_3, S_4 and the polar plane of S_3.

Edges

There are two types of edges on a quadric object: real edges and virtual edges.

A real edge is either:

1. a circuit of a surface intersection on which no vertex occurs, or
2. a segment of a surface intersection between two vertices, V_i, V_j, on which no vertex other than V_i and V_j occurs.

A virtual edge is either:

1. an entire limb (a circuit) on which no vertex occurs, or:
2. a segment of a limb between two vertices, V_i, V_j, on which no vertex other than V_i and V_j occurs.

An edge between two vertices, V_i and V_j, is denoted by $E_{i,j}$. More than one edge may share the same pair of vertices. Whenever distinction is necessary, the α^{th} (by arbitrary designation) edge between V_i and V_j is denoted by $E^{\alpha}_{i,j}$. In Fig. 3, $E^{1}_{1,2}$ is a real edge and $E^{2}_{1,2}$ is a virtual edge.

Faces

A ring of edges is a set of edges joined end to end such that every edge in the set joins exactly two other edges in the set and that no subset has this property. A border of S_r is either a ring of edges on S_r or a single edge on S_r that is an entire circuit. A face, F_m, on S_r is the whole or a portion of S_r bounded by one or more disjoint borders of S_r such that a continuous path can be traversed between two arbitrary points on F_m without crossing the border(s). Two faces may share the same border. An example of a face (F_1) on the object shown in Fig. 3 is the portion of S_1 bounded by the border that may be represented by an alternating sequence of vertices and edges: $(V_1, E^{1}_{1,2}, V_2, E^{2}_{1,2}, V_1)$.

3.2 Classification According to Orientation

The faces, edges and vertices of a quadric object are classified according to their orientation with respect to a specified vantage point. The purpose of such classification is to reduce the amount of computation that is required for the time-consuming visibility tests described in Sec. VI. The actual procedure for determining and classifying vertices and edges is described in Sec. V. Let P_k be a point on S_r. The outward unit normal vector to S_r at P_k is:

$$\vec{n}_r(P_k) = \frac{\text{grad } q_r(P_k)}{|\text{grad } q_r(P_k)|} \quad . \tag{3.1}$$

The orientation of P_k on S_r with respect to Q is defined as:

$$\begin{aligned} \Delta_r(P_k) &= +1 \quad \text{if} \quad \vec{u}_x \cdot \vec{n}_r(P_k) > 0, \\ &= 0 \quad \text{if} \quad \vec{u}_x \cdot \vec{n}_r(P_k) = 0, \\ &= -1 \quad \text{if} \quad \vec{u}_x \cdot \vec{n}_r(P_k) < 0, \end{aligned} \tag{3.2}$$

where

$$\vec{u}_x \cdot \vec{n}_r(P_k) = \frac{1}{|\text{grad } q_r(P_k)|} \frac{\partial q_r(P_k)}{\partial x} \quad . \tag{3.3}$$

P_k is said to be front-oriented, orthogonally-oriented or back-oriented on S_r with respect to Q according as $\Delta_r(P_k)$ is equal to +1, 0 or -1, respectively. In the two-dimensional illustration in Fig. 4, P_1, P_2 and P_3 are respectively back-, orthogonally- and front-oriented on S_r with respect to Q.

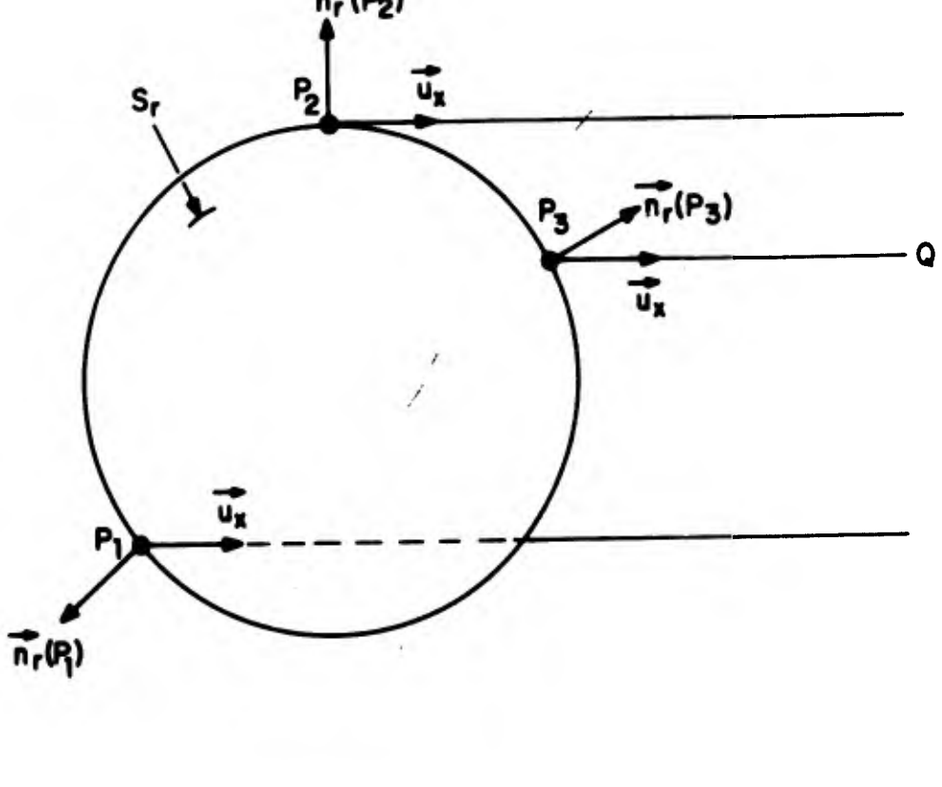


FIGURE 4

ORIENTATION OF P_k ON S_r WITH RESPECT TO Q

Face Classification

Loutrel [19] took much advantage of the property that the orientation of an entire face of a polyhedron is determined by the direction of a single normal vector on the face. On a quadric surface, the direction of a normal vector to the surface varies over the entire surface. However, the faces of a quadric object can be classified according to orientation by virtue or the following property:

Front-oriented points and back-oriented points cannot coexist on a face of a quadric object.

Proof: Let us assume that there exist two points P_1 and P_2 on F_m, which is a part of S_r, such that one of the two points is front-oriented while the other is back-oriented. F_m is continuous within its border(s), and the direction of the outward normal vector to F_m varies continuously over F_m. Choose a continuous path on F_m between P_1 and P_2 such that it does not meet the border(s) of F_m anywhere. Then, in order that $\Delta_r(P_1)$ and $\Delta_r(P_2)$ may have opposite signs there must be a point P_3 on that path where $\Delta_r(P_3) = 0$, which can occur only on a virtual edge. In other words, a virtual edge not on the border of F_m is found on F_m. Hence the definition of a face is violated. Q.E.D.

A front (back) face of S_r is a face on which all points are front- (back-) oriented except possibly on the border of the

face where orthogonally - oriented points may occur. A back face is invisible. A front face may be totally visible, or partially or totally hidden from view by other faces. Since a back face (the inner side of it, actually) is hidden by one or more front faces, any point hidden by a back face must also be hidden by one or more front faces. Therefore, the hiding effect of a back face on any other face need not be considered. A component surface may consist of a single front face, or a single back face, or both a front face and a back face. In Fig. 3, F_1, the visible portion of S_1, is a front face, whereas the invisible portion of S_1 is a back face.

Edge Classification

On a virtual edge $E_{i,j}$ in L_r, $\Delta_r(P_k) = 0$ for all P_k on $E_{i,j}$. On a real edge $E_{i,j}$ in $I_{r,s}$, $\Delta_r(P_k)$ or $\Delta_s(P_k)$ is the same for all P_k on $E_{i,j}$ except possibly for V_i or V_j. This property of invariant orientation follows directly from our definitions of edges and orientation, and is significant in that the orientation of an entire edge can be determined by testing only one point on it.

A real edge $E_{i,j}$ in $I_{r,s}$ is of Class H_1 if and only if there exists P_k other than V_i and V_j on $E_{i,j}$ such that

$$\Delta_r(P_k) \leq 0 \wedge \Delta_s(P_k) \leq 0 \qquad (3.4)$$

A real edge $E_{i,j}$ in $I_{r,s}$ is of Class H_4 if and only if there exists P_k other than V_i and V_j on $E_{i,j}$ such that

$$\Delta_r(P_k)>0 \wedge \Delta_s(P_k)>0. \tag{3.5}$$

A real edge $E_{i,j}$ in $I_{r,s}$ is of Class H_2 if and only if there exists P_k other than V_i and V_j on $E_{i,j}$ such that

$$\Delta_r(P_k)>0 \wedge \Delta_s(P_k)\leq 0 \tag{3.6}$$

and that $I_{r,s}$ is recessive.

A real edge $E_{i,j}$ in $I_{r,s}$ is of Class H_3 if and only if there exists P_k other than V_i and V_j on $E_{i,j}$ such that (3.6) is true and that $I_{r,s}$ is protrusive.

A virtual edge is always of Class H_3.

Since an H_4 edge is the intersection of two front faces, it may be called a front edge. Since an H_1 edge is the intersection of two back faces, it may be called a back edge. An H_2 or H_3 edge is the intersection of a front face and a back face, and hence may be called a boundary edge. An edge is said to be potentially visible whenever the invisibility of the edge cannot be established by considering only the surfaces forming the edge. H_1 and H_2 edges are invisible, whereas H_3 and H_4 edges are potentially visible. In Fig 3, $I_{1,2}$ contains four edges: an H_4 edge, $E^1_{1,2}$, an H_1 edge, $E_{9,10}$, and two H_2 edges, $E_{1,9}$ and $E_{2,10}$. $E_{6,7}$ in $I_{3,4}$ is a real H_3 edge. $E^2_{1,2}$ in L_1 is a virtual H_3 edge.

The classification of an edge $E_{i,j}$ may be considered a local visibility test, as it does not involve component surfaces other than the surface(s) forming $E_{i,j}$.

Vertex Classification

A front (back) vertex is a vertex that is (front-) back-oriented on all the surfaces forming it. A boundary vertex is a vertex that is orthogonally-oriented on at least one of the surfaces forming it. Front vertices and back vertices are real vertices whereas a boundary vertex may be either real or virtual. Back vertices are invisible, but front vertices and boundary vertices are potentially visible. On the object in Fig. 3, both V_6 and V_8 are the intersections of S_3, S_4 and S_5; V_8 is a back vertex, whereas V_6 and all the other vertices on the object are boundary vertices.

IV. COMPUTATION OF REAL VERTICES AND SURFACE INTERSECTIONS

Since in most design applications, many views are usually desired of an object, it is advantageous to separate the computation task into two parts: the per-object computation and the per-view computation. The per-object computation, which may be regarded as "preprocessing" for the subsequent per-view computations, is described in this section. The per-view computation is described in the two succeeding sections. It is not necessary to compute points on a planar intersection in space because the equation of its projection can be obtained and used for the computation of each view. To prepare for the per-view computation for planar intersections, the equations of the planes containing the intersections are obtained. The projection of a nonplanar surface intersection is, however, a quartic curve in general, and it is practically infeasible to use the equation of such a curve for view computation. Therefore, on each nonplanar surface intersection, closely-spaced points are computed, and for the computation of each view, the projections of the points are used instead of the equation of the curve of projection.

4.1 Computation of Real Vertices

The real vertices formed by three or more component surfaces may be computed by solving the equations of three of the surfaces simultaneously. Although there are many good

methods for solving systems of nonlinear equations, none of them is efficient or reliable for finding all solutions without initial approximations supplied by the user. Therefore, we resort to the classical method of Sylvester [23]. By the successive elimination of two variables from the three equations, a resultant equation in one variable is obtained. Subroutines for finding all roots of a polynomial equation of one variable without requiring the user to supply initial trial values can be found in many computer program libraries [15]. However, if all three equations are of the second degree, the computation of the coefficients of the 16^{th} degree resultant equation and the subsequent solution of the equation would be extremely time-consuming. In such cases, vertices can be determined in the process of computing points on a surface intersection (see Sec. 4.2). Vertices formed by three or more curved surfaces rarely occur on machine-made objects for the simple reason that it is very difficult to make such a joint with ordinary machines; most of them are formed by planes and no more than two curved surfaces.

Not all the points obtained by solving the equations of the surfaces are vertices of the object. A vertex must satisfy the bounds of all the component surfaces forming it.

Wherever possible, the relative positions of the computed vertices along the surface intersections of the object are noted.

For example, if V_1 is the only vertex formed by S_1, S_2 and S_3, and V_2 is the only vertex formed by S_2, S_3 and S_4, then V_1 must be joined to V_2 by $I_{2,3}$. However, if two or more vertices are formed by either or both of the two groups of component surfaces, the interconnection among these vertices must be determined by other means (see Secs. 4.2 and 5.3).

