A PROGRAMMING SYSTEM
FOR PRODUCING
COMPUTER GENERATED MOTION PICTURES
ON THE
GRAPHIC INTERPRETATION FACILITY
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>I. AN INTRODUCTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 A Description of the Hardware</td>
<td>I-1</td>
</tr>
<tr>
<td>1.2 The Basic Features of the Programming System</td>
<td>I-3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>II. A DETAILED DESCRIPTION OF THE PROGRAMMING SYSTEM</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 The Form of a Procedure</td>
<td>II-1</td>
</tr>
<tr>
<td>2.1.1 The %OPTIONS= Statement (Preliminary Description)</td>
<td>II-1</td>
</tr>
<tr>
<td>2.1.2 The %GEOMETRY= Statement</td>
<td>II-2</td>
</tr>
<tr>
<td>2.2 Data Entry Macros</td>
<td>II-2</td>
</tr>
<tr>
<td>2.2.1 The DSPINIT Macro</td>
<td>II-3</td>
</tr>
<tr>
<td>2.2.2 The DSPTEXT Macro</td>
<td>II-3</td>
</tr>
<tr>
<td>2.2.3 The DSPKBDRD Macro</td>
<td>II-3</td>
</tr>
<tr>
<td>2.2.4 The DSPLPEN Macro</td>
<td>II-4</td>
</tr>
<tr>
<td>2.3 Geometric Definition Macros</td>
<td>II-4</td>
</tr>
<tr>
<td>2.3.1 The POINT Macro</td>
<td>II-4</td>
</tr>
<tr>
<td>2.3.2 The LINE Macro</td>
<td>II-5</td>
</tr>
<tr>
<td>2.3.3 The SURFACE Macro</td>
<td>II-5</td>
</tr>
<tr>
<td>2.3.4 The OBJECT Macro</td>
<td>II-6</td>
</tr>
<tr>
<td>2.3.5 The VIEW Macro</td>
<td>II-6</td>
</tr>
<tr>
<td>2.3.6 The GPRINT Macro</td>
<td>II-7</td>
</tr>
<tr>
<td>2.4 Picture Generation Macros</td>
<td>II-7</td>
</tr>
<tr>
<td>2.4.1 The LEADER Macro</td>
<td>II-7</td>
</tr>
<tr>
<td>2.4.2 The PICTURE Macro</td>
<td>II-7</td>
</tr>
<tr>
<td>2.4.3 The REPEAT Macro</td>
<td>II-8</td>
</tr>
<tr>
<td>2.5 Optimizing the Program</td>
<td>II-9</td>
</tr>
<tr>
<td>2.5.1 Details of the %OPTIONS= Statement</td>
<td>II-9</td>
</tr>
<tr>
<td>2.5.2 Overlay Structures</td>
<td>II-10</td>
</tr>
<tr>
<td>2.6 Future System Development</td>
<td>II-11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>III. RUNNING THE PROGRAM</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV. A PROGRAMMING EXAMPLE: THE STELLATED DODECAHEDRON</td>
<td>IV-1</td>
</tr>
</tbody>
</table>
Table of Contents (continued)

APPENDIX A: The Geometric Data Structure ............... A-1
APPENDIX B: The Projection Matrix ..................... B-1
APPENDIX C: Some Details on the Removal of Hidden Lines .... C-1
APPENDIX D: A Pre-Processed Program ................... D-1
CHAPTER I

AN INTRODUCTION

In many of the physical sciences, and in Physics in particular, it is a
common occurrence to have either a mathematical theory of a phenomenon or a
large mass of data which measures a phenomenon. A mathematical theory pro-
vides one with a microscopic, quantitative view of the phenomenon. A macro-
scopic, qualitative view is often difficult to obtain from the theory. Any
understanding of a large mass of data is always difficult to acquire.

This report describes a programming system which can be used to gain
understanding of these types of problems. The programming system makes it
quite easy for a user to write programs which generate animated displays on
the Graphic Interpretation Facility at SLAC. The animation sequence may be
viewed on a CRT screen or it may be recorded on 16 millimeter motion picture
film for later projection.

