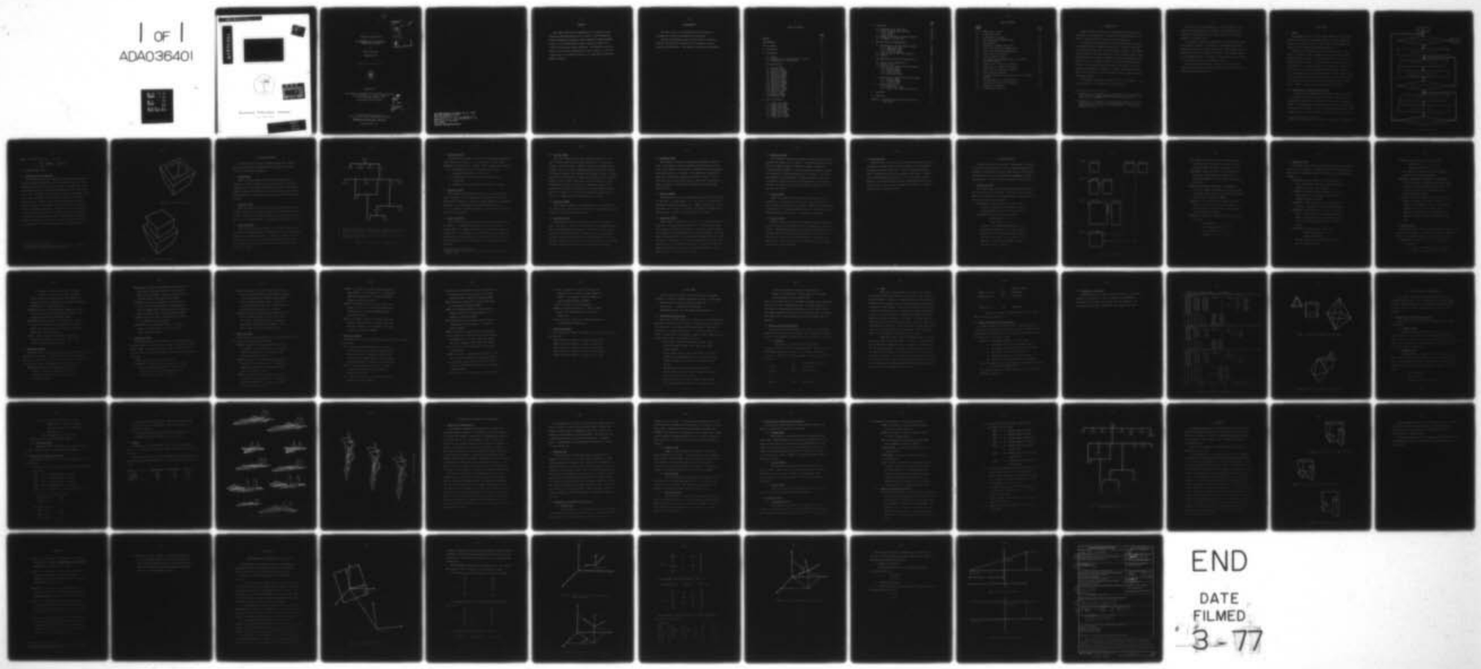
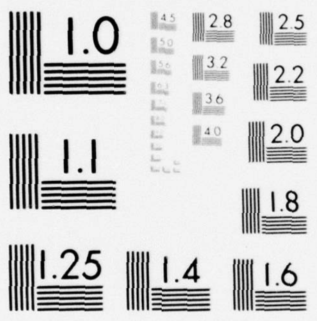
AD-A036 401 RENSSELAER POLYTECHNIC INST TROY N Y COMPUTER RESEAR--ETC F/G 9/2
AN IMPLEMENTATION OF THE LOUTREL HIDDEN-LINE ALGORITHM.(U)
SEP 76 M POTMESIL AF-AFOSR-2937-76
UNCLASSIFIED CRL-49 AFOSR-TR-77-0104 NL

1 OF 1
ADA036401



END
DATE
FILMED
3-77



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

(4

Technical Report CRL-49

AN IMPLEMENTATION OF THE LOUTREL HIDDEN-LINE ALGORITHM

by

Michael Potmesil

September 1976

Prepared for

Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research
Air Force Systems Command, USAF
Grant Number AFOSR 76-2937

(See form 1473)

Computer Research Laboratory
Electrical and Systems Engineering Department
Rensselaer Polytechnic Institute
TROY, NEW YORK 12181

ABSTRACT

This report describes an implementation of a variation of the hidden-line algorithm developed by P. Loutrel. The new algorithm produces hidden-line drawings of scenes composed of opaque polyhedra as well as of some more general types of flat-faced objects not covered by the original Loutrel algorithm. The program is written in Fortran IV and has been implemented in a batch version on a CDC 6600 computer and in an interactive version on an ADAGE AGT-30 interactive graphics computer.

ACKNOWLEDGMENT

The author wishes to thank Professor Herbert Freeman for providing the direction and guidance for this project.

The work was supported by the Air Force Office of Scientific Research, Directorate of Mathematical and Information Sciences, under grant AFOSR-76-2937. This support is gratefully acknowledged.

TABLE OF CONTENTS

1-035-315
A-036-391

LIST OF FIGURES

1. INTRODUCTION

Given a computer description of a three-dimensional scene and the coordinates of a vantage point, a hidden-line algorithm determines the parts of the scene which are visible from the vantage point.

In this report an implementation of such an algorithm is described. The algorithm is that originally developed by P. Loutrel and described in a journal article* as well as in his doctoral thesis**. The original algorithm accepted only true polyhedral objects, either convex or concave and either simply or multiply connected. The algorithm implemented here extends the set of acceptable objects to all objects bounded by planar surfaces. This includes objects whose faces may contain one or more holes (either intruding or extruding). However, the extended algorithm cannot handle non-closed objects or objects bounded by quadric or other non-planar surfaces.

The implemented program called PDRAW is written in FORTRAN IV. There are two versions of the program: batch and interactive.

---

* Loutrel, P. P., "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," IEEE Transactions on Computers, Vol. C-19, No. 3, March 1970, pp. 205-213.

** Loutrel, P. P., "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," doctoral dissertation, Department of Electrical Engineering, New York University, September 1967. (NTIS accession number: N68 13 830).

The batch version was implemented on a CDC 6600 computer. This version of the program reads all commands, object descriptions and vantage point specifications from an input data file and produces drawings on a digital plotter.

The interactive version was implemented on an ADAGE AGT-30 interactive graphics computer. This version of the program reads object descriptions from data files but program control and vantage points are specified by interactive graphics I/O devices. The drawings produced by the program are made on a CRT screen and hard copies can be obtained on a dot matrix plotter or an incremental plotter.

During the last several years a number of other hidden-line or hidden-surface algorithms have been developed. An extensive survey of these algorithms is given in [3].

2. THE PROGRAM

2.1 Review

The program consists of a main program, PDRAW, and a set of subroutines and functions. The program requires a plotting package for drawing on a digital plotter or a similar device. The program uses a system routine to measure the elapsed execution time.

The input to the program consists of a set of "scenes". Each scene is followed by a set of vantage points from which the program makes drawings of the scene. A scene consists of one or more objects bounded by planar surfaces. All the objects in a scene must be non-intersecting. All the vantage points for a scene must be outside of all the objects in the scene*. PDRAW produces three types of drawings: (1) all the edges of all the objects are drawn as solid lines, (2) only the visible edges are drawn as solid lines, and (3) the visible edges are drawn as solid lines and the invisible edges are drawn as dashed lines. Figure 2.1 gives the flow of control of the program.

2.2 Computation of a Normal Vector to a Face

A change from the original Loutrel version was made in the method of computing outward normal vectors to all faces. These vectors are vantage-point independent, computed once for an object in subroutine PRECOM. The original method devised by Loutrel failed for certain faces such as those in the shape of an "H". The new program computes a normal outward vector to a face as follows: Assume that a face has N vertices

* Two types of projections are available: perspective and orthographic.

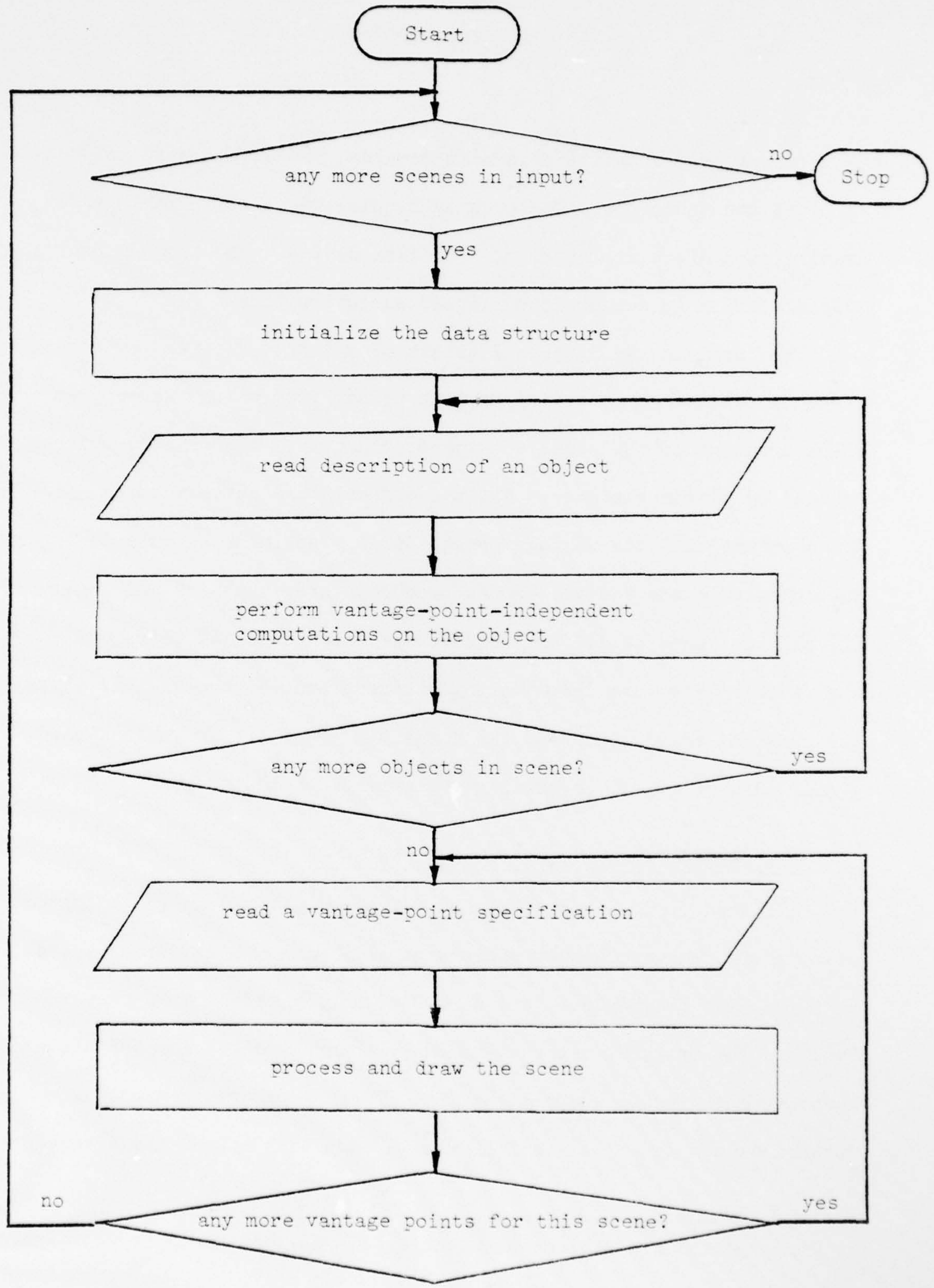

Start
any more scenes in input?
no
Stop
yes
initialize the data structure
read description of an object
perform vantage-point-independent computations on the object
any more objects in scene?
yes
no
read a vantage-point specification
process and draw the scene
any more vantage points for this scene?
no
yes

Fig. 2.1 PDRAW flow of control

whose coordinates are V_1, V_2, ... V_N then

$$\vec{n} = \sum_{j=2}^{N-1} (\overrightarrow{V_j - V_1}) \times (\overrightarrow{V_{j+1} - V_1})$$

is an outward normal vector.