4.2 Computation of Points on a Surface Intersection

A simple and efficient method is used to compute closely and evenly-spaced points on a surface intersection. The 3-D Cartesian space is partitioned by three sets of parallel planes represented by:

$$x = m\delta, \quad y = m\delta, \quad z = m\delta, \tag{4.1}$$

where $m = \pm 1, \pm 2, \pm 3, \ldots$, and δ is the chosen resolution. We want to find points in which a surface intersection pierces these planes such that the distance between two consecutive points does not exceed a specified limit (e.g. 2δ). Suppose that, having started from a vertex, we have already computed k points on $I_{r,s}$. P_{k+1} is the next point to be determined. Since δ is small compared to the radius of curvature of $I_{r,s}$ at any point, the direction of $\overrightarrow{P_{k-1}P_k}$ is used to estimate the direction of $\overrightarrow{P_kP_{k+1}}$. If the magnitude of the x-component of $\overrightarrow{P_{k-1}P_k}$ is larger than that of the other two components, x_{k+1} is calculated as follows:

$$x_{k+1} = x_k + \delta \cdot \mathrm{SIGN}(x_k - x_{k-1}). \tag{4.2}$$

x_{k+1} is then substituted into $q_r = 0$ and $q_s = 0$ to obtain two equations in y_{k+1} and z_{k+1}:

$$f_r(y_{k+1}, z_{k+1}) = 0;\ f_s(y_{k+1}, z_{k+1}) = 0. \tag{4.3}$$

These two equations can be solved simultaneously by Newton's method, using y_k and z_k as the initial approximations for y_{k+1} and z_{k+1} respectively. It is possible that $I_{r,s}$ may have such a sharp bend at P_k that it fails to intersect $x=x_{k+1}$. In that case, there is no solution for (4.3) near the initial approximation, and we must try the y or the z direction instead. The process of computing points on $I_{r,s}$ by this method is called "tracing".

If real vertices have been determined on $I_{r,s}$ by the method described in Sec. 4.1, they are used as the starting and end points for tracing $I_{r,s}$. Otherwise, starting points can be obtained by cutting $I_{r,s}$ with a suitable plane which can be given in the input object specification. Undetermined vertices on $I_{r,s}$ can then be found while $I_{r,s}$ is being traced. After $I_{r,s}$ has been traced, the interconnections among all the vertices on $I_{r,s}$ have also been established.

At the starting point of a trace, P_1, there is no preceding point with which to estimate the direction of $I_{r,s}$. Hence the second point P_2 must be found by trying one by one of the six planes (of (4.1)) closest to and surrounding P_1.

A method [27] has been developed to generate a representation of each segment of $I_{r,s}$ as a 3-D chain - a sequence of incremental vectors joining nodes of the 3-D grid formed by the planes (4.1); the nodes closest to $I_{r,s}$ are chosen for the chain. If the grid is sufficiently fine, the projection of a 3-D chain gives the appearance of a smooth curve in a drawing. This method is very efficient as no iterative solution of simultaneous equations is required. However, for visibility tests, we need the exact points on $I_{r,s}$. A chain representation may nevertheless be useful for quick display of a surface intersection without hidden-line elimination.

4.3 Obtaining the Equations of Planes Containing a Planar Surface Intersection

The projection of a section of a planar intersection between two quadric surfaces is a conic. The equation obtained by eliminating x between the equations of the two surfaces is, in general, a quartic equation. It is actually the product of the equations of the projections of two sections one of which may be an imaginary conic in some plane. In order to obtain the quadratic equation representing the projection of a section of a planar intersection rather than the entire surface intersection, the equation of the plane containing that section must be determined. To accomplish this we make use of the following theorems which, together, express the conditions for the existence

of a planar intersection between two quadric surfaces.

THEOREM 1. [10,24] Let $q_1 = 0$ and $q_2 = 0$ be the equations of two quadric surfaces. The equation

$$q_1 - \kappa q_2 = 0 \tag{4.4}$$

represents for all real values of κ a quadric surface passing through all the points common to $q_1 = 0$ and $q_2 = 0$.

THEOREM 2. [10] If the rank of the discriminant matrix of a quadric surface is less than 3, the locus of the equation of the quadric surface consists of two planes. (The discriminant matrix of a quadric surface represented by an equation $q_r = 0$ of the form of (1.1) is defined as

$$D = \begin{bmatrix} c_1 & c_4/2 & c_6/2 & c_7/2 \\ c_4/2 & c_2 & c_5/2 & c_8/2 \\ c_6/2 & c_5/2 & c_3 & c_9/2 \\ c_7/2 & c_8/2 & c_9/2 & c_0 \end{bmatrix} \tag{4.5}$$

where the c_i, $i = 0,1, \ldots, 9$, are the coefficients in $q_r = 0$.)

THEOREM 3. [10] If the sum of the (n-1)-rowed principal minors of a singular symmetric matrix vanishes, its rank is less than n-1.

Let $q_3 = q_1 - \kappa q_2 = 0$, and let a_i, b_i, c_i, $i=0,1, \ldots, 9$, be the coefficients of the terms of q_1, q_2, q_3 respectively. Then

$$c_i = a_i - \kappa b_i, \quad i = 0,1,\ldots,9 \tag{4.6}$$

and by Theorem 1, $q_3 = 0$ contains the intersection of the quadric surfaces represented by $q_1 = 0$ and $q_2 = 0$. By Theorem 2, if the locus of $q_3 = 0$ is to consist of two planes, the rank of D, its

discriminant matrix, must be less than 3. Since D is symmetric, Theorem 3 can be used to express this condition as:

$$|D| = 0; \tag{4.7}$$

$$\sum_{i=1}^{4} D_{ii} = 0, \tag{4.8}$$

where $|D|$ is the determinant of D and D_{ii} is the principal minor of order 3 obtained by striking out the i^{th} row and the i^{th} column of D. If we form D by substituting (4.6) into (4.5), then (4.7) becomes a fourth-degree equation in κ and (4.8) becomes a third-degree equation in κ. Therefore, the existence of a common real solution to (4.7) and (4.8) guarantees that the intersection between the two quadric surfaces is planar. With the value of κ thus obtained, (4.4) can be factored into the linear equations of the two planes containing the intersection. The plane that contains an unwanted section of the intersection (according to specified bounds) is rejected.

V. EDGE AND PROJECTION COMPUTATION

The first step in the computation of a view is to transform the coordinates of all the real vertices and the equations of all the surfaces to refer to O-XYZ (see Sec. 2.2). The next step is to determine virtual vertices and limbs. From the surface intersections and limbs we obtain the edges of the front faces and the projections of these edges, which will then undergo the edge visibility tests described in Sec. VI. As mentioned in the preceding section, a limb or a planar surface intersection need not be traced; an edge from such a curve can be represented by the equations and bounds of its projection. The set of techniques for handling untraced edges is deemed "analytical", whereas the set of techniques for handling traced edges is deemed "piecewise-linear".

5.1 Determining Virtual Vertices and Limbs

Of the three surfaces forming a virtual vertex, at least one is a plane - the polar plane of one of the surfaces. Therefore, virtual vertices can always be determined by solving simultaneous equations (see Sec. 4.1). Virtual vertices on a traced segment of surface intersection, $I_{r,s}$, can also be found by a simple search for points at which $\partial q_r/\partial x$ or $\partial q_s/\partial x$ vanishes or changes sign.

After all the virtual vertices on a limb L_r are found, L_r can be traced by the method of Sec. 4.2 as the intersection of S_r and its polar plane. However, since L_r is planar, it can be handled by analytical techniques.

5.2 Ordering Vertices

The interconnections among the vertices on a traced curve are already established. On an untraced, planar curve, the interconnections among the vertices on the curve can be determined by ordering the projections of the vertices along the projection of the curve in a counterclockwise sense. This method requires, first of all, getting the equation of the curve projection. Let L_r' denote the projection of L_r and $I_{r,s}'$ the projection of $I_{r,s}$. The equation of (the curve containing) L_r' is obtained by eliminating the variable x between $q_r = 0$ and $\partial q_r/\partial x = 0$. If either or both of $q_r = 0$ and $q_s = 0$ are linear, elimination is made between these two equations. If both $q_r = 0$ and $q_s = 0$ are of the second degree, elimination is made between one of these equations and the plane containing $I_{r,s}$ (see Sec. 4.3).

The equation of a curve projection may represent a pair of straight lines, such as the projection of the limb of a cylinder. In that case, the equation is factored into two linear equations. The vertex projections can be separated and ordered on each of the two straight lines by a simple sort of

the y or z coordinates.

If a curve projection is a conic, vertex projections can be separated and ordered by sorting on the single-valued sections of the conic. A single-valued section of a 2-D curve is a continuous section of the curve such that there is a one-to-one correspondence between y and z values on this section and that no portion of the curve larger than and containing this section possesses this property. A conic may have up to four single-valued sections, each of which is associated with a unique combination of the signs of dz/dy and d^2z/dy^2. The sections are numbered 1,2,3 and 4, corresponding to the four sign combinations (-,+), (+,+), (-,-) and (+,-), respectively. The point of connection of two single-valued sections is an extremum, where dz/dy or dy/dz vanishes. In Fig. 5, a tilted parabola has two extrema, M_1 and M_2, and three single-valued sections: sections 4,1 and 2. V_1' and V_4' lie on section 4, V_3' lies on section 1 and V_2' lies on section 2. The counterclockwise order of V_1', V_2', V_3' and V_4' is: V_1', V_4', V_3', V_2', which is taken as the order of V_1, V_2, V_3 and V_4 in space.

5.3 Determining the Front-face Edges and their Projections

Let V_1, V_2, ..., V_n be the counterclockwise sequence of vertices on a surface intersection or limb. On every curve segment between V_i and V_{i+1}, $i = 1,2, \ldots n-1$, an

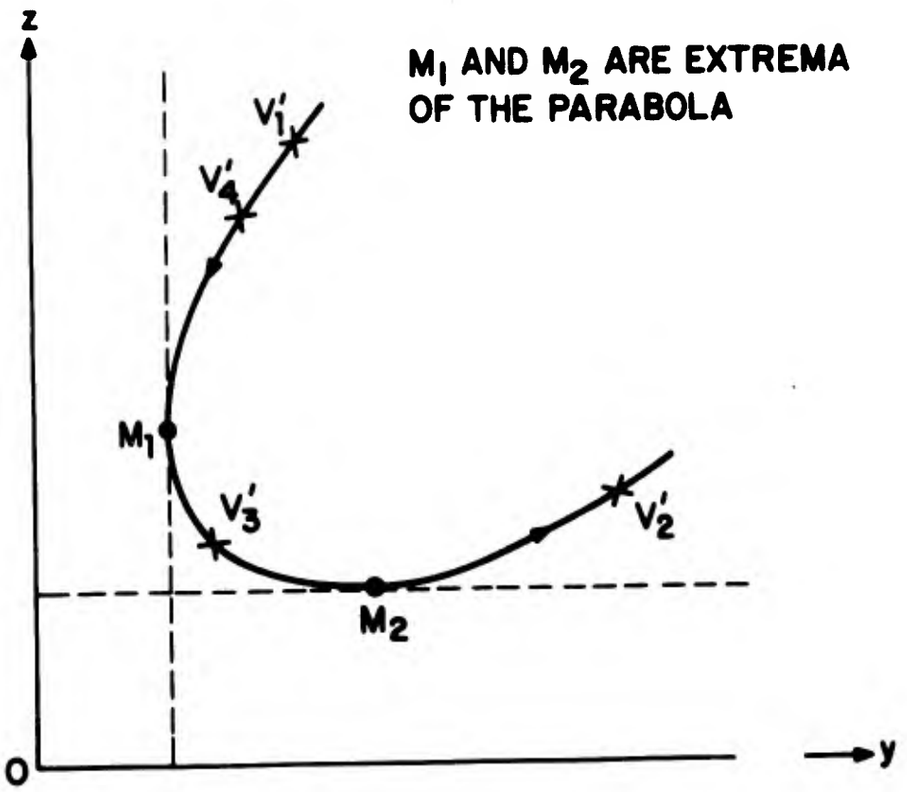


FIGURE 5

ORDERING POINTS ON A CONIC

arbitrary point P_k is tested. If P_k does not satisfy the bounds on the surfaces forming the curve, the (mathematically defined) curve segment between V_i and V_{i+1} does not actually lie on the object. If P_k satisfies the bounds, its orientations are determined to classify $E_{i,i+1}$ as described in Sec. 3.2. The H_1 edges are discarded, but all the other classes of edges are retained.

Let $E'_{i,j}$ denote the projection of $E_{i,j}$. If $E_{i,j}$ is a traced edge, $E'_{i,j}$ is represented simply by the y and the z coordinates of the list of points computed for $E_{i,j}$. If $E_{i,j}$ is not traced, $E'_{i,j}$ is represented by:

1. The equation of the curve containing $E'_{i,j}$,
2. The y and the z coordinates of V_i and V_j,
3. The numbers of the single-valued sections that are contained partially or wholly in $E'_{i,j}$, in counterclockwise order along the curve

To reduce the computation required for finding edge projection intersections (see Sec. 6.1), the maximum and minimum y and z values on each edge projection are determined. For a traced $E_{i,j}$, the maxima and minima on $E'_{i,j}$ are found in the process of computing the transformed coordinates of the points on $E_{i,j}$. For an untraced $E_{i,j}$, the maxima and minima can be obtained from the y and z coordinates of V_i, V_j and the extrema, if any, that occur on $E'_{i,j}$.