1.1 A Description of Hardware

The main component of the Graphic Interpretation Facility is an IDIOM
Graphic Console. The IDIOM consists of a Varian 620/I computer with 8192
words of 16 bits each, a teletype, and a display console made by Information
Displays Incorporated. The display console includes a CRT with a usable area
of 13 inches by 13 inches, a light pen, and a set of 32 function keys. When
the IDIOM is operating, the 620/I memory will contain a program for the 620/I
to execute, and a program (display file) for the CRT to execute. The instructions
(orders) in the display file may include orders to display characters, points,
or straight line segments, perform unconditional or subroutine jumps, or in-
terrupt the 620/I. The 620/I instruction set includes, in addition to the usual
set for a modern small computer, instructions to start and stop the CRT in its
execution of the display file and instructions to read and reset registers
associated with the display operation.

The IDIOM and the SYSTEM/360 are connected through an IBM 2701 Parallel
Data Adapter Unit. Information may be transmitted either way through this link.
In addition, the 620/I has the ability to send an attention interrupt to the
SYSTEM/360. The SYSTEM/360 is not able to interrupt the IDIOM however.

The CRT screen on the IDIOM display console has an imaginary grid of 1024
by 1024 raster units imposed on it. A point may be plotted at any addressable
location and a line may be drawn between any two addressable locations. The lower left corner of the display has coordinates (0,0) and the upper right has coordinates (1023, 1023). Characters may be plotted in four sizes, small, medium, large, and very large. These characters may be plotted either horizontally, or vertically (rotated 90 degrees counter-clockwise). The coordinates given for plotting a character specify the lower left corner of the character. The following table gives more information on the plotting of characters. In this table, the between-character dimensions are set by the programmer and the given values represent suggestions.

<table>
<thead>
<tr>
<th>CHARACTERISTIC</th>
<th>SMALL</th>
<th>MEDIUM</th>
<th>LARGE</th>
<th>V. LARGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characters per line</td>
<td>128</td>
<td>73</td>
<td>48</td>
<td>36</td>
</tr>
<tr>
<td>Lines per display</td>
<td>85</td>
<td>48</td>
<td>34</td>
<td>24</td>
</tr>
<tr>
<td>Character spacing (raster units)</td>
<td>8</td>
<td>14</td>
<td>21</td>
<td>28</td>
</tr>
<tr>
<td>Line spacing (raster units)</td>
<td>12</td>
<td>21</td>
<td>30</td>
<td>42</td>
</tr>
</tbody>
</table>

Information may be displayed in any of four intensity levels. Lines may have any of four line structures (solid, dashed, dot-dashed, or dots). Displayed information may be in a blinking mode or a steady mode. In addition, the light pen can be made sensitive or insensitive to displayed information.

The Graphic Interpretation Facility has two other devices, a 16 millimeter movie camera and a 3-D viewer, which may be used with the IDIOM. The SLAC movie camera consists of a 16 millimeter, model 16M ARRI FLEX motion picture camera with an animation motor. An interface between the movie camera and the 620/I was designed and built at SLAC. Through this interface, the 620/I can control the animation motor and sense when the shutter is open.

The movie camera may be operated in any of three movie-making modes. These three modes are:

1. Frame Synchronized Mode: In this mode the movie camera runs at its normal speed of 24 frames per second and the 620/I continually senses the shutter open signal from the camera. When the shutter initially opens, the 620/I will start the display. After going through the display file once, the display is turned off until the next time the shutter opens. This mode may, therefore, be used to take flicker-free movies of any existing interactive display program.
2. Single Pulse Animation Mode: In this mode the 620/I sends signals to the camera telling it when to open its shutter. After the shutter has opened, the display will be started. The length of time that the shutter remains open is controlled by a switch on the animation motor. This switch may be set to give exposure times of 1/8, 1/4, and 1/2 seconds.

3. Double Pulse Animation Mode: This mode is a generalization of the previous mode. It covers the case where a single frame may consist of very long display files. Thus, a single frame of movie film may contain a more complex picture than can be viewed in the normal interactive mode.