2.3 An Extension of the Algorithm

The original algorithm was extended to accept objects that have one or more face-interior holes [Fig. 2.2] or protrusion [Fig. 2.3]. When an object has a face-interior hole or protrusion, the face where the hole or protrusion occurs will be multiply connected; that is, it will have a hole in it. We shall refer to a hole in a face as an "antiface". Whenever a vertex is behind a visible face, the order of invisibility is increased by one. However, when a vertex lies behind an antiface, the order of invisibility is decreased by one. This is done because each antiface indicating the presence of a face-interior hole or protrusion is inside a larger true face*. For the face the order of invisibility is increased; for the antiface the order of invisibility is decreased. Therefore, the order of invisibility of a vertex inside an antiface does not change after both the face and the antiface have been processed.

* An antiface is thus bounded by a chain of edges that is disjoint from the chain bounding the containing face.

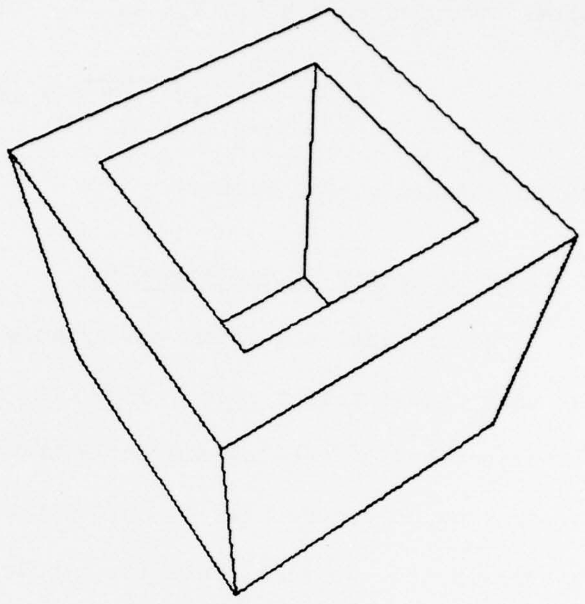


Figure 2.2 A cube with a hole.

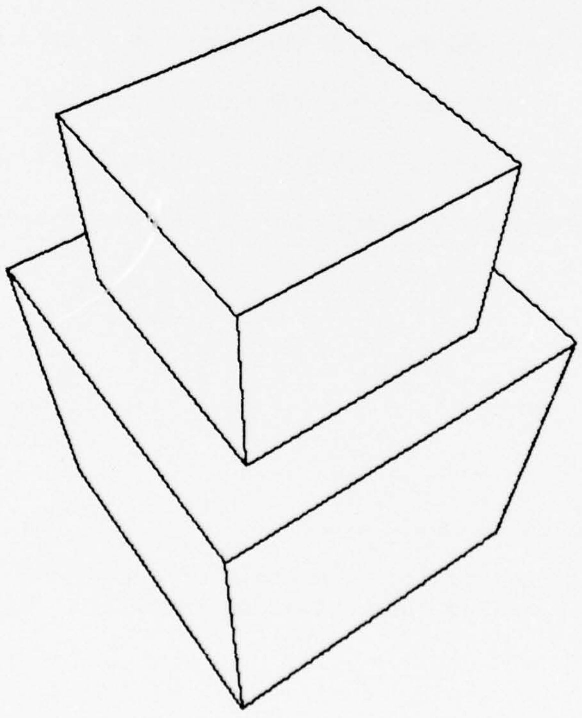


Figure 2.3 A cube with a protrusion

3. THE PDRAW ROUTINES

In this section all routines in PDRAW are described. PDRAW consists of a main program, 9 subroutines and 4 functions. The chart of Fig. 3.1 gives their connections.

3.1 Program PDRAW

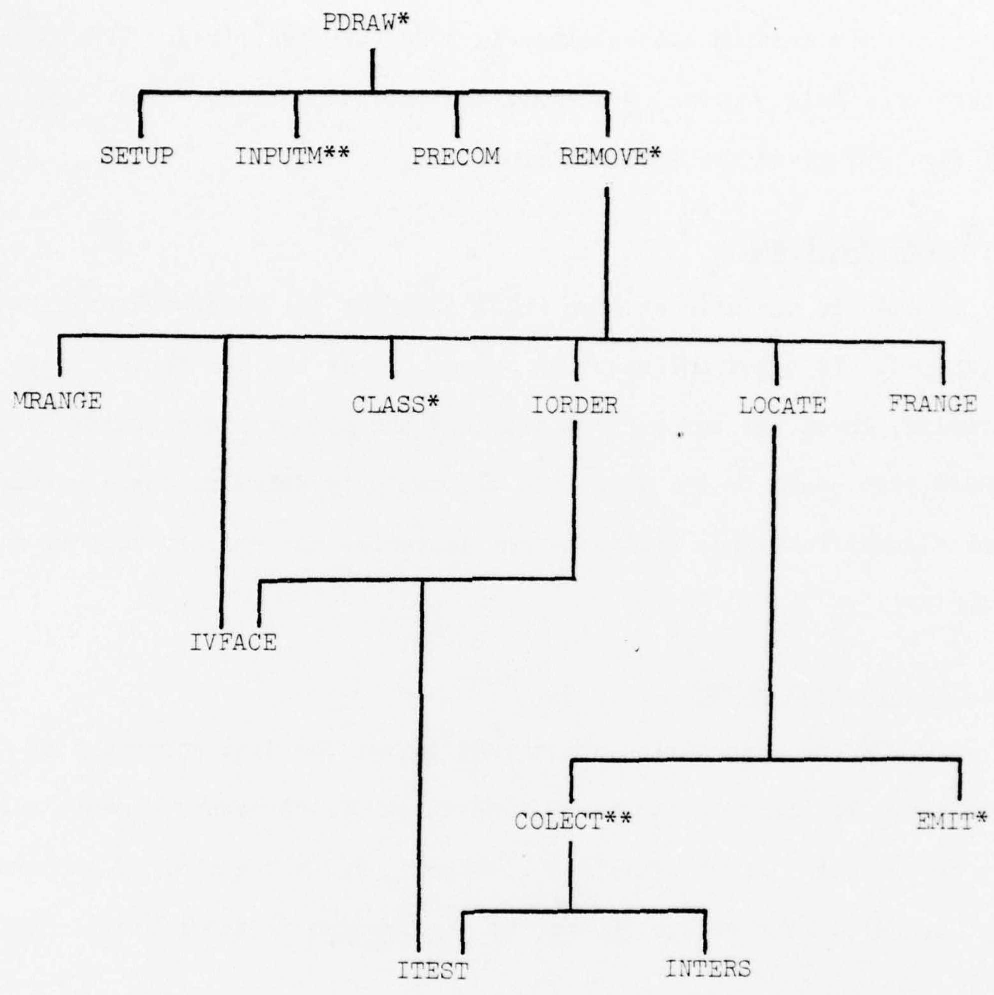
PDRAW is the main program which controls the execution of all routines. It opens the plotting output, moves the pen to the first drawing, moves the pen between drawings and closes the plotting output. PDRAW also reads from a data file all the data delimiters and vantage point specifications, analyzes them and calls the main functional subroutines.

3.2 Subroutine SETUP

SETUP is a subroutine which initializes the data structure of the program. Pointers to the various sections of the data structure are set to point to the first available locations, and the number of objects is set to zero. The data structure of the program is described in Section 4.

3.3 Subroutine INPUTM

This subroutine reads the description of an object from the input data file to the data structure of the program. The format of an object description is described in Section 5. Whenever an attempt is made to read more data into the data structure than the allowable maximum, an error message is written, the plotting output is closed and program execution is stopped.



* Calls a plotting routine to draw a vector. PDRAW also calls library routines to open and close the plot output and to measure time.

** Calls a plotting routine to close the plot output in a case of data structure or stack overflow.

Figure 3.1 The connections of PDRAW routines

3.4 Subroutine PRECOM

This subroutine performs the preliminary, vantage-point-independent classification of each object. It is called once for each object immediately following a call to INPUTM. PRECOM computes the following:

a) an outward normal vector to each face of the object (the method is described in Section 2.2)

b) classification of each edge as either type H2 or H3 [1, page 208]

c) classification of the object as either convex or concave.

3.5 Subroutine TRANS

This subroutine transforms all vertices of all objects from the object coordinates to the vantage point coordinates and then to the picture plane coordinates. TRANS uses either perspective or orthographic projection. All picture plane coordinates are scaled down to a (-1.0, 1.0) square for further computations. The transformation process is described in Appendix A.

3.6 Subroutine REMOVE

This is the main subroutine controlling the actual task of hidden-line removal. It computes the visibility of all faces by calls to function IVFACE. All edges and vertices are classified by calling subroutine CLASS. Visible edges in a non-intersecting convex polyhedron* are determined in this subroutine. Each scene is divided into sets of potentially intersecting objects* and passed to subroutine LOCATE for processing.

* Intersections are determined for projections of objects into the picture plane.

3.7 Function IVFACE

This is an integer function which returns the value of +1 if a face is visible from a given point and the value of -1 if the face is invisible from a given point. IVFACE is invoked at two points in the program. First, it is used to determine whether each face is invisible or potentially visible from a vantage point. Second, in order to compute the order of invisibility of a vertex, this function is used to determine whether a face is visible from a vertex, which is identical to determining whether the face is in front of or behind the vertex. This second use of the function, however, uses only faces which were determined to be visible in the first use of the function, since invisible faces do not change the order of invisibility of a vertex.

3.8 Subroutine MRANGE

This subroutine computes an enclosing envelope for each object in the current scene. This is done by searching for the maximum and minimum x and y picture coordinates of each object.

3.9 Subroutine FRANGE

This subroutine computes an enclosing envelope for each visible face of each object. This is done by searching for the maximum and minimum x and y picture plane coordinates of each visible face in a scene. Although the enclosing envelopes of visible faces are not required by the basic algorithm, they are used to speed up substantially the computations of the order of invisibility of vertices as well as the searching for edge intersections performed in subroutines COLECT and IORDER.

3.10 Subroutine CLASS

This subroutine performs the vantage-point-dependent classifications. Each edge in a scene is classified as either type H1, H2, H3 or H4 [1, pp. 208-209]. Each vertex is classified as either a boundary or an inner vertex [1, pp. 210-211]. This subroutine is called each time a set of potentially intersecting objects is found. If it is called with only a single object which is convex, the vertex classification is omitted. This subroutine identifies all invisible edges (type H1 and H2).

3.11 Function IORDER

This function returns as its integer value the order of invisibility [1, page 209] of a vertex passed to it. The order of invisibility is a non-negative integer corresponding to the number of potentially visible faces which are in front of the vertex, i.e. between the vantage point and the vertex. Note that invisible faces are not counted.