VI. EDGE VISIBILITY TESTS

Having determined the edges and their projections, we are ready to perform the global visibility tests to determine the hiding effect of front faces on potentially visible edges. Only the projections of the visible edge segments will be drawn. The order of invisibility of a point P_k, denoted by $\Omega(P_k)$, is defined by Loutrel [19] as a non-negative integer equal to the number of front faces hiding P_k from Q. An edge segment every point on which has an order of invisibility of zero is visible. The principle of the method to be presented is illustrated in Fig. 6. Consider a point P moving along an edge $E_{i,j}$. $\Omega(P)$ changes only where QP intersects a boundary edge at a point between Q and P; the change is +1 at W_1 where P just begins to hide behind the front face, the border of which contains the boundary edge; the change is -1 at W_2 where P just comes out from behind the front face. There is no change in $\Omega(P)$ when QP intersects a front edge between P and Q because P is then simultaneously coming out from behind one front face and going behind another. The points on $E_{i,j}$ where $\Omega(P)$ changes are called Ω-transition points. The projections of these points are the points of intersection of $E'_{i,j}$ with the projections of boundary edges.

6.1 Determining Intersections of Edge Projections

Finding the intersections of the projections of edges is

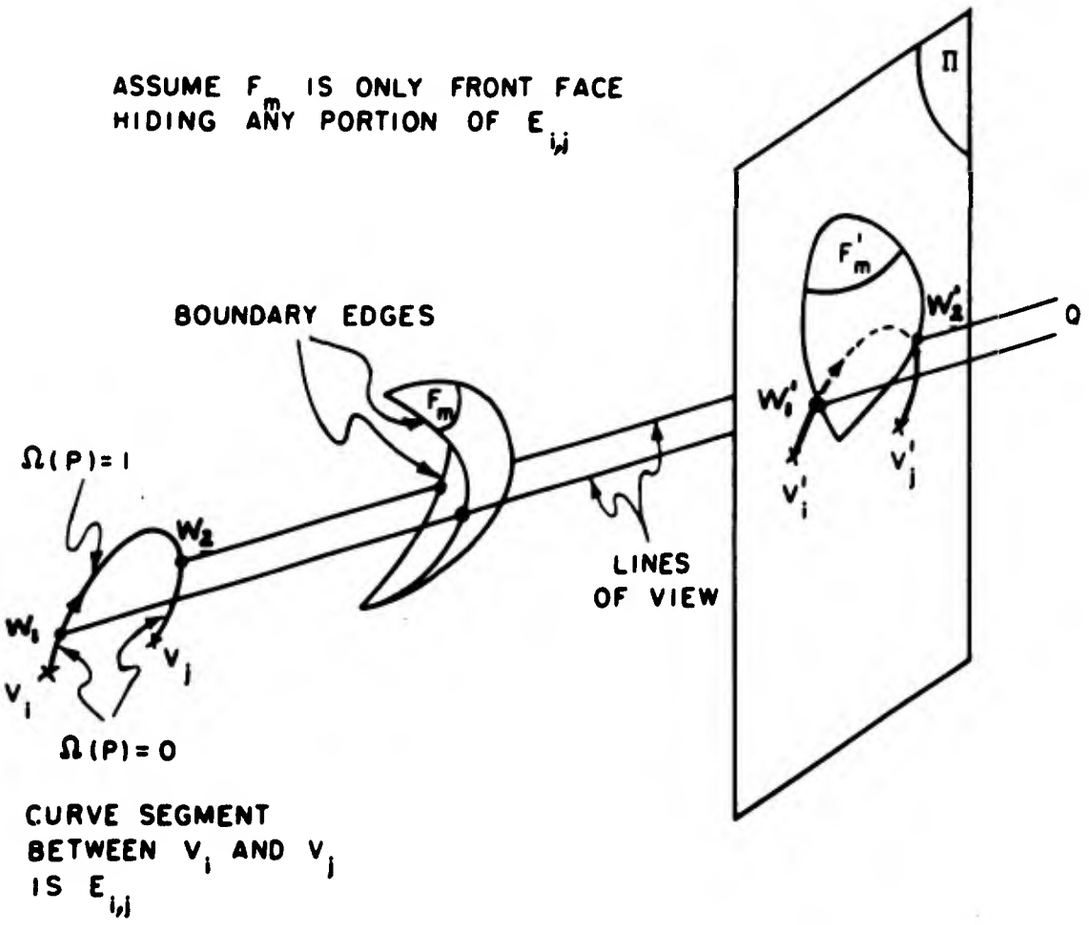


FIGURE 6

CHANGES IN THE ORDER OF INVISIBILITY ALONG AN EDGE

the most time-consuming part of visibility computation for a quadric object. A simple envelope test is first performed on edge projections to detect certain non-intersecting cases. The envelope of an edge projection is a rectangle whose sides correspond to the maximum and minimum values of y and z on the edge. Let $YMIN_{i,j}$, $ZMIN_{i,j}$, $YMAX_{i,j}$, $ZMAX_{i,j}$ denote respectively the minima and maxima of y and z values on $E'_{i,j}$. The edge projections $E'_{i,j}$ and $E'_{a,b}$ cannot intersect if their envelopes do not overlap. The nonoverlapping condition can be expressed as follows:

$$YMIN_{i,j} > YMAX_{a,b} \text{ V } ZMIN_{i,j} > ZMAX_{a,b} \text{ V } YMIN_{a,b} > YMAX_{i,j} \text{ V } ZMIN_{a,b} > ZMAX_{i,j} \tag{6.1}$$

Let $E_{i,j}$ be an untraced edge in a planar surface intersection $I_{r,s}$ and $E_{a,b}$ be an untraced edge in a planar intersection $I_{u,v}$. The intersection of $E'_{i,j}$ and $E'_{a,b}$ can be determined by solving the equations of the projection of $I_{r,s}$ and $I_{u,v}$ simultaneously. If both equations are of the second degree, Sylvester's method of elimination is used. Let (y_k, z_k) be a solution to the equation pair. (y_k, z_k) is immediately rejected as an intersection if it falls outside the envelope of either $E'_{i,j}$ or $E'_{a,b}$. Otherwise, it is tested for boundedness as follows. Let x_1 be a common solution of $q_r(x,y_k,z_k) = 0$ and $q_s(x,y_k,z_k) = 0$, and x_2 be a common solution of $q_u(x,y_k,z_k) = 0$ and $q_v(x,y_k,z_k) = 0$. If (x_1,y_k,z_k) satisfies the bounds of S_r

and S_s and (x_2, y_k, z_k) satisfies the bounds of S_u and S_v, then (y_k, z_k) is an actual intersection of $E'_{i,j}$ and $E'_{a,b}$. If either of the edges, say $E_{i,j}$, is a virtual edge, the polar plane of S_r takes the place of S_s.

The intersections of the projections of two traced edges, $E_{i,j}$ and $E_{a,b}$, are determined by comparing the y and the z coordinates of the list of points representing $E_{i,j}$ with those representing $E_{a,b}$. An associative list method is devised that requires essentially one pass through each point list to find the intersections, if any, of the two edge projections.

The list of y and z coordinates of the points on each edge is first converted into a 2-D chain [12]. A 2-D chain is a sequence of concatenated vectors, called the elements of the chain, each of which extends from one node to another of a mesh in a 2-D square grid. As shown in Fig. 7, the eight possible combinations of directions and (relative) magnitudes of chain elements are designated by the numbers 0 through 7, called the values of the chain elements. The nodes on the chain are called chain points. For the purpose of drawing, the mesh size is determined by the desired picture resolution. All coordinates are normalized so that they lie between zero and N_R, the maximum number of resolution units on either axis (e.g., $N_R = 512$). Conversion to chain is achieved simply by rounding off the coordinates of each point to the nearest node of the

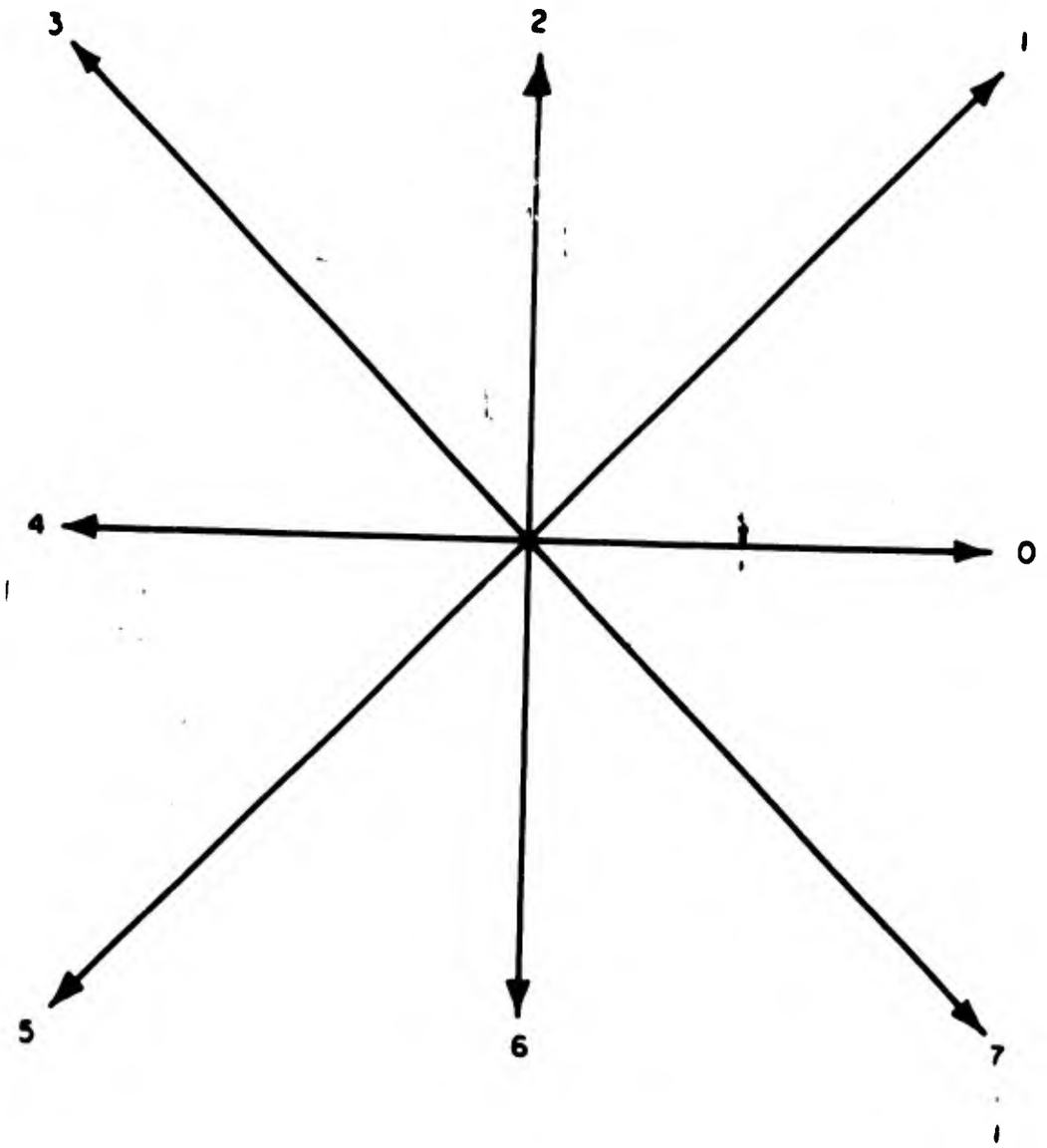


FIGURE 7

CHAIN ELEMENTS

picture grid. If two successive points fall on the same node, no chain element is generated; however, if they are more than one mesh apart, interpolation is made to generate the necessary chain elements between them.