The SLAC 3-D viewer is a device which contains a motor driven rotating disk. The disk contains clear areas and opaque areas. When the user looks through this device, the left and right eyes will alternately be blocked by the opaque areas on the disk. An interface has been built at SLAC which allows the 620/I to sense when the left or right eye has a clear view. By flashing different images on the screen when each eye is open, it is possible to present the user with a stereoscopic picture.

1.2 The Basic Features of the Programming System

This project is based on previous work which is described in the publication, CGTM No. 80 "THE SLAC SCOPE PACKAGE FOR THE IDIOM". That report describes a set of PL/I procedures which may be used to write interactive programs for the IDIOM. Among the many things that those procedures can do is to synchronize the display with the movie camera and the 3-D viewer.

The work being described in this report consists of an effort to use the macro facility of PL/I to greatly simplify the writing of programs which produce animation on the IDIOM. This programming system is basically designed to produce standard and 3-D movies; however, any animated sequence may be viewed on the CRT, so it is also useful for on-line animation. 3-D movies are generated by putting left and right eye images on alternate frames and then merging the film through an optical printer to obtain film which may be projected for an audience wearing polaroid viewing glasses.

The programming system provides a programmer with a simple means of defining a geometric figure in space. The geometric figure may be anything that can be constructed from lines and points. For added realism, surfaces may be
defined and the hidden lines removed from the figure.

To specify the animation sequence, a conceptual camera is positioned in space, pointing at the geometric figure and a picture taken. Between successive pictures in an animation sequence, the camera position may be changed, the geometric figure changed, or both may be changed.

Once a program has been written and checked out, it is ready to be run in any of the four basic modes, 2-D interactive, 3-D interactive, 2-D movie making, or 3-D movie making, that are available under this programming system.

For the remainder of this report, familiarity with PL/1 is assumed.
CHAPTER II

A DETAILED DESCRIPTION OF THE PROGRAMMING SYSTEM

The user of this programming system prepares standard PL/1 procedures with certain required conventions. These conventions are described in Section 2.1. In addition to all of the usual PL/1 statements, the user has available certain PL/1 macros to create interactive displays (Section 2.2), define geometric figures (Section 2.3), and generate animation sequences (Section 2.4). In Section 2.5, some techniques to optimize a program are discussed, and Section 2.6 suggests some extensions to this programming system.

2.1 The Form of a Procedure

There are a few conventions that a procedure must follow if it is a main program, or if it uses any of the macro statements in Sections 2.2, 2.3, or 2.4. Three PL/1 pre-processor statements must be included in such a procedure at very specific locations. These pre-processor statements are:

1. %INCLUDE HEAD1; This statement must be placed immediately after the PROCEDURE statement.
2. %INCLUDE HEAD2; This statement must be placed after the HEAD1 statement and before the first executable statement.
3. %INCLUDE TAIL; This statement must be placed immediately before the final END statement. In addition, the final END statement should not have a statement label, and should serve to end only the procedure block.

The correct way for the main program to terminate is to fall through and execute the %INCLUDE TAIL; statement. Sub-procedures may use the RETURN statement but main programs should not. Two other statements, the %OPTIONS= and %GEOMETRY= statements, must be placed between the HEAD1 and HEAD2 statements.

2.1.1 The %OPTIONS= Statement (Preliminary Description)

The identifier OPTIONS is a compile-time character string variable. The string that OPTIONS may contain consists of a number of fields separated by ampersands. The discussion of most of these fields will be deferred until Section 2.5.1. The only field which need be mentioned now is the keyword MAIN which must be used to indentify a main program. Thus,

- II-1 -
valid %OPTIONS= statements are:

   %OPTIONS= 'MAIN';
   %OPTIONS= ' ';

2.1.2 The %GEOMETRY= Statement

The identifier GEOMETRY= is also a compile-time character string variable. The string that GEOMETRY contains will define all of the geometric variables which the programmer will use. The %GEOMETRY= statement should be identical in all procedures which constitute the final program.