3.12 Subroutine LOCATE

REMOVE passes to LOCATE a set of objects in a scene which potentially intersect in the picture plane [1, page 211]. This routine processes in two passes all H3 and H4 type edges. In the first pass all H3 edges are processed; in the second pass all H4 edges are processed. Each edge that is processed has its initial and final coordinates put into a stack. The order of invisibility of the initial vertex of the edge is put into another stack. These stacks are then passed to subroutine COLECT which computes any possible intersection of this edge with any H3 or H4 type edge in the set of objects passed to LOCATE.

3.13 Subroutine COLECT

This subroutine computes all possible intersections that one edge may have with another [1, pp. 209-210]. The initial and final coordinates of an edge are passed to COLECT from LOCATE. COLECT uses functions ITEST and INTERS to find any possible intersections and adds the coordinates of any found intersection into a stack. It also computes whether the processed edge is coming from behind a face or is going behind a face at the point of the intersection. This change in the order of invisibility along the edge, which is either +1 or -1, is also put into a stack.

3.14 Function ITEST

ITEST is an integer function which tests two edges for a valid intersection. It returns the value of +1 if a valid intersection is found; otherwise the value of 0 is returned. ITEST is also used to find an intersection of a line going from a vertex to infinity with an edge during computations of the order of invisibility of a vertex.

3.15 Function INTERS

INTERS is called from COLECT after ITEST found that two edges have a valid intersection and computed the x and y coordinates of the intersection. INTERS determines whether at the point of the intersection, the processed edge is in front of the other edge or not [1, page 210]. If it is behind, it means that the processed edge is either coming from behind a face or is going behind a face and INTERS returns the value of +1, otherwise a zero is returned.

3.16 Subroutine EMIT

This subroutine is passed a stack containing the initial and final coordinates of an edge as well as the coordinates of all intersections this edge has with all other edges. A stack containing the initial order of invisibility as well as the changes in the order of invisibility at each intersection, either +1 or -1, is also passed to EMIT. EMIT sorts out the intersections by their increasing distance from the initial vertex and then causes the segments of the edge to be drawn as either visible or invisible vectors.

4. THE DATA STRUCTURE

In this section the data structure of the program is described. All the data describing objects as well as several parameters used throughout the program are stored in common blocks. The portions of the data structure which are vantage-point independent are marked with asterisks. A chart of the data structure is given in Fig. 4.1.

4.1 Common Block MODEL

This common block contains the data describing the objects that have been entered into the program. All arrays in this block are dimensioned to MAXM which is the maximum number of objects in a scene that can be entered into the program.

MODEL(MAXM)* contains pointers to the first face of an object. The pointer of each object is set by subroutine INPUT just before the object is read in.

MTYPE(MAXM)* contains the type of an object:

convex object	0
concave object	1

The type is set by subroutine PRECOM after the preliminary classification of all edges as either type H2 or H3. If there are no edges of type H2, the type of the object is set to convex (0); if there are one or more type H2 edges, the type of the object is set to concave (1).

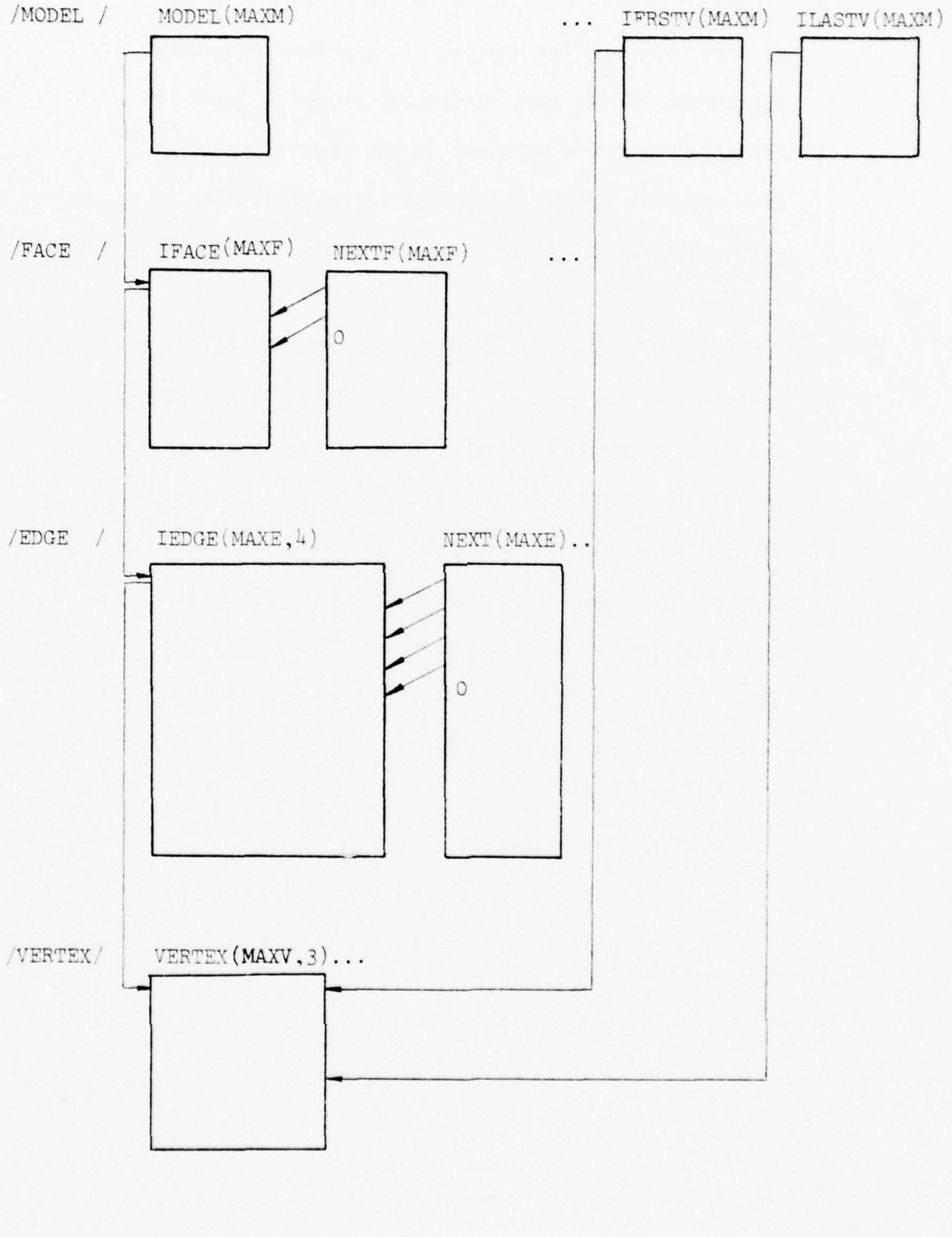


Fig 4.1 Data structure

IFRSTV(MAXM)* contains pointers to the first vertex of of each object. The pointer of an object is set by subroutine INPUTM just before the object is read in.

ILASTV(MAXM)* contains pointers to the last vertex of each object. The pointer of an object is set by subroutine INPUTM after the last vertex of the object was read in.

XMAXM(MAXM),XMINM(MAXM),YMAXM(MAXM) and YMINM(MAXM) contain the maximum and minimum x and y picture plane coordinates of each object. These values are computed for all objects in subroutine MRANGE.

MLIST(MAXM) is a processing status list which contains pointers to a set of objects that have potential intersections in the picture plane. These pointers are set in subroutine REMOVE.

MSTATE(MAXM) contains the processing status of each object in a scene:

not processed	0
add to the processing list ...	1
in the processing list	2
processed	3

4.2 Common Block FACE

This common block contains the data describing the faces of objects entered into the program. All arrays in this block are dimensioned to MAXF, which is the maximum number of faces that can be entered into the program.

IFACE(MAXF)* contains pointers to the first edge of a face. The pointer of each face is set by subroutine INPUTM just before a face is read in.

FNV(MAXF,3)* contains the three components of an outward normal vector of each face. The vector is computed in subroutine PRECOM.

IVPNT(MAXF)* contains pointers to the first vertex of each face. The pointer is set by subroutine PRECOM.

NEXTF(MAXF)* contains pointers to the next face of an object. All pointers are initialized by subroutine SETUP. End-of-list pointers are inserted by subroutine INPUTM after the last face of an object was read in.

IFTYPE(MAXF)* contains the type of each face;

inner face description	0
outer face description	1

The type of each face is read from the input data file by subroutine INPUTM.

ICF(MAXF) indicates the visibility of each face:

invisible face .... -1

potentially visible face ...... +1

The visibility is computed in function IVFACE and stored to ICF by subroutine REMOVE.

XMAXF(MAXF), XMINF(MAXF), YMAXF(MAXF) and YMINF(MAXF) contain the maximum and minimum x and y picture plane coordinates of each potentially visible face. These values are computed for all potentially visible faces by subroutine FRANGE.

ZFNV(MAXF) is used in computing the visibility of faces when the orthographic projection is selected. It contains the dot product of the z row of the transformation matrix with a face normal vector. A negative value indicates an invisible face; a non-negative face value indicates a potentially visible face.

4.3 Common Block EDGE

This common block contains the data describing the edges of objects entered into the program. All arrays in this block are dimensioned to MAXE which allows MAXE/2 edges to be entered into the program.

IEDGE(MAXE,4)* contains the following four pointers for each edge:

1. Pointer to the initial vertex of the edge.
2. Pointer to the final vertex of the edge.

3. Pointer to the face on the right of the edge.

4. Pointer to the face on the left of the edge.

Pointers 1 and 2 are read in by subroutine INPUTM from the input data file. Pointers 3 and 4 are set by subroutine INPUTM after all the data was read it.

IHTYPE(MAXE)* contains the vantage-point-independent classification of each edge as either type H2 or type H3. This classification is done in subroutine PRECOM.

NEXTE(MAXE)* contains pointers to the next edge in a face of an object. All pointers are initialized by subroutine SETUP. End-of-list pointers are inserted by subroutine INPUTM after a face was read in.

IH(MAXE) contains vantage-point-dependent classification of each edge as type H1, H2, H3 or H4. This classification is done by subroutine CLASS.

4.4 Common Block VERTEX

This common block contains the data describing the vertices of objects entered into the program. All arrays in this block are dimensioned to MAXV which allows MAXV - 1 vertices to be entered into the program. The last location is used by the program for a point at "infinity".

VERTEX(MAXV,3)* contains the coordinates of vertices described in the object coordinate system. The coordinates are read in from the input data file by subroutine INPUTM.

VTRANS(MAXV,3) contains the coordinates of vertices in the vantage point coordinate system. The transformation from the object coordinate system to the vantage point coordinate system is done by subroutine TRANS.

V2D(MAXV,2) contains the coordinates of vertices in the picture plane coordinate system. The transformation from the vantage point coordinate system to the picture plane coordinate system is done by subroutine TRANS.

IOV(MAXV) contains the order of invisibility of a vertex. It is computed by function IORDER.

IVTYPE(MAXV) contains the classification of a vertex as either a boundary vertex (non-zero value) or an inner vertex (zero value).

4.5 Common Block STACK

This common block contains information about intersections that an edge may have with any other edge. All arrays in this block are dimensioned to MAXS. This allows each edge to have up to MAXS - 2 intersections, since two locations are used for the initial and final vertices of the edge.

ISTPNT is a pointer to the top of the stack.

STACK(MAXS, 2) contains the picture plane x and y coordinates of all edge intersections. Locations 1 and 2 contain the picture plane coordinates of the initial and final vertices of the edge.

KS(MAXS) contains the increment of the order of invisibility of each intersection. This value may be either +1 or -1 indicating that the edge is going behind a face or coming from behind a face. Locations 1 and 2 of the stack contain the order of invisibility of the initial and final vertices of the edge.