The chain representing one of the edge projections whose intersections are to be found, say $E'_{i,j}$, is linked to an "associative store" of N_R cells as follows. All the chain points whose z coordinates equal k are linked by pointers to the k^{th} cell of the associative store. As shown in Fig. 8, a pointer is stored in the k^{th} cell, pointing to the first of a pointer-linked list of chain points in ascending sequence of the y coordinates of the chain points. A zero would be stored in the k^{th} cell if there were no chain point whose z coordinate equals k. Having thus linked the chain of $E'_{i,j}$ to the associative store, we test the chain of $E'_{a,b}$ point by point to look for its intersections with the chain of $E'_{i,j}$. Let P_m be the m^{th} point in the chain of $E'_{a,b}$, and $z_m = k$. A search is made through the list of chain points of $E'_{i,j}$ that are linked to the k^{th} cell. One of the following situations may occur:

1. y_m equals the y coordinates of one of the chain points of $E'_{i,j}$; a <u>nodal intersection</u> is found. An example of a nodal intersection is shown in Fig. 9a.
2. y_m differs from the y coordinate of a chain point of $E'_{i,j}$ by one. A "nearness flag", $NEAR_m$, is set

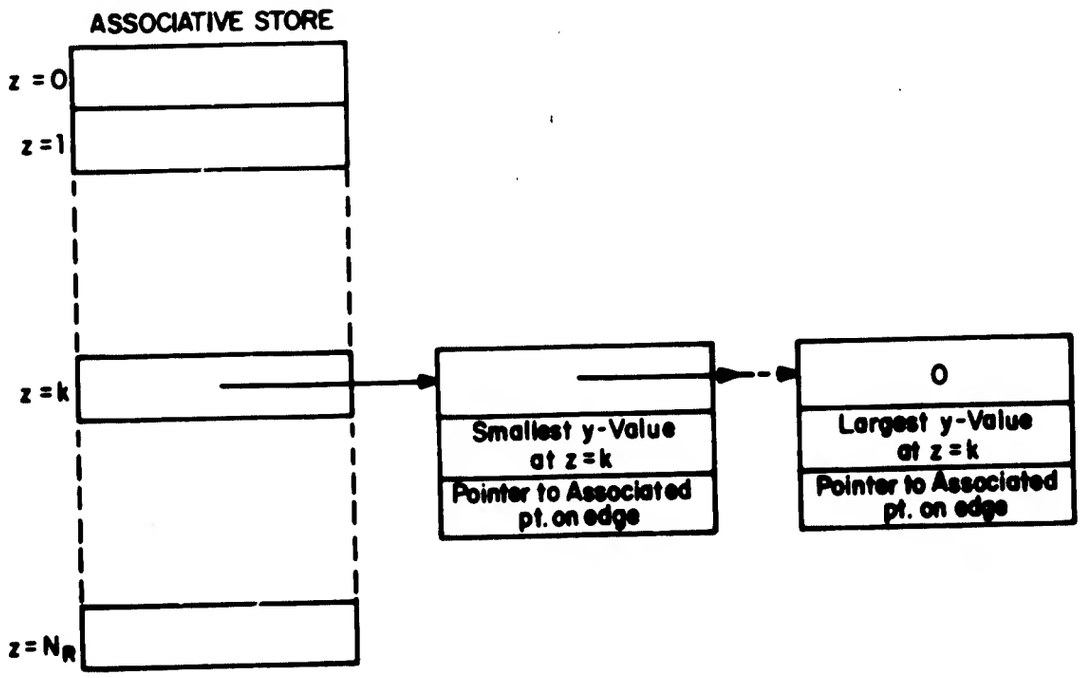
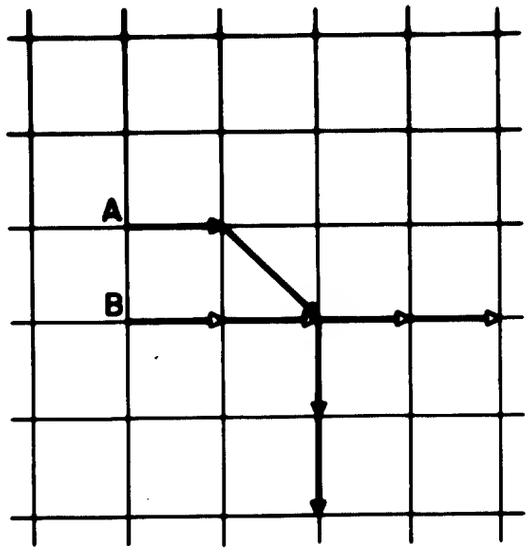
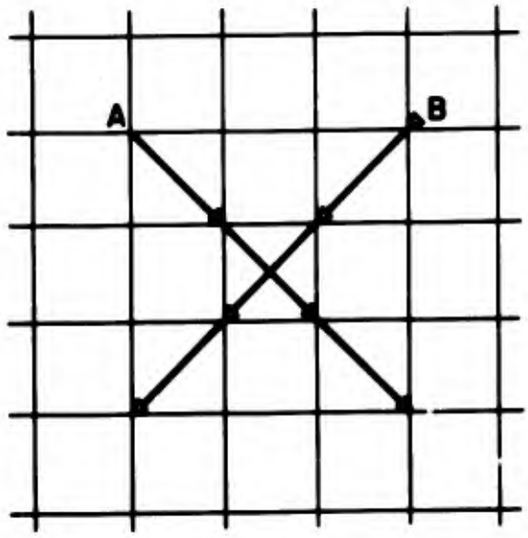


FIGURE 8

ASSOCIATIVE LIST FOR FINDING CHAIN INTERSECTIONS



(a)
NODAL INTERSECTION



(b)
NONNODAL INTERSECTION

FIGURE 9

INTERSECTION OF TWO CHAINS

such that $NEAR_m = +1$ or -1 according as y_m is greater or smaller than the other y coordinate, respectively. If a nearness flag has been set for P_{m-1} and that $NEAR_{m-1} \neq NEAR_m$, a <u>nonnodal intersection</u> has been detected. An example of a nonnodal intersection is shown in Fig. 9b. If $NEAR_{m-1}=NEAR_m$ or if $NEAR_{m-1}$ has not been set, there is no intersection yet.

3. y_m differs from the y coordinates of all the chain points linked to the k^{th} cell by more than one. There is no intersection.

After a chain intersection has been detected, the points in the point list of $E_{i,j}$ and $E_{a,b}$ corresponding to the intersection are linked to each other via an entry in a list of edge projection intersections. Since an intersection of two chains is not, in general, exactly the intersection of the curves represented by the chains, the chain intersection is used only as a very good estimate of the exact point in the visibility test described in Sec. 6.3.

If $E_{a,b}$ is a traced nonplanar edge and $E_{i,j}$ is an untraced planar edge, a "hybrid" method can be used to find their intersection. Let $f(y,z) = 0$ be the equation of the curve containing $E'_{i,j}$. At every point P_m on the chain representing $E'_{a,b}$ that falls within the envelope of $E'_{i,j}$, $f(y_m, z_m)$ is

evaluated. A possible intersection between $E'_{i,j}$ and the chain $E'_{a,b}$ occurs at P_m if $f(y_m, z_m) = 0$, or at a point between P_{m-1} and P_m if $f(y_m, z_m)$ and $f(y_{m-1}, z_{m-1})$ have opposite signs. The derivatives of $f(y,z)$ at this point are then evaluated to determine if the point actually lies on $E'_{i,j}$. The equation of the general conic is:

$$f(y,z) = a_1y^2 + a_2z^2 + a_3yz + a_4y + a_5z + a_6 = 0. \qquad (6.2)$$

$f(y,z)$ at every chain point can be evaluated incrementally, using the value of $f(y,z)$ at the preceding chain point. Let ν_i and ζ_i be the y and z increments respectively of a chain element of value i, and let the chain element $\overrightarrow{P_{m-1}P_m}$ have the value i. $f(y_m, z_m)$ is evaluated incrementally as follows:

$$\begin{aligned} f(y_m, z_m) &= f(y_{m-1}+\nu_i,\ z_{m-1}+\zeta_i) \\ &= f(y_{m-1},\ z_{m-1}) + (2a_1\nu_i + a_3\zeta_i)y_{m-1} + (2a_2\zeta_i + a_3\nu_i)z_{m-1} \\ &\quad + [a_1(\nu_i)^2 + a_2(\zeta_i)^2 + a_3(\nu_i)(\zeta_i) + a_4\nu_i + a_5\zeta_i] \\ &= f(y_{m-1},\ z_{m-1}) + g_1(i)y_{m-1} + g_2(i)z_{m-1} + g_3(i), \qquad (6.3) \end{aligned}$$

where g_1, g_2 and g_3 are functions whose values do not depend on the y and z values, and need be evaluated once only for each i, $i = 0, 1, \ldots, 7$.

On a computer for which arithmetic operations are considerably slower than operations such as comparison and data fetching, a more efficient alternative to the hybrid method is to obtain the chain representation of $E'_{i,j}$ and then determine

the intersections of two chains by the associative list method. Freeman and this author [13] have developed a simple algorithm for generating a chain representation of a section of a curve represented by $f(y,z) = 0$. It makes use of a common property of curves that the immediate neighborhood of a curve is "polarized" into one region in which $f(y,z)$ is positive and another region in which $f(y,z)$ is negative. Guided by the sign of $f(y,z)$, successive chain points are determined by selecting nodes of the square grid that lie "closest" to the curve, closeness being measured by $|f(y,z)|$ evaluated at the nodes. $f(y,z)$ is evaluated incrementally by (6.3).

Yet another alternative to the hybrid method is to trace all curves, convert their projections to chains, and then find the projection intersections. This has the advantage that no separate procedures are required for handling planar and nonplanar curves.

It may seem that the computation time for determining edge projection intersections would increase very rapidly with the number of edges. Let N_E be the number of edges. If we had to determine the intersections of <u>all</u> the edge projections with one another, indeed $N_E(N_E - 1)/2$ intersection determinations would be required. But the actual number of necessary intersection determinations is always very much smaller than that. First of all, a large number of edges are usually eliminated as

H_1 edges. No intersections need be determined between edges of the same face. Envelope tests rule out many more possible intersections. Furthermore, edge classification enables us to test only the H_4 and H_3 edges against the H_2 and H_3 edges.

6.2 The Order of Invisibility of a Point

A point P_k is hidden by the front face of a component surface S_r if and only if:

1. $q_r(x,y_k,z_k) = 0$ has real roots, and
2. one of the real roots, x_h, satisfies the following conditions:
   a. $x_h > x_k$, that is, (x_h,y_k,z_k) lies between P_k and Q,
   b. (x_h,y_k,z_k) satisfies the bounds of S_r, and
   c. (x_h,y_k,z_k) is front-oriented on S_r.

$\Omega(P_k)$ is equal to the number of front faces hiding P_k. Of course, if (y_k,z_k) does not fall inside the envelope of the projection of the front-face border on S_r, we know P_k is not hidden by that front face without further testing. $\Omega(P_k)$ is not defined if QP_k intersects any boundary edge. If QP_k intersects a front edge, only one of the faces forming the edge will be counted in determining $\Omega(P_k)$.

6.3 Propagating the Order of Invisibility Along an Edge

A constant-Ω segment on an edge $E_{i,j}$ is a segment between two neighboring Ω-transition points on $E_{i,j}$. The order of

invisibility of any point on a constant-Ω segment is said to be the "Ω of the segment." To determine the Ω of every constant-Ω segment on $E_{i,j}$, we can start at a point P_o on $E_{i,j}$, find $\Omega(P_o)$, and going toward V_i or V_j, increment or decrement Ω at the Ω-transition points. P_o may be any point whose projection, P'_o, does not lie on the projection of a boundary edge. Let $W'_1, W'_2, \ldots W'_T$, be the intersections between $E_{i,j}$ and the projections of boundary edges, encountered from P'_o to V'_j. To determine the Ω-transition points on $E_{i,j}$ between P_o and V_j, we examine each W'_t, starting with W'_1. Let $E_{a,b}$ be a boundary edge of F_m whose projection, $E'_{a,b}$, intersects $E'_{i,j}$ at W'_t. Suppose the line of view through W'_t intersects $E_{i,j}$ at W_t, and $E_{a,b}$ at W''_t (see Fig. 6). For W_t to be an Ω-transition point, W''_t must lie between Q and W_t. Let W_τ, the τ^{th} Ω-transition point on $E_{i,j}$ between P_o and V_j, be caused by F_m. Take two points, W_τ^- and W_τ^+ on $E_{i,j}$ a small distance away from W_τ and on each side of W_τ; W_τ^- is on the same side of W_τ as P_o. ω_τ, the change in Ω at W_τ is equal to -1 if W_τ^- is hidden by F_m and W_τ^+ is not; it is equal to +1 if W_τ^+ is hidden by F_m and W_τ^- is not. For T Ω-transition points between P_o and V_j, $T \leq T'$,

$$\Omega(V_j) = \Omega(P_o) + \sum_{\tau=1}^{T} \omega_\tau. \qquad (6.4)$$

When $\Omega(V_j)$ has been computed by (6.4), $\Omega(P_o)$ is said to have been <u>propagated</u> from P_o. to V_j.

If an intersection between the chains representing $E'_{i,j}$ and $E'_{a,b}$ has been found, we must perform a 3-D test on as many points on $E_{i,j}$ as necessary before and after the approximate W_τ. If visibility change is found to occur between P_{k-1} and P_k, $P_{k-1}P_k$ is further subdivided to find a closer approximation of the exact Ω-transition point.

6.4 Propagating the Order of Invisibility from Edge to Edge

To reduce the number of evaluations of $\Omega(P_o)$, the order of invisibility may be propagated from one edge to another via the connecting vertex instead of starting at a new P_o on each edge. However, an ambiguity in the order of invisibility may arise at a boundary vertex. Suppose we are approaching V_j along $E_{i,j}$. A point P_1 on that segment of $E_{i,j}$ which lies between V_j and the Ω-transition point nearest to V_j may be hidden by one or more of the front faces forming V_j. Since V_j is not hidden by any of the front faces forming V_j, $\Omega(V_j)$ is actually smaller than $\Omega(P_1)$ instead of being equal to $\Omega(P_1)$ as indicated by (6.4). To account for this discrepancy, we define the local order of invisibility of a boundary vertex V_j with respect to $E_{i,j}$, denoted by $\Omega_L(V_j/E_{i,j})$, as the number of front faces hiding P_1 on the segment of $E_{i,j}$ between W_N and V_j, W_N being the Ω-transition point nearest to V_j. $\Omega(P_1)$ is deemed the apparent order of invisibility of V_j as V_j is approached on $E_{i,j}$, and is denoted by $\Omega(V_j/E_{i,j})$. We thus have the relationship:

$$\Omega(V_j) = \Omega(V_j/E_{i,j}) - \Omega_L(V_j/E_{i,j}). \qquad (6.5)$$

Clearly, $\Omega(V_j)$ is independent of the edges at V_j. To start propagating $\Omega(V_j)$ to another edge, $E_{j,k}$, we simply compute $\Omega_L(V_j/E_{j,k})$ and add it to $\Omega(V_j)$ to obtain $\Omega(V_j/E_{j,k})$.