The keywords which may appear in the GEOMETRY string are: POINTS, LINES, SURFACES, OBJECTS, and VIEWS. The geometric variables may be simple variables or arrays. The keyword groups are separated by ampersands and variables within a keyword group are separated by commas.

Some examples of valid %GEOMETRY= statements are:

   %GEOMETRY= 'POINTS= Pt(3) & LINES=LN(2)';
   %GEOMETRY= 'POINTS=P,q(2,3),R(7) & LINES=LN(17) &
               SURFACES=S(12), T & OBJECTS=X & VIEWS=Y';

OBJECT variables are collections of points, lines, and surfaces while VIEW variables specify camera position.

The user of this programming system may treat these geometric variables as additional data types that have been added to PL/1. If a geometric variable is passed, as an argument, to a procedure, then the receiving procedure should declare the parameter as a POINTER. After preprocessing by the PL/1 compiler, all geometric variables are, in fact, converted to POINTER variables.

2.2 Data Entry Macros

This section describes some macros which may be used to enter data interactively into a program. By using these macros, the programmer can give the console operator the ability to enter parameters through the keyboard or select alternatives with the light pen.

Four macros are involved here. DSPINIT is an initialization macro; DSPTEXT is used to put explanatory text on the screen; DSPKBRD is used to put messages on the screen which may be modified by the console operator using the keyboard, and DSPLPEN is used to put messages on the screen which may be selected with the light pen. The logical order that these macros must be used is:
(1) initialization with DSPINIT, (2) any number of uses of DSPTEXT, and (3) either DSPKBRD or DSPLPM1 (but not both).

2.2.1 The DSPINIT Macro

This macro clears the screen and initializes the system so that it is ready to accept an interactive display.

The macro is invoked by the statement:

DSPINIT;

This macro has no arguments.

2.2.2 The DSPTEXT Macro

Each use of this macro will add up to five lines of text to an interactive display.

The macro is invoked by the statement:

DSPTEXT(X,Y,OPTIONS, TEXT1,...,TEXT5);

The values of X and Y give the screen coordinates, in terms of the 1024 by 1024 raster unit grid, of the first character in TEXT1. The OPTIONS argument may consist of a number of fields separated by ampersands.

DPARM= ... any of (STDY, WINK), (DIMM, MEIM, BRIT, VBRT) ...
CPARM= ... any of (SMAL, MEIM, LARG, VLRG) ...
VSPAC= ... vertical spacing between lines ...

The default values of OPTIONS are DPARM=STDY BRIT & CPARM=MEIM & VSPAC= -25. Some examples of DSPTEXT statements are:

DSPTEXT (200,1000,,'TITLE FOR DISPLAY');
DSPTEXT (X1, Y1, CPARM=LARG & VSPAC= -30, LINE1, LINE2, LINE3);

2.2.3 The DSPKBRD Macro

This macro will add up to five keyboard input buffers to an interactive display. The initial contents of the buffers will appear on the screen and the program will wait for operator action. The console operator may then change the contents of any of the input buffers by typing on the keyboard. When all of the data is correctly entered, the console operator should press the ESCAPE key. At this time, the contents of the buffers are read from the screen into the program.

The macro is invoked by the statement:

DSPKBRD(X,Y,OPTIONS,BUFF1, ..., BUFF5);
All of the arguments are the same as those in the DSPTEXT macro except that the default values of OPTIONS are: DPARM=STDYBRTT & CPARM=MEIM & VSPAC=-35. An example of a use of the DSPKBRD macro is:

DECLARE C10 CHARACTER (10)
INITIAL (' 75.00');
DSPKBRD (200, 500,, C10);

2.2.4 The DSPLPEN Macro

This macro will add up to ten light buttons to an interactive display. The program then waits for the console operator to use the light pen to select one of the messages. The index of the selected message is made available to the program.