DIST(MAXS) is used for sorting out edge intersections by their increasing distance from the initial vertex of the edge. The sorting is done in subroutine EMIT before the subroutine emits segments of the edge as either visible or invisible vectors.

4.6 Common Block FREE

This common block contains pointers to the next available locations in the blocks MODEL, FACE, EDGE and VERTEX.

IMFREE* is a pointer to the next available location in block MODEL. It is initialized to 1 by subroutine SETUP and incremented by 1 in subroutine INPUTM before a new object is read in.

IFFREE* is a pointer to the next available location in block FACE. It is initialized to 1 by subroutine SETUP and incremented by 1 in subroutine INPUTM before a new face is read in.

IEFREE* is a pointer to the next available location in block EDGE. It is initialized to 1 by subroutine SETUP and incremented by 1 in subroutine INPUTM before a new edge is stored in.

IVFREE* is a pointer to the next available location in block VERTEX. It is initialized to 1 by subroutine SETUP and incremented by 1 in subroutine INPUTM before a new vertex is read in.

NMODEL* is the number of objects in the scene in the program. It is set to 0 by subroutine SETUP and incremented by 1 in subroutine INPUTM before a new object is read in.

NMSET is the number of objects in the scene which have potential intersections in the picture plane and which are currently in the list MLIST. The number is computed in subroutine REMOVE.

4.7 Common Block PARAMS

This common block contains several parameters which are used throughout the program.

VP(3) contains the current vantage point x, y and z coordinates given in the object coordinate system. They are read from a vantage point specification card by the main program PDRAW. In some routines these variables are set equivalent to XØ, YØ and ZØ.

ORTHO is a logical variable which is set to .TRUE. in the main program PDRAW when the orthographic projection is selected.

ZR(3) contains the z row of the transformation matrix, computed in subroutine TRANS.

DD contains the transformation value D, the distance of the vantage point from the origin of the object coordinate system, computed in subroutine TRANS.

ZMAX contains the picture scale of the current image. It is computed in subroutine TRANS.

RADIUS is the maximum distance of a vertex from the origin of the object coordinate system. It is computed in subroutine INPUTM.

EPS1, EPS2, EPS3 are small numbers which are used as tolerances in some computations. All must be positive numbers.

IENPNT is a pointer to the currently processed edge.

N1 is a pointer to the initial vertex of the currently processed edge. It is set in subroutine LOCATE and used in programs COLECT, ITEST, and INTERS.

N2 is a pointer to the final vertex of the currently processed edge.

IEIPNT is a pointer to an edge which may have an intersection with edge IENPNT and is currently tested.

I1 is a pointer to the initial vertex of edge IEIPNT. It is set in subroutine COLECT and used in programs ITEST and INTERS.

I2 is a pointer to the final vertex of the edge IEIPNT.

X is the x coordinate in the picture plane of the intersection (if it exists) of edges IENPNT and IEIPNT. It is computed in function ITEST and used in programs COLECT and INTERS.

Y is the y coordinate of the intersection.

DASHED is a logical variable set to .TRUE. by PDRAW whenever the hidden edges are to be drawn as dashed lines.

SCALE sets the plot output for a maximum plotting area of 2* SCALE by 2* SCALE inches.

4.8 <u>Common Block DIMENS</u>

This common block contains the dimensions of the arrays used in the data structure.

MAXM* contains the dimension of arrays in block MODEL.

MAXF* contains the dimension of arrays in block FACE.

MAXE* contains the dimension of arrays in block EDGE.

MAXV* contains the dimension of arrays in block VERTEX.

MAXS* contains the dimension of arrays in block STACK.

5. DATA FORMAT

In this section the format of data files read by the program is described. A data file contains three types of cards - or card images.

delimiter cards	- delimit sections of a data file.
object cards	- contain object description.
vantage-point cards	- contain vantage point description.

5.1 The Structure of Data Files

Input data to PDRAW consists of a description of one or more scenes. Each scene contains one or more objects. The last object in a scene is followed by one or more vantage point specifications. The format of description of an object is given in 5.2. A scene, an object and a group of vantage points are delimited by special cards - delimiter cards - which contain a digit in column one as follows:

1 Scene header; indicates that a scene follows; columns 2-42 may contain the name of the scene or any comment which is printed.

2 Object header; indicates that a description of an object follows (in the format described in 5.2); columns 2-42 may contain the name of the object or any comment which is printed.

6 Vantage-point header; indicates that vantage point cards follow; this card is placed following the description of the last object in a scene.

9 End-of-scene card. This card follows the last vantage point card and indicates the end of the current scene.

It may be followed by an end-of-file mark or by the header of the next scene. The last scene in a file need not be followed by an end-of-scene card.

The size of the input data, i.e. the number of objects, faces, edges and vertices in a scene is limited by the size of the data structure. An error message is printed when the permitted size of the data structure is exceeded, indicating the part of the data structure that was exceeded. The program does not limit the number of scenes or vantage points in a data file.

5.2 Format of Object Specifications

In order to enter a planar-faced, solid object into the program, a description of the object must be created. Such a description consists of a list of the vertices, followed by a list of the faces of the object.

5.2.1 Vertices

Each vertex V_i is defined by three coordinates $V_i(x_i,y_i,z_i)$ given in the object coordinate system. The index 'i' may be arbitrarily assigned to a vertex. It is, however, assumed that $i = 1,2,3,....,M$ where M is the total number of vertices. The format of the list is:

M	I3	(total number of vertices
x_1 y_1 z_1	3F8.2	(first vertex
x_2 y_2 z_2	3F8.2	(second vertex
.. .. ..		
.. .. ..		
x_M y_M z_M	3F8.2	(last vertex

5.2.2 Faces

Each face of a planar-faced, solid object is a polygon. The program allows faces to have one or more holes in them, that is, to contain antifaces. Therefore, to describe a face, a list of vertices defining the outer edges as well as lists defining any possible holes or protrusions must be given. Each such list contains the indices 'i' of vertices as defined in 5.2.1. The direction in which vertices are listed for a face is such that the interior lies always to the right relative to an observer outside of the object. For an antiface, the direction is taken so that the interior lies to the left. Each list of vertices must be closed; i.e., the first vertex must appear also as the last one. Note that all the vertices and edges of a face (including those of all its antifaces, if any) must lie in a plane.

There are N lists of vertices describing faces. Each list has a flag 'iflag' as its first entry. A value of '1' indicates that the list describes the edges of a face, a value of '0' indicates that the list describes edges of an antiface. The next entry in the list is 'nv' which gives the number of vertices in the list. Following 'nv' there are the indices 'i' of the vertices. The first index is repeated as the last one. Note, that lists describing antifaces need not follow the list describing the corresponding face but may be placed anywhere in the list of faces. The format of this list is:

N	I3	(number of faces
$iflag_1$ nv_1 a b c d	20I3	(first face
$iflag_2$ nv_2 e b a	20I3	(second face
....		
....		
$iflag_N$ nv_N f g h	20I3	(last face

Note, that for a pair of indices of vertices 'i j', there must be one and only one pair 'j i'.

5.3 Format of Vantage Point Specifications

The type of drawing to be obtained from a vantage point and the coordinates of the vantage point given in the object coordinate system are described by ITYPE, XØ, YØ, ZØ on a single card image which is read with format (1X, I4, 5X, 3F10.3) where

ITYPE =	+1	orthographic projection; all lines drawn
	+2	orthographic projection; hidden lines dashed
	+3	orthographic projection; only visible lines drawn
	-1	perspective projection; all lines drawn
	-2	perspective projection; hidden lines dashed
	-3	perspective projection; only visible lines drawn

XØ, YØ, ZØ are the x, y and z coordinates of the vantage point given in the object coordinate system.

The projections and the coordinate systems used by the program are described in Appendix A.

5.4 An Example of a Data File

A sample data file of a scene is listed in Fig. 5.1. The scene consists of three objects: a cube, a pyramid and an octahedron. Following the last object, two vantage points are specified. The views obtained from these vantage points are drawn in Fig. 5.2-3.

```
1  A SCENE WITH THREE OBJECTS   (AN EXAMPLE)
2  A CUBE ************************************************************
 8 VERTICES
  10.00  -10.00  -10.00    VERTEX:  1    BOTTOM  FRONT  LEFT
  10.00   10.00  -10.00             2                   RIGHT
 -10.00   10.00  -10.00             3            BACK   RIGHT
 -10.00  -10.00  -10.00             4                   LEFT
  10.00  -10.00   10.00             5    TOP     FRONT  LEFT
  10.00   10.00   10.00             6                   RIGHT
 -10.00   10.00   10.00             7            BACK   RIGHT
 -10.00  -10.00   10.00             8                   LEFT
 6 FACES
 1  5  1  5  6  2  1      FRONT  FACE
 1  5  2  6  7  3  2      RIGHT  FACE
 1  5  5  8  7  6  5      TOP    FACE
 1  5  1  4  8  5  1      LEFT   FACE
 1  5  4  3  7  8  4      BACK   FACE
 1  5  1  2  3  4  1      BOTTOM FACE
2  A PYRAMID *********************************************************
 5 VERTICES
  30.00  -30.00   20.00    VERTEX:  1   TOP
  40.00  -40.00    0.00             2   BASE FRONT LEFT
  40.00  -20.00    0.00             3              RIGHT
  20.00  -20.00    0.00             4         BACK RIGHT
  20.00  -40.00    0.00             5              LEFT
 5 FACES
 1  4  1  2  5  1         LEFT  FACE
 1  4  1  3  2  1         FRONT FACE
 1  4  1  4  3  1         RIGHT FACE
 1  4  1  5  4  1         BACK  FACE
 1  5  2  3  4  5  2      BASE  FACE
2  AN OCTAHEDRON *****************************************************
 6 VERTICES
 -30.00   30.00   20.00    VERTEX:  1   TOP
 -30.00   30.00  -20.00             2   BOTTOM
 -20.00   20.00    0.00             3   BASE FRONT LEFT
 -20.00   40.00    0.00             4              RIGHT
 -40.00   40.00    0.00             5         BACK RIGHT
 -40.00   20.00    0.00             6              LEFT
 8 FACES
 1  4  1  3  6  1       TOP    LEFT  FACE
 1  4  1  4  3  1              FRONT FACE
 1  4  1  5  4  1              RIGHT FACE
 1  4  1  6  5  1              BACK  FACE
 1  4  2  6  3  2       BOTTOM LEFT  FACE
 1  4  2  3  4  2              FRONT FACE
 1  4  2  4  5  2              RIGHT FACE
 1  4  2  5  6  2              BASE  FACE
6 --------------------------------------------------------------------
  -1           4.50     94.75     22.55     VANTAGE POINT 1
  -3        -106.20     56.10     37.50     VANTAGE POINT 2
9
```