Since all the edges at a front vertex are H_4 edges by definition, the local order of invisibility of a front vertex V_j with respect to any of the edges at V_j is equal to zero. Hence, for a front vertex,

$$\Omega(V_j/E_{i,j}) = \Omega(V_j/E_{j,k}) = \Omega(V_j). \qquad (6.6)$$

By means of (6.4), (6.5) and (6.6), the order of invisibility can be propagated from a single starting point to many potentially visible edges via the connecting vertices. Loutrel's basic strategy [19] for traversing a path of edges is applicable here, namely: whenever a front vertex V_j is reached, the edges at V_j will be tested next. If one or more W_τ occurs at V_j, $\Omega(V_j)$ is not defined and no edge will ever be tested from V_j.

VII. IMPLEMENTATION

7.1 The Program

A complete procedure for drawing orthographic projections of quadric objects has been implemented in a program named "QUADRAW" written in FORTRAN IV - the language was chosen for its machine independence and wide availability in the academic community. QUADRAW has run successfully on an IBM 360/91 computer, and is also being tested on a UNIVAC 1108 computer. The program is quite efficient; it uses about a second of the 360/91 CPU time, on the average, to compute a view of the "spacecraft" shown in Fig. 12 and Fig. 13 (see Sec. 7.2). The object program excluding the data structure and the plotter subroutines occupies about 15,000 words of memory of the 360/91. The data structure for the spacecraft, on the average, takes up about 25,000 words. QUADRAW was written in such a way as to minimize debugging effort. Its modular design allowed for easy experimentation with various alternative algorithms. Much of the code can be "tightened up" considerably to make the program run even faster. Like most hidden-line elimination programs, QUADRAW really calls for a fast computer with a large memory. The complexity of the object that can be drawn by QUADRAW is limited only by the memory space available. However, many of the large number of subroutines of QUADRAW as well as some parts of the data structure may be overlaid during execution.

A considerable amount of effort was spent on implementing the analytical techniques for handling planar curves, but it failed to yield satisfactory results. Finding the intersections of two conics by solving the resultant quartic equation is just not reliable enough, especially when the two curves are nearly tangent to each other, which is quite common in a picture. Moreover, the analytical techniques turned out to be not more efficient overall than the piecewise-linear techniques that are applicable to both planar and nonplanar curves. Hence in QUADRAW, all curves on an object are traced and subjected to the same visibility test procedure.

From the beginning, it was realized that loss of accuracy through computer arithmetic operations would be a formidable problem for QUADRAW which performs a large number of calculations with nonlinear equations. Algorithms were carefully formulated, and error analyses were made to set suitable error tolerances for various steps of computation (for checking convergence in root-finding, singularity of a matrix, satisfaction of an equation, etc.) After much struggle, 32-bit single-precision arithmetic was found to be adequate for all subroutines except the library subroutine for obtaining roots of a polynomial [15].

The quality of the pictures produced by QUADRAW is very good. Because the algorithm for computing points on space curves yields evenly-spaced points, the curves appear smooth

from all angles even if the grid used is fairly coarse. QUADRAW computes all curves on an object with the same precision. This is wasteful of computing time. However, it would require only a small change in the program to compute the larger features of an object, such as the body of the spacecraft, with a coarser grid.

A subroutine package in a program library [16] was used to plot the output of QUADRAW on either a CALCOMP plotter or a Stromberg-Carlson 4020 microfilm recorder. It may be of interest to note that just outputing the vectors representing a view of the spacecraft onto a magnetic tape for the off-line SC 4020 consumed as much CPU time as the view computation itself. For interactive computer-aided design, the efficiency of program and hardware for the actual drawing of the vectors is certainly as important a consideration as the efficiency of view computation. Besides pictures from which hidden lines are omitted, QUADRAW can generate "wire-frame" pictures in which all lines are shown, or pictures in which the hidden lines are dashed. QUADRAW can also draw a set of nonintersecting objects.

During the execution of QUADRAW, numerous operations are performed just to store, link and retrieve data. The data structure was carefully designed for execution speed and memory conservation. A group of data items that are usually required together in computation is stored in a block of contiguous

memory, and contiguous blocks containing similar groups of data form a table. Storage for all data is dynamically allocated from a single storage pool in the form of a one-dimensional FORTRAN array placed in Block COMMON. An INTEGER-type name as well as a REAL-type name is given to the array by means of an EQUIVALENCE statement so that the array can accommodate data of mixed types. Whenever two tables are being built up simultaneously, one grows from the top down, the other grows from the bottom up. Linkage between groups of data items is effected by pointers, which are themselves elements of the array whose integer values indicate the positions of the linked items in the array. Storage allocation and data structure handling are delegated exclusively to a set of "access subroutines". Consequently, major changes to the data structure have been made without disturbing the main program logic. However, the numerous subroutine calls do cost computing time. To achieve still higher speed in QUADRAW, these access subroutines can be converted to in-line code in the most frequently executed portions of the program.

The main program logic of QUADRAW is shown in a flow diagram in Fig. 10.

7.2 Illustrative Examples

Several orthographic views of the test object used to illustrate object characterization are shown in Fig. 11. Even

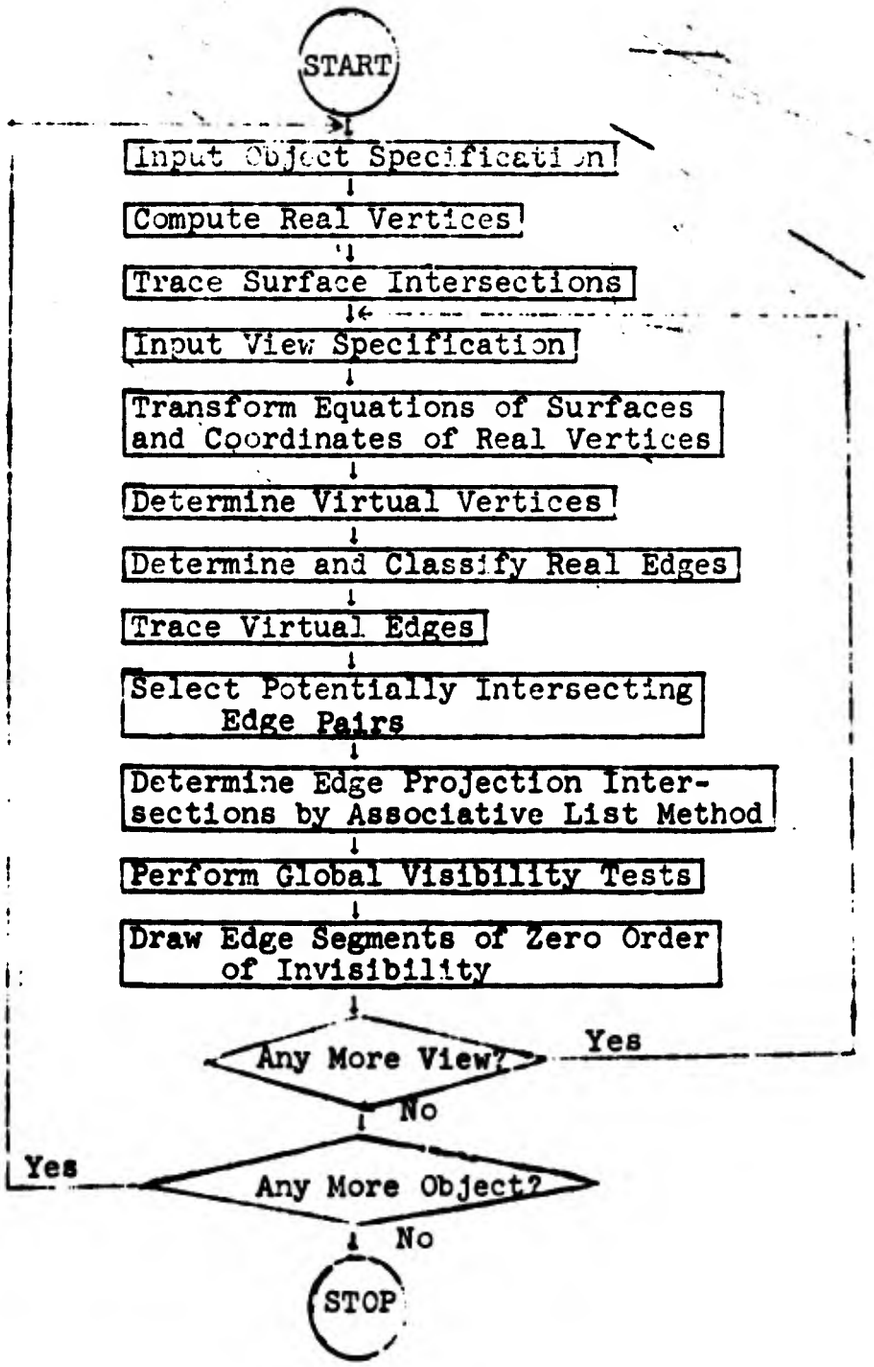


FIGURE 10

QUADRAW MAIN PROGRAM FLOWCHART

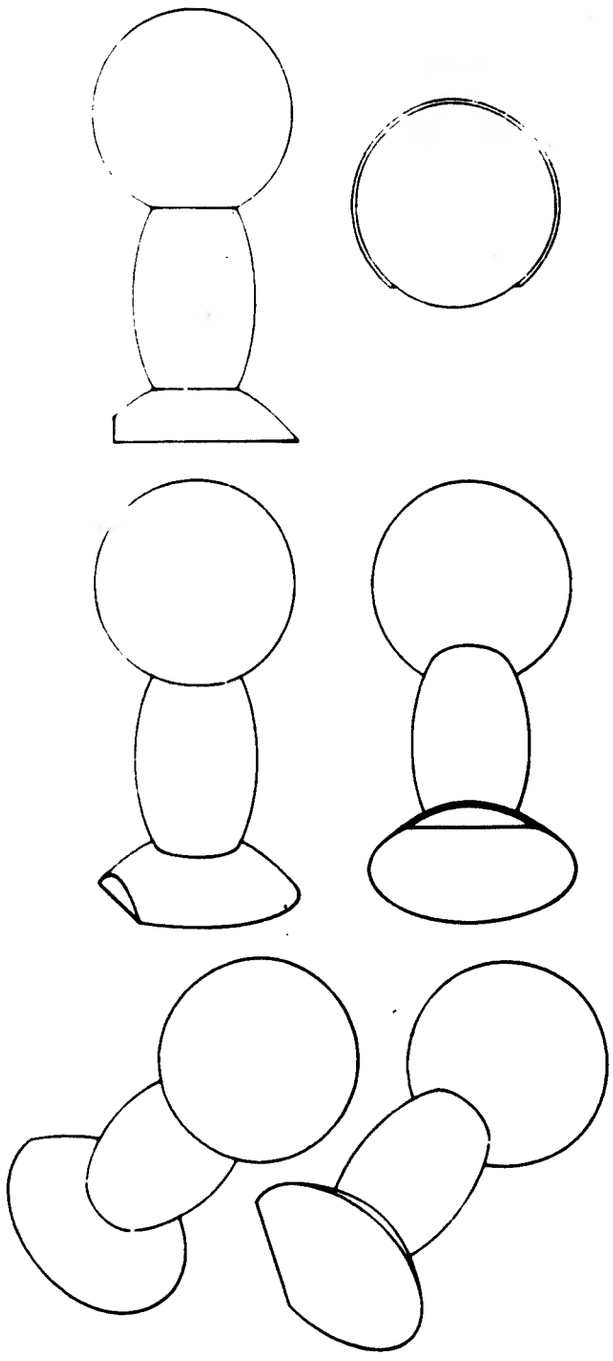


FIGURE 11

TEST OBJECT

for such a simple object, it would be quite a task to specify a planar approximation of the object.

The normal views of a spacecraft are shown in Fig. 12. Auxiliary views are shown in Fig. 13. Although QUADRAW in its present form is not easy to use, it took only a few hours (total elapsed time) of interaction between one person and a batch-processing computer to attain the final shape of the spacecraft. If a planar approximation method had been used, it would have been very difficult to make the many changes in the object specification. The midsection of the spacecraft is a circular cylinder. It is joined at one end with a nose-cone capped by a paraboloid of revolution, and at the other end with a tail section consisting of concatenated sections of three paraboloids of revolution. The body of the spacecraft is actually a thick shell with a set of inside surfaces back-to-back with the outside surfaces. Two elliptic cylinders are poked through the cone to make two windows. Protruding from the side is an antenna made up of two cylindrical rods supporting a spherical dish. (Note that spheres are used there instead of paraboloids for variety rather than reality). As an element of surprise, a small cube floats inside the spacecraft which can be peeked at from certain angles. To give an appreciation of the complexity of this object, two views of it with the hidden lines shown are given in Fig. 14. In Fig. 14a, a "wire-frame" view is shown, and in Fig. 14b, the hidden lines appear dashed.

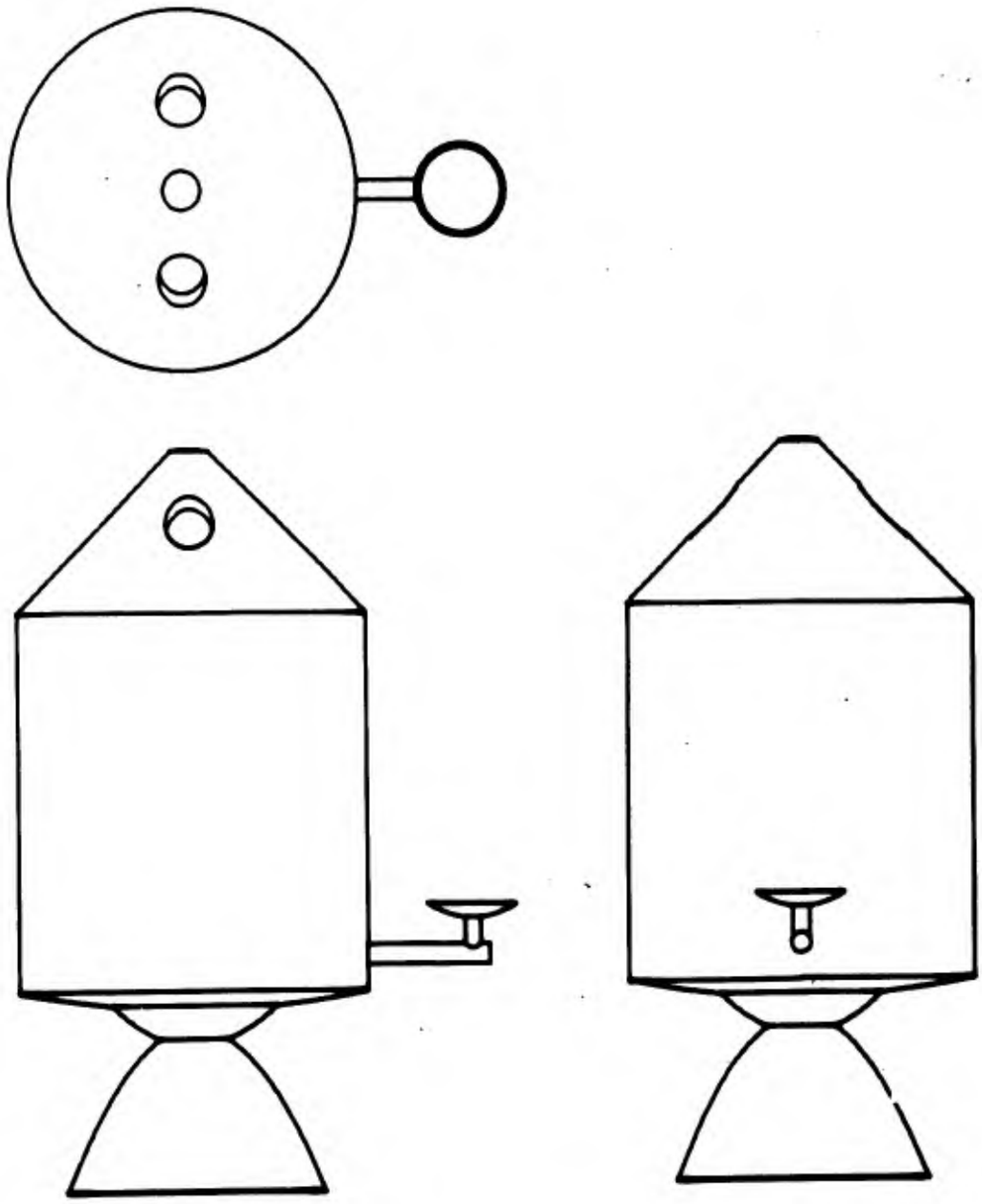


FIGURE 12

NORMAL VIEWS OF A SPACECRAFT

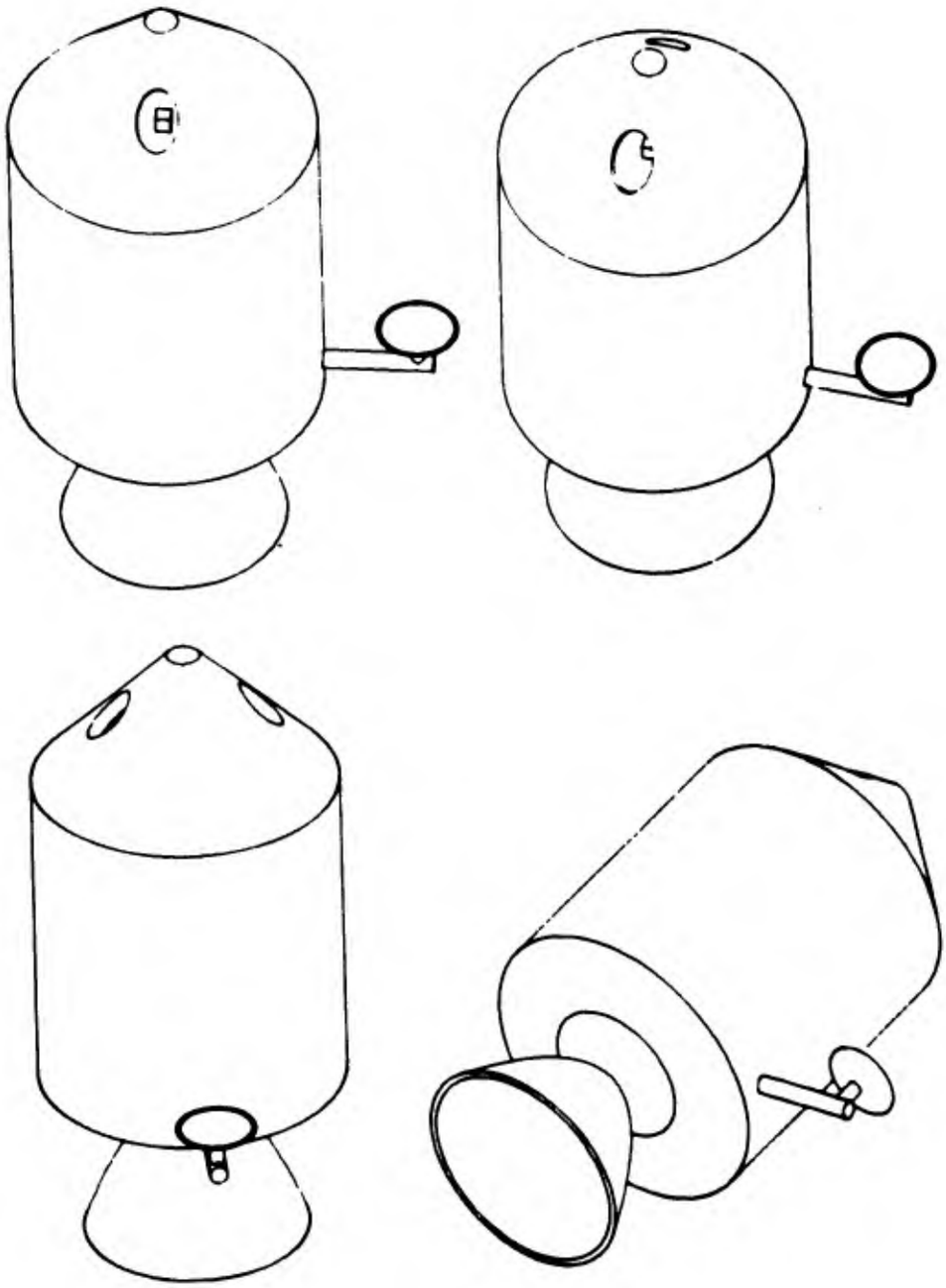
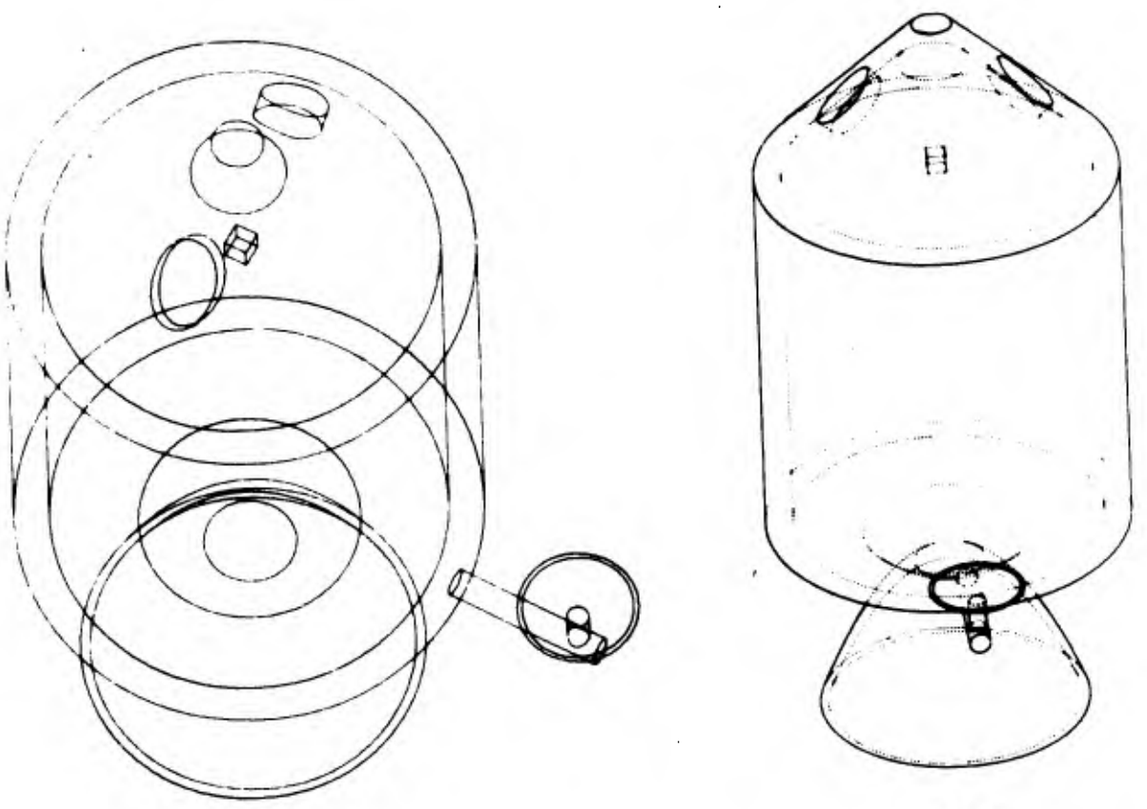


FIGURE 13

MORE VIEWS OF THE SPACECRAFT



(a)

WIRE-FRAME

(b)

HIDDEN LINES DASHED

FIGURE 14

VIEWS OF THE SPACECRAFT WITH THE HIDDEN LINES SHOWN

Two stacks of spheres are shown in Fig. 15, and three other simple objects are shown in Fig. 16.

Some 360/91 execution times of QUADRAW are tabulated in Fig. 17. It is clear that per-view execution time varies with view complexity, which may be measured by the number of edge projection intersections. To get some idea of how execution time varies with object complexity, one measure of which may be the number of component surfaces of the object, a succession of simplified versions of the spacecraft were drawn, each with fewer parts than the preceding one. First, the small cube inside was put away. Then the two windows and the inside surfaces were eliminated. Then the spacecraft was stripped of its antenna. Finally, the tail section was taken off. The table also contains the execution times for stacks of 3, 9, ..., 27 spheres. For these two particular sets of objects (spacecrafts and spheres), per-view execution time increases approximately linearly with the number of component surfaces for a given view. However, we cannot establish from these times a meaningful figure for the rate of variation of execution time with object complexity since it depends very much on the object configurations considered.