Figure 5.1 Listing of an example data file

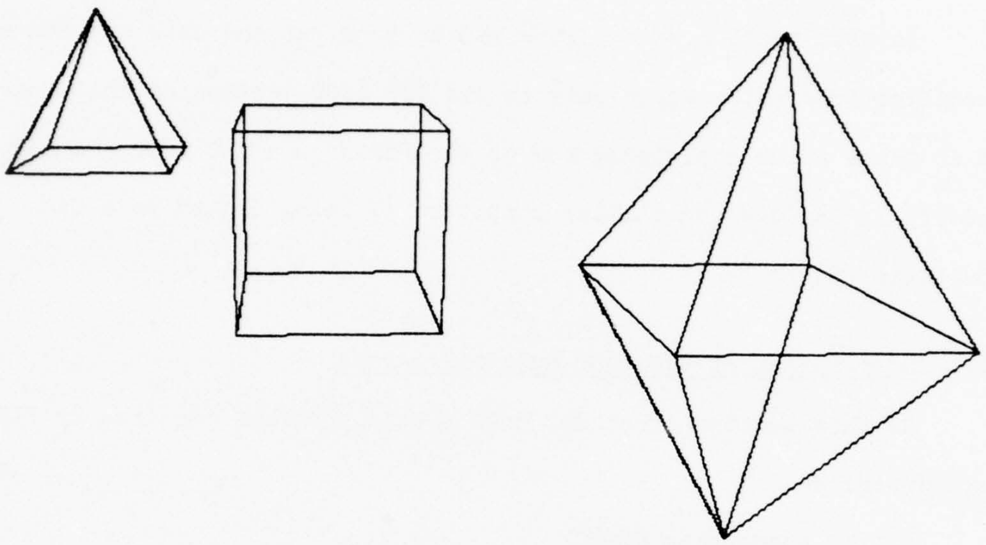


Figure 5.2 View of the scene from vantage point 1

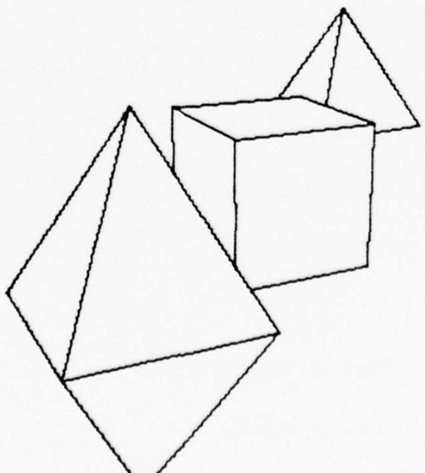


Figure 5.3 View of the scene from vantage point 2

6. THE CDC 6600 BATCH IMPLEMENTATION

In this section we describe system programs and data structure modifications which apply only to the CDC 6600 version of the program. A complete guide explaining how to run PDRAW on a CDC 6600 and how to modify it for use on similar computers is being issued as a User Bulletin [5].

6.1 Description of CDC 6600 System Programs

In this section three CDC 6600 system programs required by PDRAW are described.

6.1.1 Subroutine PLOTS

Subroutine PLOTS(LIMIT, TEXT) is used to initialize the plotting output. It is called before any other plotting subroutine. LIMIT is the total length of a plot in integer inches. TEXT is a Hollerith string of up to 40 characters identifying the user. PLOTS is a part of the CDC 6600 SCOOP plotting package.

6.1.2 Subroutine PLOT

Subroutine PLOT(X, Y, IPEN) is used to control all pen movements, create an origin and terminate the use of the plotting package. X and Y are coordinates (in inches) to which the pen is moved from its present position relative to the current origin. IPEN can have these values:

1	draw a dashed line
2	draw a solid line
3	move the pen in the up position

-1, -2, -3	the pen is moved as for 1, 2 and 3 respectively. In addition the final position of the pen (X, Y) becomes the new origin (0, 0)
999	terminates the plotting output.

PLOT is a part of the SCOOP plotting package.

6.1.3 Subroutine SECOND

Subroutine SECOND (TIME) returns the elapsed CPU time in seconds. SECOND is a part of the CDC FTN Fortran Library.

6.2 Notes to the CDC 6600 Data Structure

The following notes apply to the CDC 6600 implementation of the data structure:

1. The arrays in the data structure common blocks are dimensioned as follows:

MAXM.....	20	(the maximum number of objects)
MAXF.....	200	(the maximum number of faces)
MAXE.....	800	(the maximum number of edges *2)
MAXV.....	200	(the maximum number of vertices +1)
MAXS.....	40	(the maximum size of edge intersection stack)

This requires the following CM fields:

required to load	66400_8
required to run	54200_8

The program sizes are:

PDRAW programs	10000_8
Common Blocks	24500_8
system programs	20000_8

The dimensions of arrays in common blocks can be easily changed by using the text editor UPDATE and a common deck containing all common block statements.

2. The following numbers were found to be good computational tolerances EPS1, EPS2 and EPS3: 0.1, 0.01, 1.0E-10.

6.3 Examples

The program was tested with scenes of one, two and three aircraft [Fig. 6.1-2]. Each aircraft consists of about 340 vertices, 330 faces and 660 edges.

The following table gives memory requirements and execution times. The packed CDC version packs integer data into 60-bit memory words as described in [5].

Scene	Unpacked memory	Packed memory	Time [sec]
3 aircraft	210k	115k	11-25
2 aircraft	150k	74k	6-15
1 aircraft	110k	53k	3-6
Example [Fig. 5.2-3]	33k	33k	0.01-0.05

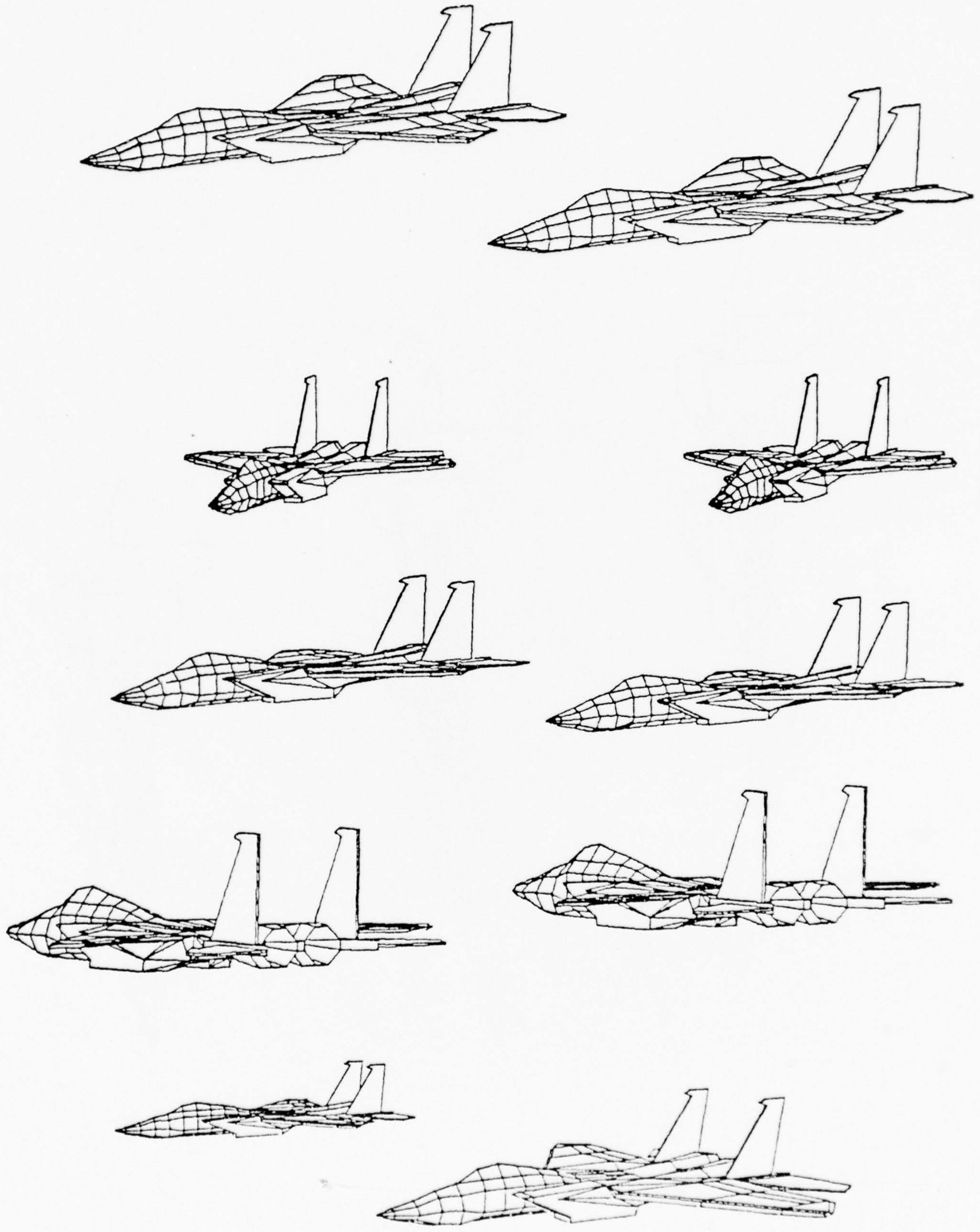


Fig. 6.1 Five views of two aircraft

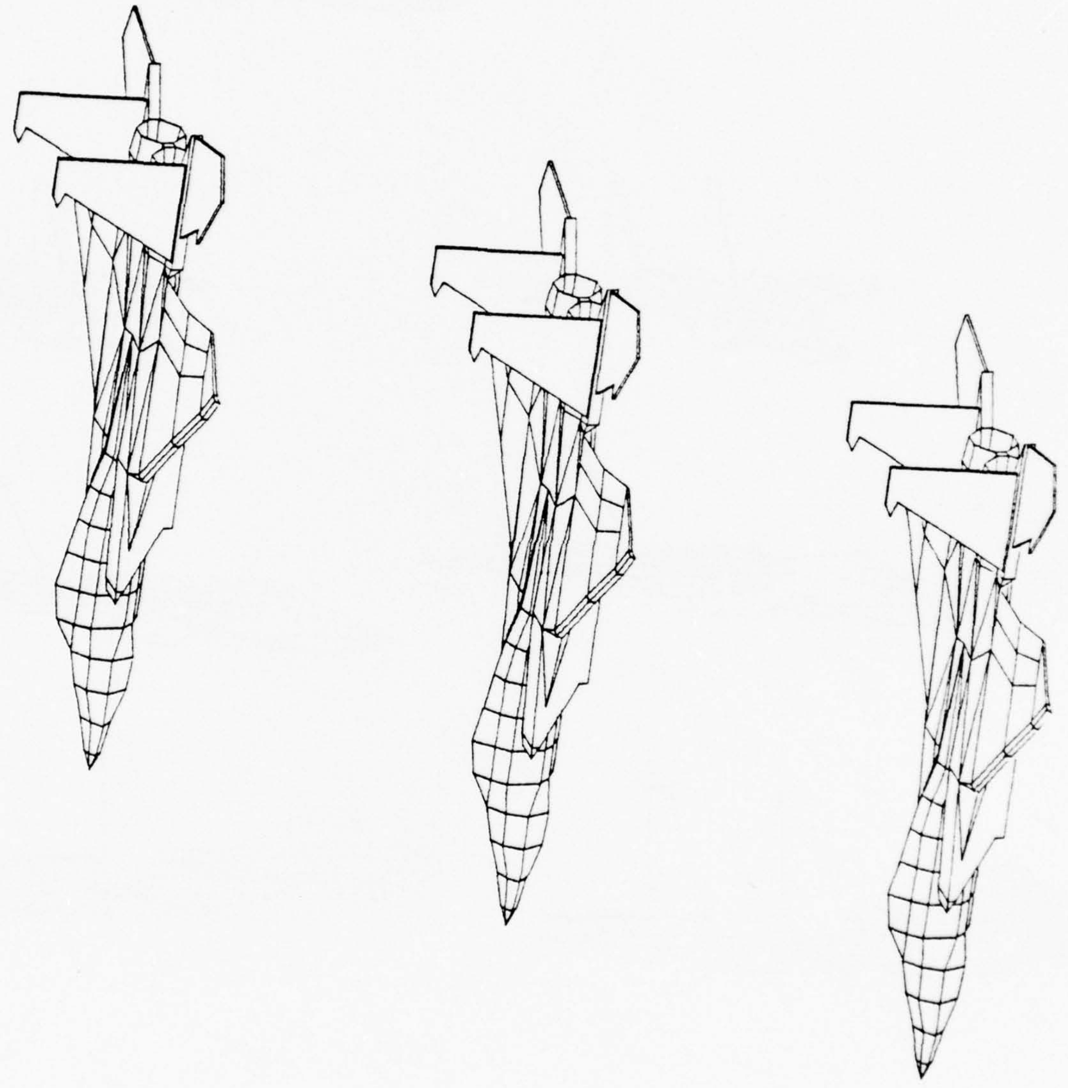


Fig. 6.2 A view of three aircraft

7. THE ADAGE AGT-30 INTERACTIVE IMPLEMENTATION

7.1 Review of the Implementation

It was decided to implement the algorithm on an Adage AGT-30 interactive graphics computer. The hidden-line scenes are displayed on a refresh CRT terminal. Hard copies of the drawing may be made on a dot matrix plotter or on an incremental plotter. The user can interact with the program through the use of interactive graphics I/O devices. The Adage computer also has a good Fortran IV compiler which produces efficient code, and an easy-to-use display text editor. A major problem encountered in writing the program was the limited amount of available main core memory. The Adage has 16k of 30-bit words of core memory, of which about 12k is available to the user. It was decided to make the program entirely core resident for minimum execution time. This is important to facilitate interactive graphic design.