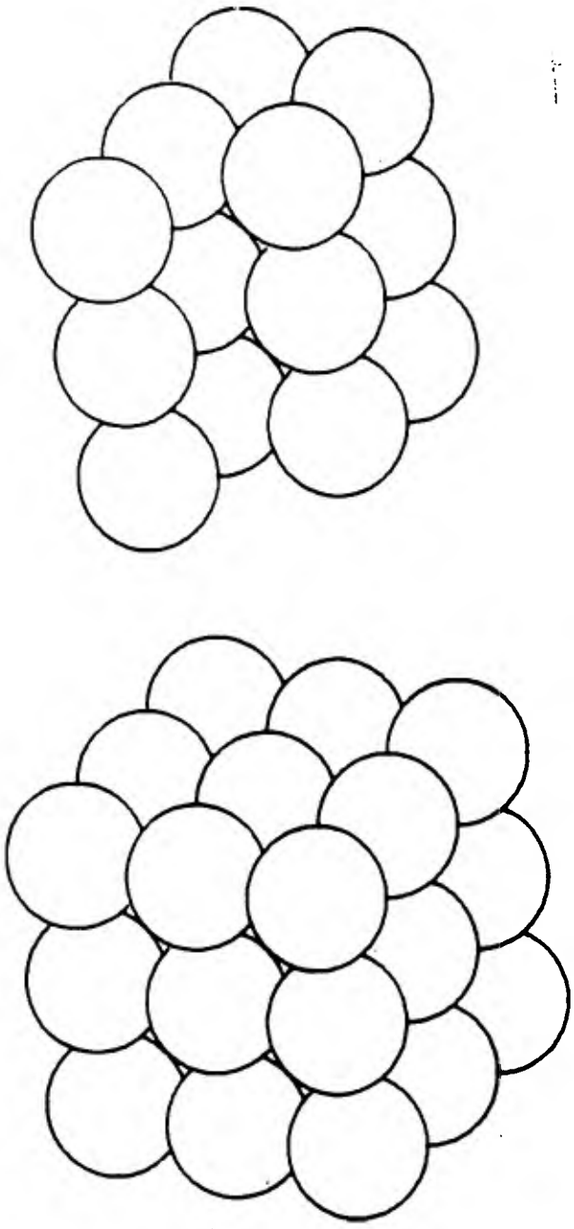


FIGURE 15

STACKS OF SPHERES

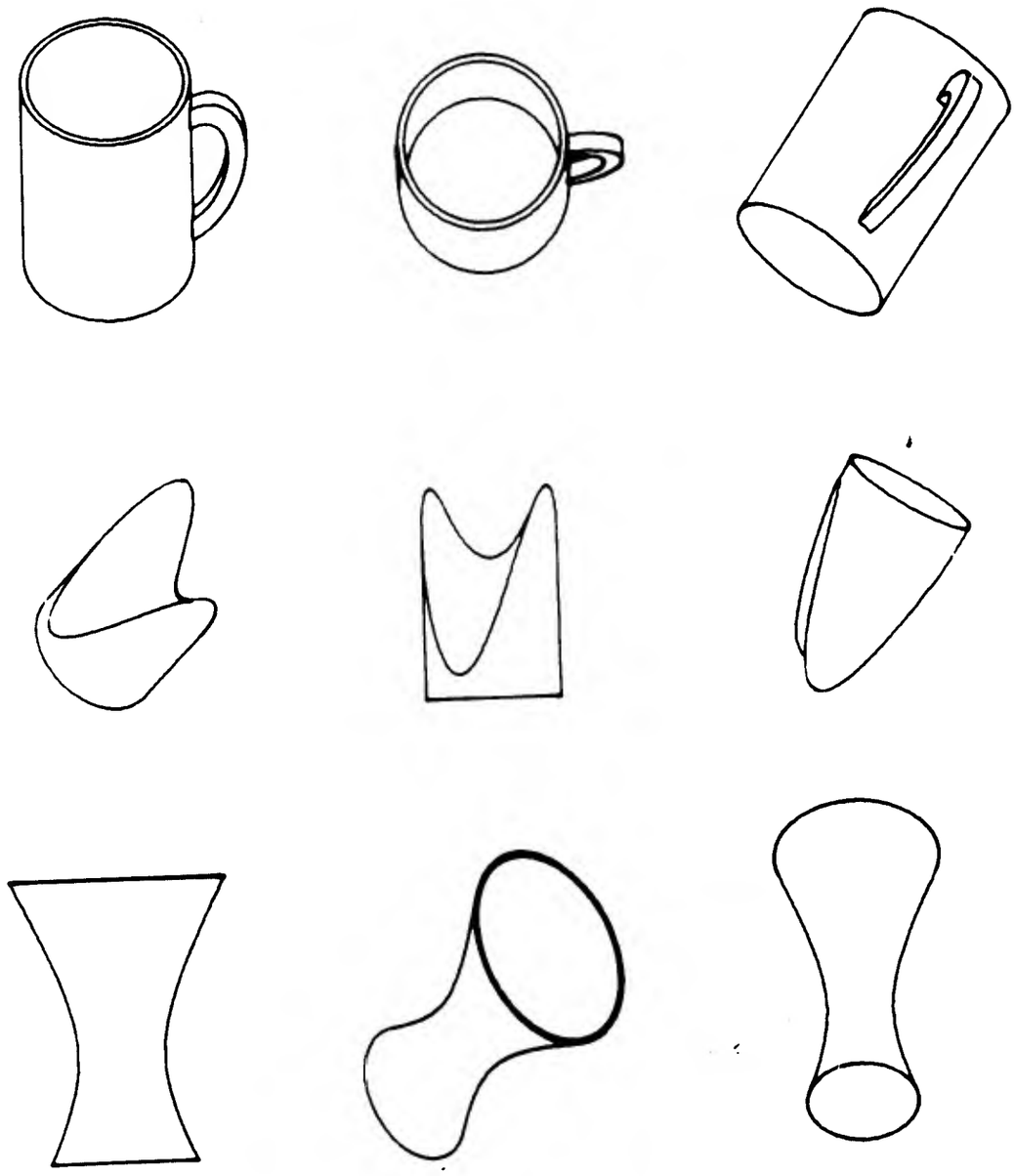


FIGURE 16

THREE SIMPLE OBJECTS

OBJECT	NO. OF SURF.	VIEW ANG. IN DEG.	PER-OBJ. TIME IN SEC.	PER-VIEW TIME IN SEC.
Test Object	5	0, 0, 0 45,45,45	.35	.30 .36
Full Spacecraft	27	0, 0, 0 5,42, 0	.92	.91 1.04
Cube Deleted	21	0, 0, 0 5,42, 0	.80	.85 .96
& Windows Deleted	14	0, 0, 0 5,42, 0	.61	.63 .68
& Antenna Deleted	8	0, 0, 0 5,42, 0	.50	.51 .53
& Tail Deleted	4	0, 0, 0 5,42, 0	.28	.26 .27
27 Spheres	27	0, 0, 0 45,45,45	.21	1.44 1.45
24 Spheres	24	0, 0, 0 45,45,45	.21	1.27 1.33
.	21	0, 0, 0 45,45,45	.13	1.10 1.10
.	18	0, 0, 0 45,45,45	.13	.92 .96
.	15	0, 0, 0 45,45,45	.11	.70 .76
and so on	12	0, 0, 0 45,45,45	.11	.50 .57
	9	0, 0, 0 45,45,45	.09	.27 .42
	6	0, 0, 0 45,45,45	.09	.19 .27
	3	0, 0, 0 45,45,45	.07	.09 .12

FIGURE 17

A TABLE OF QUADRAW EXECUTION TIMES

VIII. EXTENSIONS

Some extensions of this research work are suggested in this section. They are well-defined projects that can be pursued immediately, using QUADRAW as the base. The perspective drawing capability only needs implementation, whereas the others call for more research.

8.1 Perspective Projection

Fig. 18 illustrates how P'_k, the perspective projection of a point P_k, can be obtained. A rotational transformation of the coordinates of P_k (O-X"Y"Z" to O-XYZ) is first performed as in orthographic projection. Besides the three angles, two more parameters are needed to specify a perspective view: the distance from Q to the origin of the object reference frame, denoted by D, and the distance of π from Q, denoted by d. Q is now the point (D,0,0), and π is the plane x=D-d, both referred to O-XYZ. We can choose the picture reference frame O'-Y'Z', such that the Y' and the Z' axes are parallel to the Y and the Z axes respectively, and O' is the point (D-d,0,0) referred to O-XYZ. By similar triangles, the coordinates of P'_k referred to O'-Y'Z' are given by

$$\frac{y'_k}{y_k} = \frac{z'_k}{z_k} = \frac{d}{D-x_k} \tag{8.1}$$

The equation (2.8) can also represent the polar plane for perspective projection, but now

$$\vec{u} = \frac{\vec{PQ}}{|\vec{PQ}|} = \frac{1}{|\vec{PQ}|}\,[(D-x)\vec{u}_x - y\vec{u}_y - z\vec{u}_z]. \tag{8.2}$$

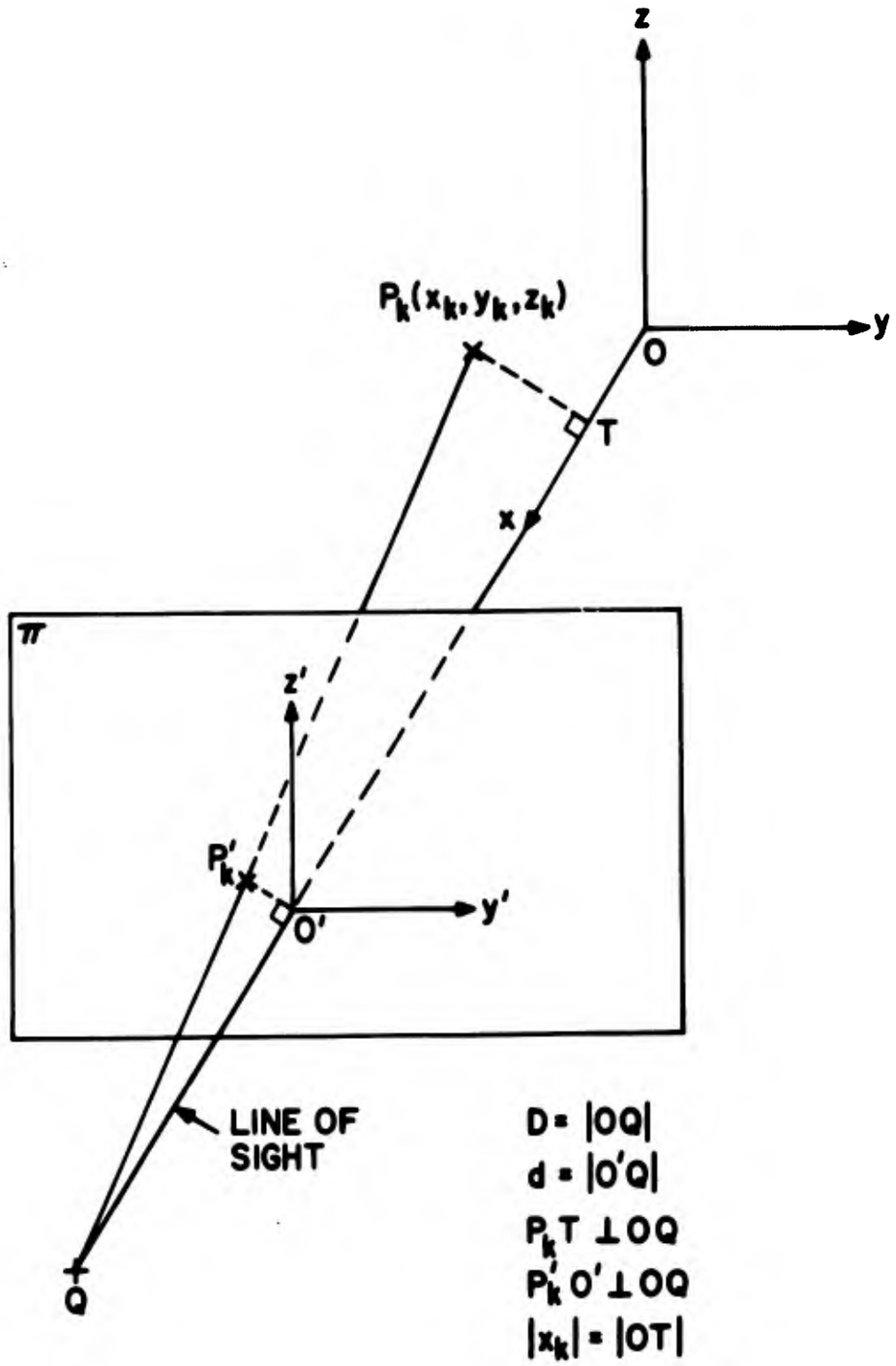


FIGURE 18

PERSPECTIVE PROJECTION OF P_k

For the global visibility test, a subroutine is required to find the point of intersection between a surface $q_r = 0$ and P_kQ, where P_k is a point on the object, or $W'_\tau Q$, where W'_τ is an edge projection intersection in π.

8.2 Shading

The associative list method described in Sec. 6.1 can be used in an algorithm for shading a line drawing of a quadric object. After all the visible curve segments have been determined, the points on these segments are all linked together to the associative store. A nonempty list of points linked to the k^{th} cell of the associative store represents the sequence of intersections of a left-to-right scan-line at z=k with the projections of the visible surface boundaries. At every such intersection, we can determine whether the scan-line is entering or exiting from the projection of a front face and begin a new set of shading calculations.

Let $P'_{i,1}$ denote the ith intersection on a scan-line z=k, and let it be a point of entry into the projection of a face on S_r. For each point $P'_{i,j}$ on z=k such that $y'_{i,j}=y'_{i,j-1} + \delta_P$, δ_P being the picture resolution, and $y'_{i,j} < y'_{i+1,1}$, we can solve for $x_{i,j}$ in $q_r(x_{i,j}, y'_{i,j}, k)=0$ to obtain $P_{i,j}=(x_{i,j}, y'_{i,j}, k)$. Then an intensity function, $I(P_{i,j})$, is evaluated. On a CRT display console with intensity variation capability, $I(P_{i,j})$ gives the brightness of a displayed light spot. On a device

such as the SC 4020, $I(P_{i,j})$ gives the greyness level. Depending on the nature of the drawing device and the desired picture contrast, $I(P_{i,j})$ is usually chosen to be an increasing function of the cosine of the angle of incidence of $QP_{i,j}$ at $P_{i,j}$, that is, $\vec{u}_x \cdot \text{grad } q_r(P_{i,j})$, and a decreasing function of distance from Q [26]. Since intensity calculation is performed at a great nember of points, it must be as efficient as possible. Because δ_p is small, $x_{i,j}$ can be determined without solving any equation by trying small increments or decrements on $x_{i,j-1}$ and substituting into $q_r=0$. $I(P_{i,j})$ may also be evaluated incrementally. On many drawing devices, the intensity or greyness resolution is not very fine; therefore, a few points may be skipped between successive intensity calculations.

In Fig. 19, three scan-lines are shown crossing the projections of front faces F_1, F_2, F_3 and F_4. The visible edge segments in the borders of these faces are labelled by lower-case letters whereas their intersections with the scan-lines are labelled by upper-case letters. b is a front edge; a,c, and d are boundary edges; e and f are segments of boundary edges. At A_k on the scan-line at z=k, intensity calculations based on F_1 begin. At B_k, intensity calculations are switched to F_2. Complications arise at an exit from a face projection on a boundary edge. Between C_k and D_k, for instance, the scan-line is in the projection of some face whose border has not yet been

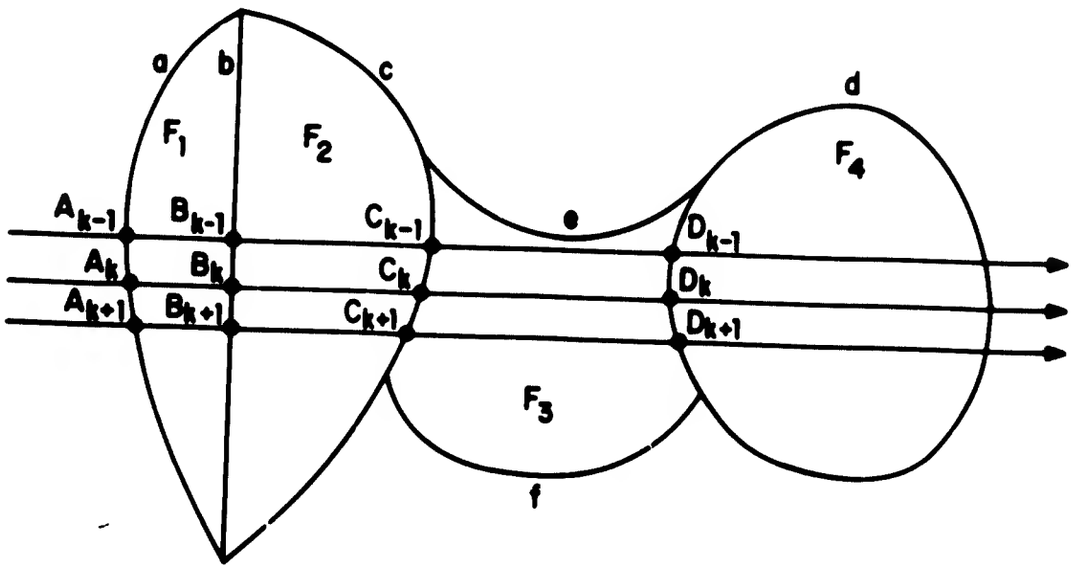


FIGURE 19

SCANNING A LINE DRAWING FOR SHADING

crossed by this scan-line. Therefore, a visibility test must be performed to determine what is the face visible from Q between C_k and D_k - a time-consuming process. A method is needed to take advantage of the information gained from the preceding scan in order to minimize the number of faces that must be examined. This is the "crux" of the problem.

8.3 Higher-order Surfaces

The procedure for drawing quadric objects can be generalized, in theory, to higher-order algebraic surfaces or even to other classes of curved surfaces. The algorithm for tracing surface intersections is applicable to any surfaces, provided that subroutines for evaluating q_r, $\partial q_r/\partial x$, etc. are supplied. The associative list method for finding chain intersections is obviously independent of what kinds of curves are represented by the chains. The object characterization presented in Sec. III can be generalized to higher-order surfaces. (2.8), in general, can represent any surface besides a plane. For actual implementation, however, there are problems. For example, tracing a surface intersection is simple except that we need a starting point on the curve, which is not easy to get for higher-order surfaces. Finding the intersection of a straight line with a higher-order surface is much more difficult than it is for quadric surfaces.

It seems that generalization of QUADRAW to handle arbitrary higher-order surfaces would not be practical because of computation and object specification difficulties. (One should probably resort to Coons' surfaces [8] for modeling objects with very complicated surfaces and use planar approximation for hidden-line elimination). Nevertheless, some surfaces especially useful for design, such as a toroid, may be admitted by adding the necessary subroutines to QUADRAW.

8.4 A 3-D CRT Sketchpad

It is inherently difficult for most of us to translate a mental conception of a solid into mathematical terms. A computer-aided designer of solids should not have to have a working knowledge of solid analytical geometry. An interactive system that enables a user to construct a 3-D object from components on a CRT display console will be an exceedingly valuable design tool.

A "menu" of geometric components formed of sections of quadric surfaces may be displayed on the side of a CRT screen. Each component is shown as a wire frame composed of a few suitably chosen surface curves. By means of a light-pen, the user may select any one of the components, move it to the center of the screen and start manipulating it. With the aid of input devices such as a "joystick" and function keys, the user may turn the component around in space, or cause it to change its

size and shape. For example, he may make an ellipsoid grow longer or fatter. After he is satisfied with the look of the component, the user can work on another component and then join the two together by manipulating them in different views. If he pushes a particular function key, the intersection of the two components will appear. If a component is cut into several sections by other components, the light-pen can be used to eliminate the unwanted sections. Thus, component by component, an object of complicated shape can be constructed. The pressing of a "hidden-line" key will change the wire-frame representation into a rendering free from hidden lines. While the "sketching" is going on, the data structure containing the object description is constantly being updated. When the hidden-line key is pressed, the final description is used to execute QUADRAW. To give a reasonably fast response to the designer, QUADRAW requires a powerful computer; yet it would be too costly for this computer to interact directly with the designer. Therefore, the 3-D sketchpad just described is an ideal application for a system consisting of a time-shared large computer coupled to a "graphics terminal" - a small computer controlling a CRT console. The large computer will do the heavy computation on demand while the graphics terminal will play the interactive part. If a camera or video tape recorder is attached to such a system, the 3-D Sketchpad will be all ready for 3-D animation.

IX. CONCLUSION

A view-dependent characterization of a quadric object has made it possible to develop a hidden-line determination method that is analogous to one that has been successfully applied to polyhedral objects. Employing a combination of analytical and piecewise-linear techniques, a complete procedure for drawing visible-line projections of quadric objects has been developed. Some of the algorithms embodied in this procedure, such as those for computing surface intersections and finding chain intersections, may have applications beyond computer drawing. The procedure is implemented in a FORTRAN program whose capability has been demonstrated by many examples.

Current research on computer-drawing algorithms leans heavily towards the planar-approximation and sample-space approach (see Sec. 1.3), often relying on special-purpose hardware to speed up picture generation. Although shading adds realism to a picture, a line drawing is still the most efficient form of rendering that can be produced on any ordinary plotting device or CRT screen. Without hardware aids, the procedure presented can efficiently generate line drawings of higher quality than that can be produced by any planar-approximation method. The procedure is potentially extendible to handle shading and higher-order surfaces.

X. BIBLIOGRAPHY

1. Appel, A., "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," Proc. ACM 22nd Nat. Conf., ACM Pub. P-67, Thompson Book Co., Washington, D.C., pp. 387-393, 1967.

2. Appel, A., "Some Techniques for Shading Machine Rendering of Solids," AFIPS Conf. Proc. 32, pp. 37-45, 1968.

3. Bouknight, W.J., "A Procedure for Generation of Three-dimensional Half-toned Computer Graphics Presentations," Comm. ACM 13, pp. 527-536, September 1970.

4. Burbaum, W., "Visual Simulations: Computer vs. Conventional," G. E. News Bureau Release 3439-967-396, October 1967.

5. Comba, P.G., "A Language for Three-Dimensional Geometry," IBM Systems Journal 7, 3, 4, 1968.

6. Comba, P.G., "A Procedure for Detecting Intersections of Three-dimensional Objects," J. ACM 15, 3, pp. 354-366, July 1968.

7. Coons, S.A. "An Outline of the Requirements for a Computer-aided Design System," AFIPS Conf. Proc. 23, pp. 299-304, 1963.

8. Coons, S.A., "Surfaces for Computer-aided Design of Space Forms," MAC-TR-41, Clearing House for Federal Scientific and Technical Information, Springfield, Virginia, 1967.

9. Davis, J.R., Nagel, R., and Guber, W., "A Model Making and Display Technique for 3-D Pictures," Private report, Mathematical Applications Group, MAGI, INC.

10. Dresden, A., Solid Analytical Geometry and Determinants, Dover Publications, Inc., New York.

11. Elson, B., "Color TV Generated by Computer to Evaluate Spaceborne Systems," Aviation Week and Space Technology, October 1967.

12. Freeman, H., "On the Encoding of Arbitrary Geometric Configurations," IRE Trans. Electron, Comp. EC-10, 2, June 1961.

13. Freeman, H. and Woon, P., "An Algorithm for the Incremental Generation of Optimal Curve Approximations on a Square Grid," Unpublished report, Department of Electrical Engineering, New York University, New York, New York, 1967.

14. Galimberti, R., and Montanari, U., "An Algorithm for Hidden Line Elimination," Comm. ACM 12, pp. 206-211, April 1969.

15. IBM Corp., "System/360 Scientific Subroutine Package (360A-CM-03X) Version III," Programmer's Manual, Form No. H20-0205-3, pp. 181-197, 1968.

16. IBM Corp., "SC 4020 Reference Information," Computing Center Newsletter, T.J. Watson Research Center, Yorktown Heights, New York, 1970.

17. Kelley, K.C., "A Computer Graphics Program for the Generation of Half-tone Images with Shadows," Report R-444, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois, November 1969.

18. Kubert, B., Szabo, J., and Giulieri, S., "The Perspective Representation of Functions of Two Variables," J. ACM 15, 2, pp. 193-204, April 1968.

19. Loutrel, P., "A Solution to the Hidden-line Problem for Computer-Drawn Polyhedra," IEEE Trans. Computers, C-19, (3), March 1970, pp. 205-213.

20. Luh, J.Y.S., and Krolak, R.J., "A Mathematical Model for Mechanical Part Description," Comm. ACM 8, pp. 125-129, February 1965.

21. McIlroy, M.D., "A Language for Drawing Pictures by Computer," Private communication, Bell Telephone Laboratories, Murray Hill, New Jersey.

22. Roberts, L.G., "Machine Perception of Three Dimensional Solids," Tech. Rep. No. 315, Lincoln Lab., MIT, Cambridge, Massachusetts, 1963.

23. Salmon, G., Modern Higher Algebra, Metcalfe and Son, 1885.

24. Sommerville, D.M.Y., Analytical Geometry of Three Dimensions, Cambridge University Press, 1959.

25. Warnock, J.E., "A Hidden Line Algorithm for Halftone Picture Representation," Tech. Rep. No. 4-5, University of Utah, Salt Lake City, Utah.

26. Warnock, J.E., "A Hidden Surface Algorithm for Computer Generated Halftone Pictures," Tech. Rep. No. 4-15, University of Utah, Salt Lake City, Utah, June 1969.

27. Watkins, G.S., "A Real-time Visible Surface Algorithm," Tech. Rep. No. UTECH-CSC-70-101, University of Utah, Salt Lake City, Utah, June 1970.

28. Weiss, R.A., "BE VISION, A Package of IBM 7090 FORTRAN Programs to Draw Orthographic Views of Combinations of Planes and Quadric Surfaces," J. ACM 13, 2, pp. 194-204, April 1966.

29. Woon, P.Y., "On the Computer Drawing of Solid Objects Bounded by Quadric Surfaces," Tech. Rep. 403-3, Department of Electrical Engineering, New York University, New York June 1969.

30. Wylie, C., Romney, G., Evans, D. and Erdahl, A., "Half-tone Perspective Drawings by Computer," Tech. Rep. No. 4-2, University of Utah, Salt Lake City, Utah, 1967.

31. Zajac, E.E., "Computer-made Perspective Movies as a Scientific and Communications Tool," Comm. ACM 7, pp. 169-170, March 1964.