The entire data structure of the program is stored in several common blocks and divided into two parts: vantage-point-independent data and vantage-point-dependent data. The vantage-point-dependent data is stored in three buffers of the Adage operating system AMOS/2.5. The three buffers are available to the user when the computer does not perform any disk, tape or printer/plotter I/O operations. This saved about 2k of user core memory. Another saving of about 1.2k of memory was made by simplifying the graphics processor GRAFX into a version which displays only two-dimensional vector lists and does not transform images except for scaling.

The vantage-point coordinates are computed from sampled values of either the joystick or three variable-control dials. An object des-. cription, called a model, is permanently stored in a disk file and entered into the program by a keyboard command. This allows the user to specify interactively a scene and a vantage point. A complete guide to running PDRAW on an Adage computer is available in the form of a User Bulletin [6].

7.2 Execution Time

The execution time of hidden-line removal increases with the complexity of the scene. For a simple scene, consisting of a single convex polyhedron, the execution time, including the time spent by refreshing the CRT, is about 50 ms. A model of a cube can "follow" the movements of the joystick in "real" time. For more complicated scenes the execution time increases to several seconds. In order to avoid a blank CRT screen, during hidden-line computations the output buffers for display lists were partitioned into two parts. While lists in one part are being displayed, the program computes new display lists into the other part of the display buffers. When the program finishes the removal calculations, the display processor is switched to display the new lists.

7.3 Description of Interactive PDRAW Programs

7.3.1 Program PDRAW

The main program in the interactive version is quite different from the main program in the batch version. This program samples the function switches and branches to various sections upon the depression of one of them. It also samples the variable control dials and the joystick and computes the coordinates of the vantage point from their values. PDRAW includes two image subprograms, PICTURE1 and PICTURE2, which alternate in displaying two sets of image lists. PDRAW starts the modified display processor XGRAFX by a DISPLAY statement and stops it by a call to NHALT which is an entry point into the display processor.

7.3.2 Function ALINE

ALINE is a real junction which returns as its value the y-axis intersection of an edge. It also returns the slope of the edge. ALINE is called twice by ITEST to help in determining the intersection of two edges. In the batch CDC 6600 version, this function is inserted in-line in function ITEST; in this version of the program it is made into an external function in order to save core memory.

7.3.3 Function CROSS

This is a real function which returns the magnitude of the cross-product of two three-dimensional vectors. It is called from 4 locations in function INTERS. The reasons for making CROSS into an external function are the same as those for function ALINE.

7.3.4 Subroutine VECTOR

This subroutine is called in the interactive version where calls to subroutine PLOT to draw a vector are made in the batch version. VECTOR converts the coordinates of an edge or a segment of an edge into a 2-D display format and appends them to a display list of either visible or invisible vectors.

7.4 Description of Interactive System Programs

In this section three interactive system programs required to run PDRAW on an ADAGE AGT-30 computer are described.

7.4.1 Program XGRAFX

This program is a simplified version of the graphical processor GRAFX. XGRAFX displays only lists of two-dimensional vectors and, except for scaling, does not perform any image transformations. These simplifications save about 1.2k of user main memory. The display processor has an entry point NHALT which stops the display of an image. It is called before a call to INPUTM to stop the display while INPUTM is reading from a disk.

7.4.2 Program XPUNCHX

This program punches the currently displayed image in a special format on paper tape via the fast paper tape punch. The paper tape then may be used to plot the image on an incremental digital plotter connected to a PDP-8 computer. The PDP-8 plotting program is called PLOT.

7.4.3 Function TIMER

This function measures elapsed execution time. There are two entry points in TIMER called TIMERON and TIMEROFF.

7.5 The Data Structure

7.5.1 Common Block OUTPUT

This common block, used only in the Adage interactive version, contains lists of all the visible and hidden vectors generated by the program. These lists are in a format suitable for direct display on the Adage CRT by the modified display processor XGRAFX.

IVIPNT is a pointer to the next available location in a list of visible vectors.

IHIPNT is a pointer to the next available location in a list of hidden vectors.

These two pointers are initialized by program PDRAW and incremented in subroutine VECTOR.

IVISIBLE (MAXVI) is an array containing two lists of visible vectors.

IHIDDEN(MAXHI) is an array containing two lists of hidden vectors.

Arrays IVISIBLE and IHIDDEN are each partitioned into two parts; while lists in one part of the arrays are being displayed, the program computes new lists into the other part of the arrays and then switches the display to these new lists. The visible vectors are displayed as solid lines, the hidden vectors can be displayed as dashed lines.

7.5.2 Notes to the Adaga Data Structure

The following notes apply to the Adage AGT-30 implementation:

1. The arrays described in this part which contain vantage-point dependent information are put into three common blocks MONBUFFER1, MONBUFFER2, and MONBUFFER3. These common blocks are stored in three overlay buffers of the Adage operating system. In this way they share space with I/O drivers and save about 2k of user core memory.

2. The arrays in the data structure common blocks are dimensioned as follows:

MAXM	10	(maximum number of objects)
MAXF	50	(maximum number of faces)
MAXE	200	(maximum number of edges * 2)
MAXV	50	(maximum number of vertices + 1)
MAXS	20	(maximum size of edge intersection stack)
MAXVI	300	(maximum size of visible vector display lists)
MAXHI	200	(maximum size of hidden vector display lists)

3. The subroutine TRANS scales down all vertices in the picture plane (array V2D) to (+1,-1) square in order to make them suitable for display on the Adage CRT. The point at "infinity" stored in V2D (MAXV,1) and V2D(MAXV,2) is set to (1.1, 1.1).
4. The computational tolerances in EPS1, EPS2 and EPS3 in common block PARAMS are set to 0.1, 0.01 and 0.000005, respectively.
5. SCALE in common block PARAMS is set to 1.0 but not used in this version.
6. DASHED in common block PARAMS is used only in the main program.

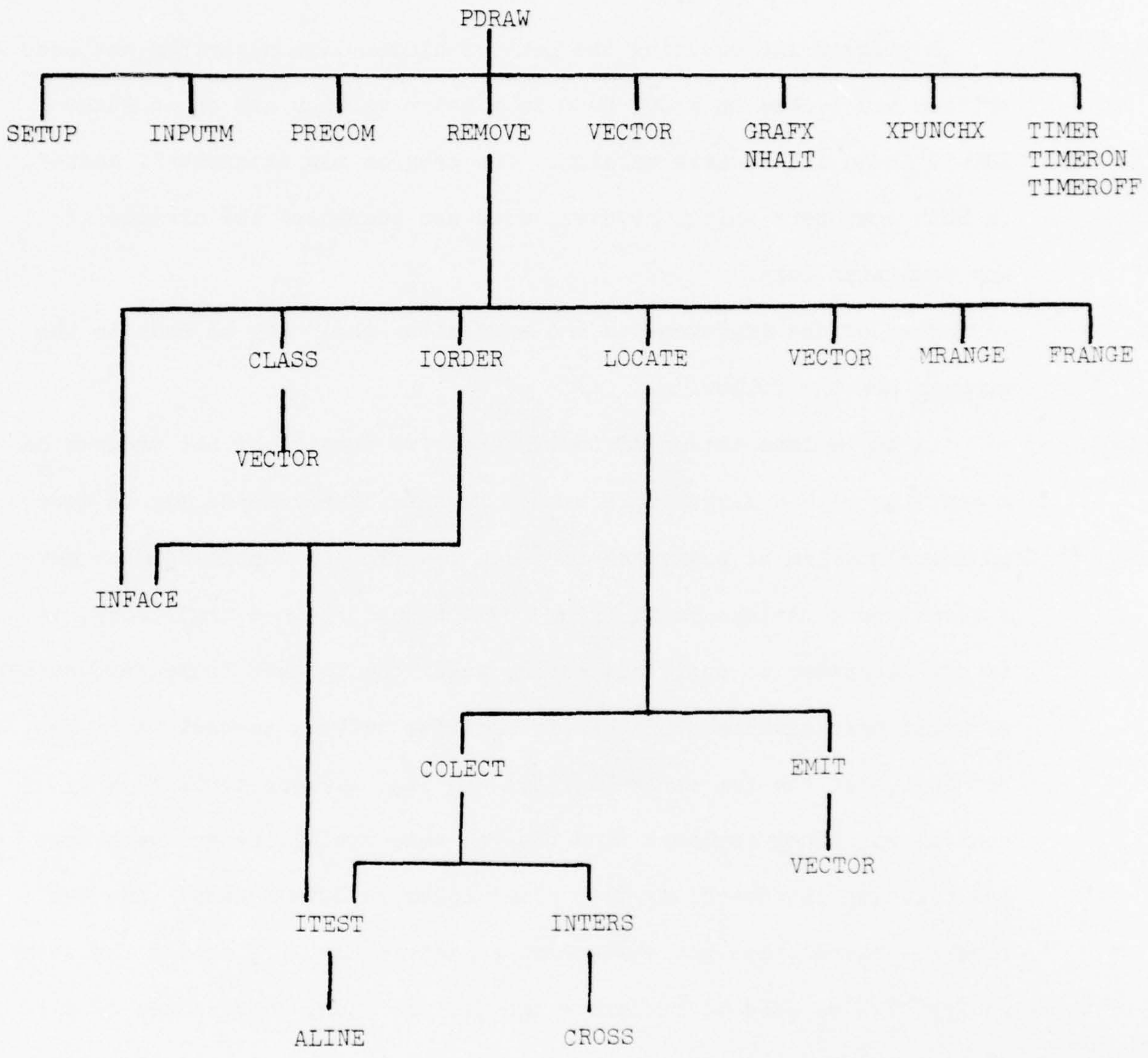


Figure 7.1 The connections of PDRAW routines in the ADAGE interactive version.

8. CONCLUSION

A program implementing the Loutrel hidden-line algorithm has been written and tested on a CDC 6600 in a batch version and on an Adage AGT-30 in an interactive version. The program was extensively tested on both computers which, however, does not guarantee the absence of any remaining bugs.

Some of the improvements and extensions which may be made to the program are the following.

In an implementation of the interactive version of the program on a computer with a larger main memory further improvements may be made to the algorithm as suggested in [4]. The problem considered is: given a scene and a vantage point which moves around in on a trajectory, it is not necessary to apply the entire algorithm to each frame, but rather we would take advantage of the similarities between successive frames. We note, that the scenes in Fig. 8.1 and Fig. 8.2 are topologically equivalent. That is, each face has the same visibility and each edge has the same intersections with other edges in both scenes. The two scenes, however, are not equivalent geometrically. To obtain the scene in Fig. 8.2 we need to recompute the picture plane coordinates of all vertices and edge intersections from Fig. 8.1. The view of the scene in Fig. 8.3 is not topologically equivalent to the views of the scene in Fig. 8.1 and Fig. 8.2 and we need to re-apply the algorithm. This method would significantly reduce the execution time for most frames and should be used in an interactive version where the execution time is considered to be important.

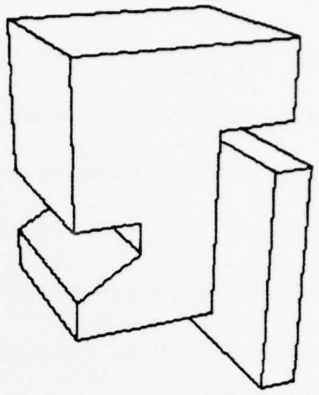


Figure 8.1 First view of a scene of two objects

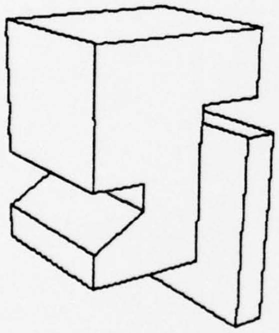


Figure 8.2 Second view of a scene of two objects

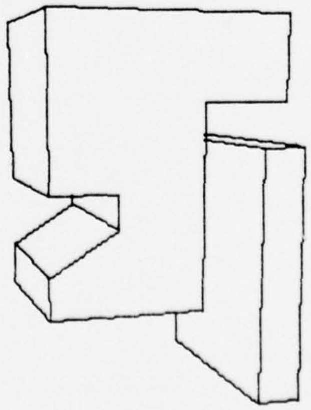


Figure 8.3 Third view of a scene of two objects

Other improvements may include the shading of surfaces and drawing the scenes on gray-level display devices. The program could be extended to allow a vantage point to be "inside" a scene, that is inside an object or between objects.

In other implementations of this program, subroutines SETUP, INPUTM and PRECOM may be made into overlays, which would save space for the data structure.

REFERENCES

1. Loutrel, P. P., "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," IEEEE Transactions on Computers, Vol. C-19, No. 3, March 1970, pp. 205-213.

2. Loutrel, P. P., "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," doctoral dissertation, Department of Electrical Engineering, New York University, September 1967. (N68 13 830)*

3. Sutherland, I. E., Sproull, R. F., and Schumacker, R. A., "A Characterization of Ten Hidden-Surface Algorithms," Computing Surveys, Vol. 6, No. 1, March 1974, pp. 1-55.

4. Lavin, M. A., "An Application of Line-Labeling and Other Scene-Analysis Techniques to the Problem of Hidden-Line Removal," Working Paper, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, March 1974.

5. Potmesil, M., "A Guide to PDRAW - A CDC 6600 Implementation of the Loutrel Hidden-Line Algorithm", User Bulletin U-032, Computer Research Laboratory, Electrical and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, N. Y. 12181, September 20, 1976.

* Available under this number from NTIS, U. S. Dept. of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

6. Potmesil, M., "A Guide to PDRAW - An Adage AGT-30 Implementation of the Loutrel Hidden-Line Algorithm", User Bulletin U-033, Computer Research Laboratory, Electrical and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, N. Y. 12181, September 20, 1976.

Appendix A

THREE-DIMENSIONAL TRANSFORMATION AND PROJECTIONS

We describe here the transformation from a three-dimensional coordinate system to a two-dimensional picture plane coordinate system as well as the two projections used by the program.

We define the following rectangular coordinate systems [Fig. A.1]:

1. The object coordinate system (x,y,z) in which the vertices of all objects as well as all vantage points are defined. The vertex coordinates are entered in the program from input data files. The coordinates of a vantage point specified by $Q(x_o,y_o,z_o)$ are computed from sampled values of either the joystick or variable-control dials. The object coordinate system is a right-handed system.

2. The vantage-point coordinate system (x',y',z') is a left-handed system with origin at the vantage point. The z' axis goes from $Q(x_o,y_o,z_o)$ through the origin of the (x,y,z) system, and the x' axis is parallel to the x-y plane of the (x,y,z) object coordinate system.

3. The picture plane coordinate system (x",y") is a two-dimensional coordinate system whose axes lie in a plane that is perpendicular to the z' axis and goes through the origin of the (x,y,z) system. The x" axis lies in the x-y plane of the (x,y,z) system.

In the program, the coordinates of the object system (x,y,z) are stored in array VERTEX. The coordinates of the vantage-point system (x',y',z') are stored in array VTRANS. They are saved only in order

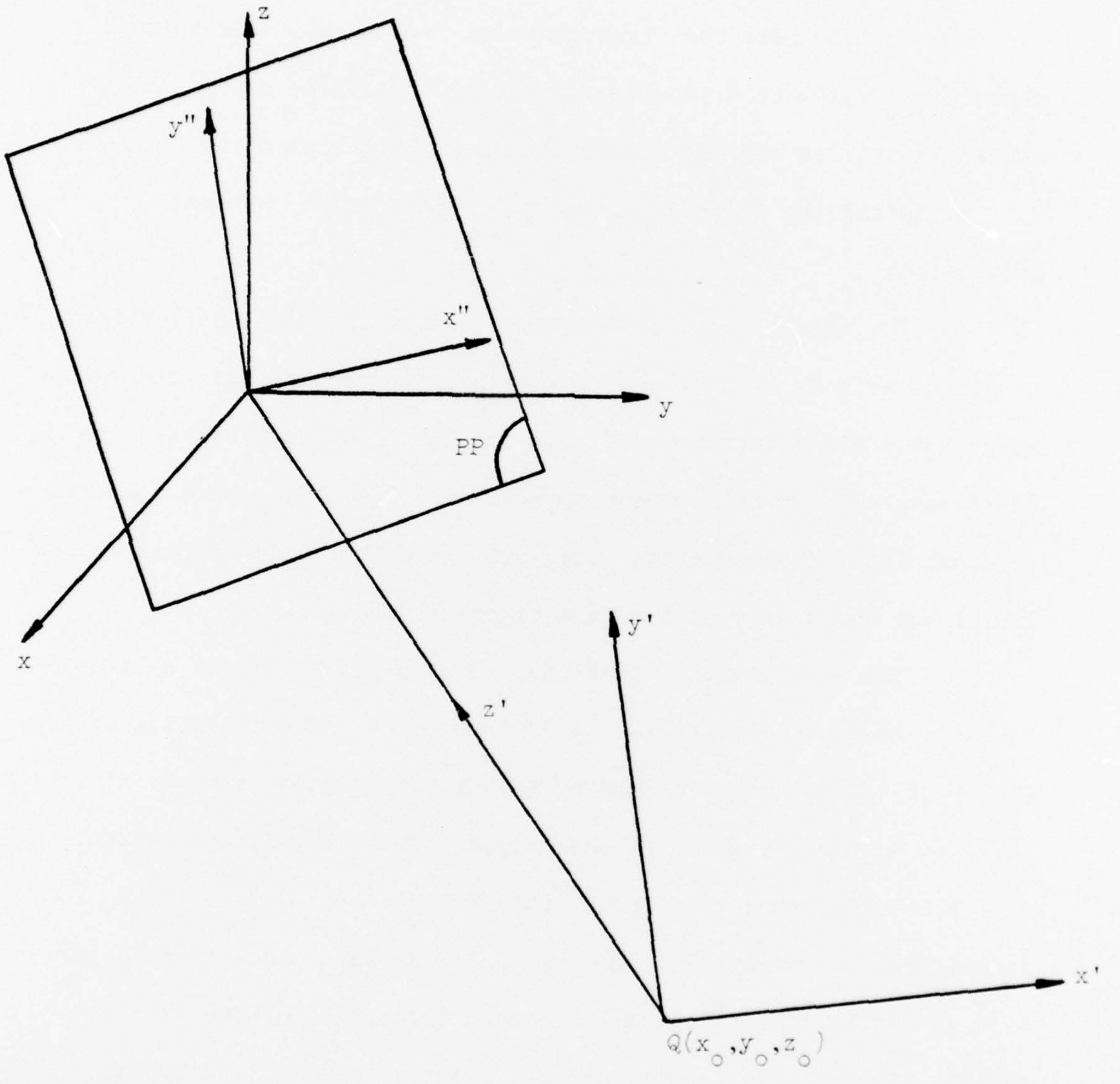


Figure A.1 Three coordinate systems

to speed up computations during hidden-line removal in function INTERS. Coordinates of the picture-plane coordinate system (x",y") are stored in array V2D. They are used for display as well as in computing edge intersections.

The transformation from the object system (x,y,z) to the vantage point system (x',y',z') consists of the following four steps:

1. Map $Q(x_o,y_o,z_o)$ into the origin of the (x',y',z') system:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -x_o \\ 0 & 1 & 0 & -y_o \\ 0 & 0 & 1 & -z_o \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Rearrange the axes in to a left-handed coordinate system [Fig. 8.2]:

$$T_2 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Rotate about the y' axis by angle α [Fig. A.3]:

$$d = x_o^2 + y_o^2 \qquad \cos\alpha = y_o/d \qquad \sin\alpha = x_o/d$$

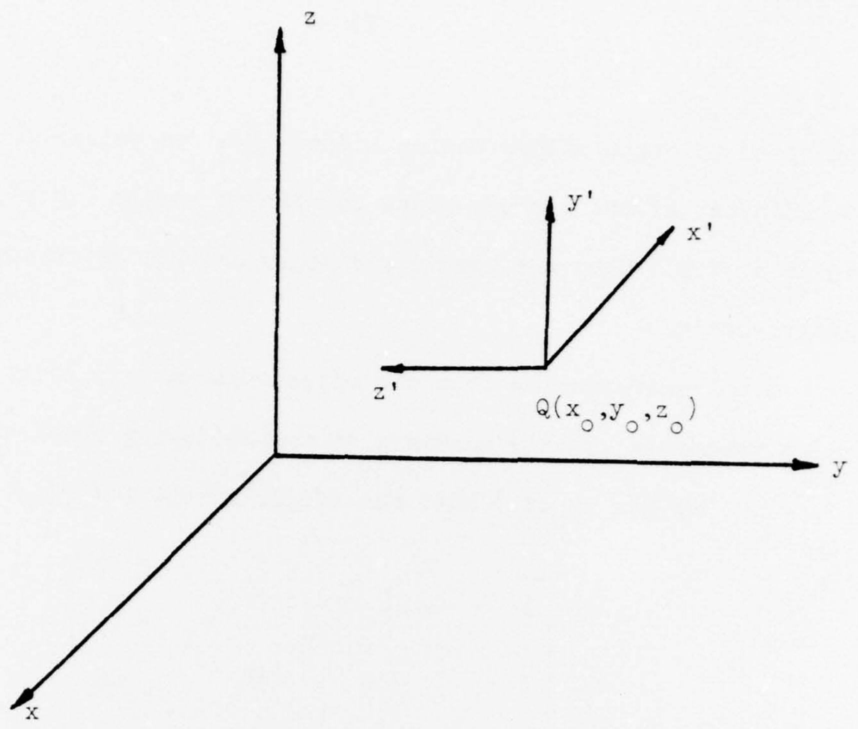


Figure A.2 Initial position of (x,y,z) and (x',y',z') coordinate systems.

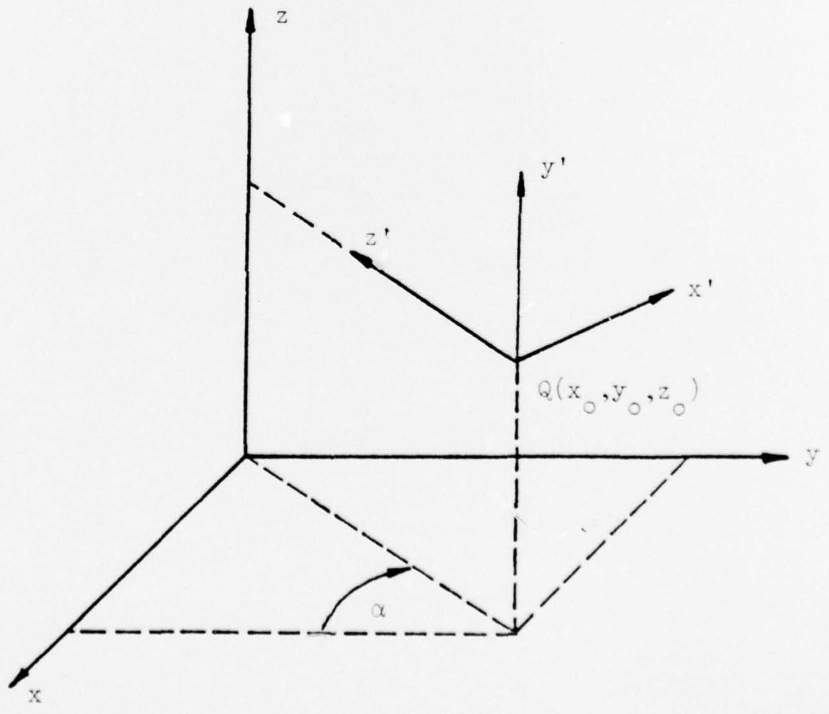


Figure A.3 Rotation about the y' axis by angle α.

$$T_3 = \begin{bmatrix} y_o/d & 0 & -x_o/d & 0 \\ 0 & 1 & 0 & 0 \\ x_o/d & 0 & y_o/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Rotate about the x' axis by angle β [Fig. A.4]:

$D = x_o^2 + y_o^2 + z_o^2$ $\cos \beta = d/D$ $\sin \beta = z_o/D$

$$T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & d/D & z_o/D & 0 \\ 0 & -z_o/D & d/D & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The four matrices can be multiplied into a single transformation matrix: $T = T_1 \, T_2 \, T_3 \, T_4$

We obtain:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -y_o/d & x_o/d & 0 & 0 \\ -x_o z_o/dD & -y_o z_o/dD & d/D & 0 \\ -x_o/D & -y_o/D & -z_o/D & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

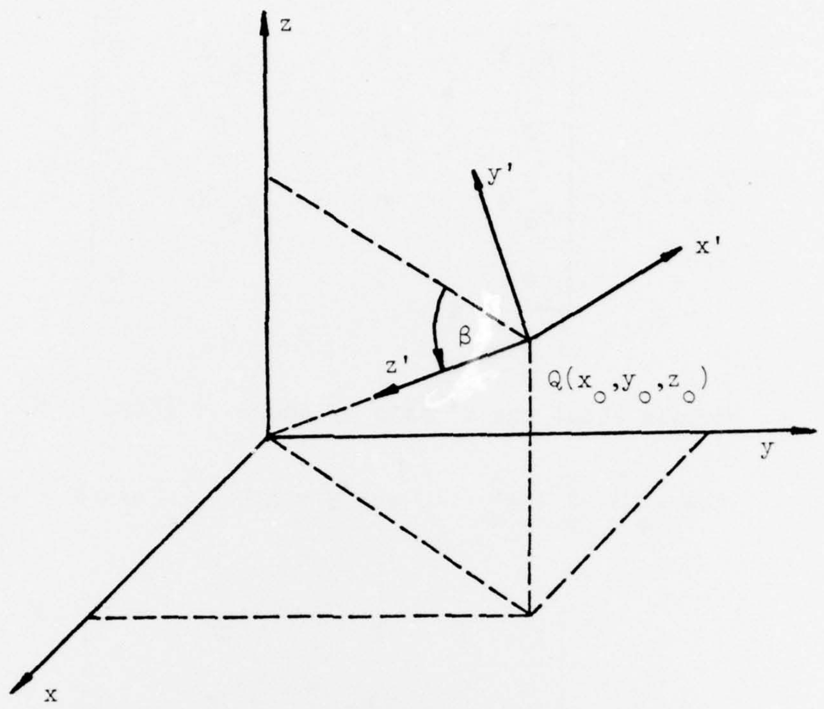


Figure A.4 Rotation about the x' axis by angle β.

The transformation from the vantage-point coordinate system (x',y',z') to the picture-plane coordinate system (x'',y'') depends on the projection system that is selected:

1. Perspective projection:

In this projection we have to transform using these two equations [Fig. A.5]:

$$x'' = Dx'/z'$$

$$y'' = Dy'/z'$$

2. Orthographic projection:

In this projection we do not divide by the depth of each point. The equations are [Fig. A.6]:

$$x'' = x'$$

$$y'' = y'$$

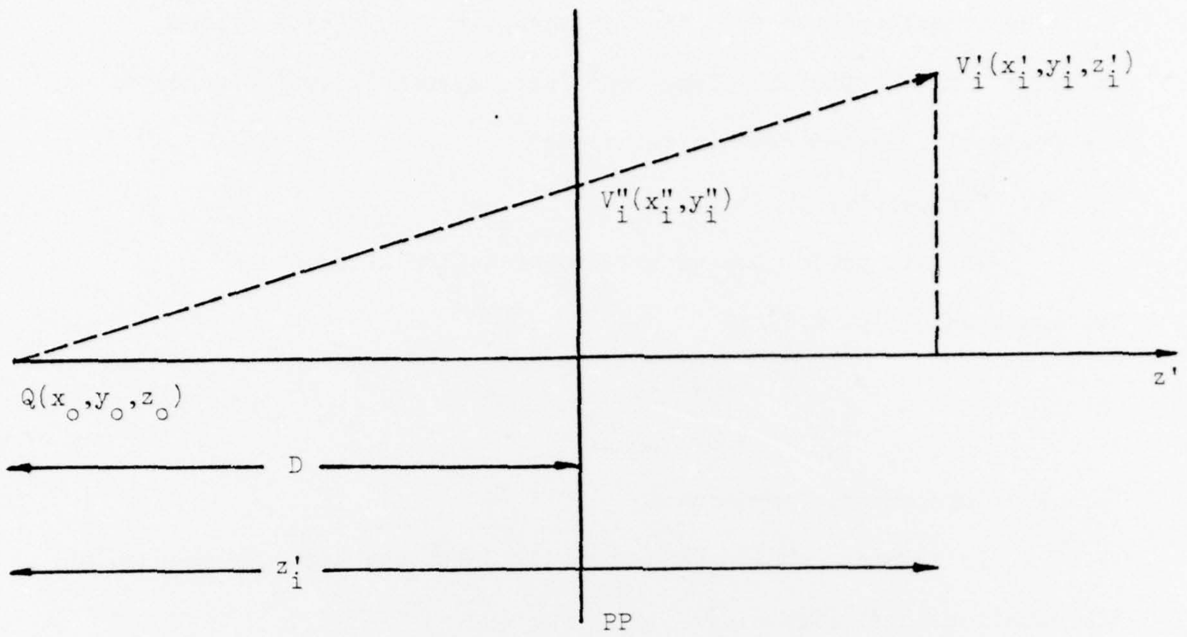


Figure A.5 Perspective projection

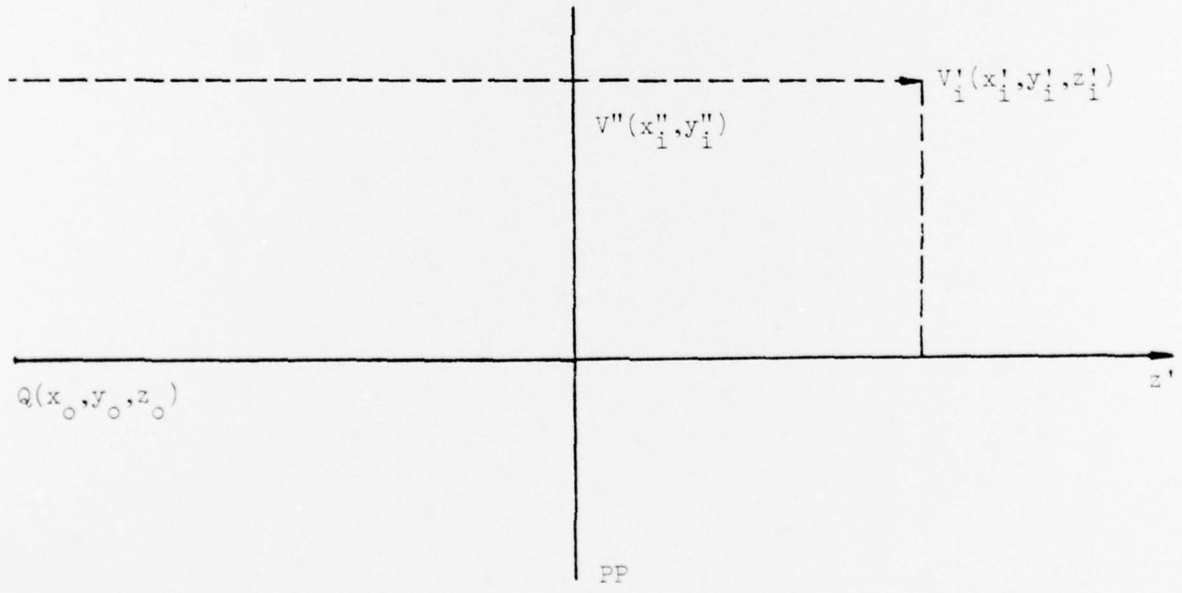


Figure A.6 Orthographic projection

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR - TR - 77 - 0104	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER (9)
4. ~~TITLE (and Subtitle)~~ (6) AN IMPLEMENTATION OF THE LOUTREL HIDDEN-LINE ALGORITHM		5. TYPE OF REPORT & PERIOD COVERED Technical ~~Interim~~ rept.,
		6. PERFORMING ~~ORG. REPORT~~ NUMBER
7. ~~AUTHOR(s)~~ (10) Michael/Potmesil		8. CONTRACT OR GRANT NUMBER(s) AFOSR 76-2937
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electrical and Systems Engineering Department Rensselaer Polytechnic Institute Troy, New York 12181		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A3 (12) 63p.
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB DC 20332		12. ~~REPORT DATE~~ (11) Sep 1976
		13. NUMBER OF PAGES 62
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (14) CRL-49		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

(15) AF-AFOSR-2937-76

17. DISTRIBUTION STATEMENT (~~of the abstract~~ entered in Block 20, if different from Report)

(16) 23Ø4 (17) A3 (18) AFOSR

18. SUPPLEMEN~~TARY NOTES~~

(19) TR-77-Ø1Ø4

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Computer Graphics
Hidden-Line elimination
Computer-aided design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes an implementation of a variation of the hidden-line algorithm developed by P. Loutrel. The new algorithm produces hidden-line drawings of scenes composed of opaque polyhedra as well as of some more general types of flat-faced objects not covered by the original Loutrel algorithm. The program is written in Fortran IV and has been implemented in a batch version on a CDC 6600 computer and in an interactive version on an ADAGE AGT-30 interactive graphics computer.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

409 952

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)