The macro is invoked by the statement:

DSPLPEN(X,Y,OPTIONS,INDEX,BUTN1,...,BUTN10);

The variable INDEX is set to one if BUTN1 was selected with the light pen, to two if BUTN2 was selected, etc. The other arguments are the same as in the DSPKBRD macro. An example of a use of the DSPKBRD macro is:

DSPLPEN(600, 900,, I, 'BUTTON1', 'BUTTON2', 'BUTTON3');

2.3 Geometric Definition Macros

This section describes the macros which may be used to define three dimensional geometric figures. The creation of some geometric entities, for example lines, require other geometric entities, in this case points, to complete their definition. In cases like these, the points must be defined before the line is defined. In general, all geometric entities must be defined before they are used.

2.3.1 The POINT Macro

This macro is used to define or redefine a point in space. It is invoked by the statement:

POINT (GPOINT, OPTIONS, COORDS);

GPOINT is the geometric point being specified and COORDS is a three component array containing its coordinates. OPTIONS may contain SFLAG=CHAR where CHAR is a single character selection flag. A value of blank is used if SFLAG is not given. The use of this flag is for generating color animated films as described.
in Chapter III. Examples of POINT macros are:

DECLARE A(3) FLOAT BINARY;
POINT (PT(4), A);
POINT (Q, SFLAG='X', A);

2.3.2 The LINE Macro

The LINE macro may be used to define or redefine a straight line segment in space. It is invoked by:

LINE(GLINE, OPTIONS, GPOINT1, GPOINT2);

GLINE is the line being specified. It joins the points GPOINT1 and GPOINT2. OPTIONS is the same as in the POINT macro. An example of the LINE macro is:

LINE(L, SFLAG='A', PT(4), Q);

2.3.3 The SURFACE Macro

This macro is used to define or redefine a surface in space. The reason for the SURFACE macro is to define opaque areas for the hidden line processors to use. The hidden line processors can be used to delete any point or (partial) line segment which lies behind a surface. A surface may be bounded by as many as ten lines. However, the surface must form a convex, plane figure. The SURFACE macro is invoked by the statement:

SURFACE(GSURFACE, OPTIONS, GLINE1,..., GLINE10);

GSURFACE is the surface being defined and GLINE1,...,GLINE10 are its boundaries. OPTIONS may contain the keyword MINSIZE which will cause storage space for the surface to be minimized. It is not recommended that MINSIZE be used.

The order in which the boundary lines are given can be important. If the object being defined by the surfaces is a solid object, that is, it has an inside and an outside, then the lines should be ordered in a counterclockwise direction as seen from the outside. This will allow a more efficient hidden line removal algorithm to be used. Examples of the SURFACE macro are:

SURFACE (S, MINSIZE, L, M, N);
SURFACE (S,, LN(1), LN(2), LN(3), LN(4));
2.3.4 The OBJECT Macro

The OBJECT macro is used to group points, lines, and surfaces together as a single geometric entity. Up to ten geometric entities may be added to an object in a single OBJECT macro. The macro is invoked by the statement:

```
OBJECT(GOBJECT,OPTIONS,ENTITY1,...,ENTITY10);
```

GOBJECT is the object which is being defined, redefined, or expanded by the entities ENTITY1,...,ENTITY10. OPTIONS may contain the keyword EXPAND to indicate that ENTITY1,...,ENTITY10 are to be added to whatever is already in GOBJECT, otherwise the object is redefined to contain only ENTITY1,...,ENTITY10. Examples of OBJECT statements are:

```
OBJECT(OBJ,,PT(1),PT(2));
OBJECT(OBJ,EXPAND,LN(K+3),SU);
```

2.3.5 The VIEW Macro

This macro is used to define or redefine the camera position and the relative dimensions of the viewing area (either the display screen or the projection screen). The macro is invoked by the statement:

```
VIEW(GVIEW,OPTIONS,REFP,VDIR,HDIR,EYED,SCRD,SCRZ);
```

GVIEW is the view being defined and OPTIONS is not currently used. REFP, VDIR, and HDIR are each three component arrays and EYED, SCRD, and SCRZ are scalars. To understand the meaning of these arguments refer to Figure 2.1.

REFP is a reference point in space with the vector, VDIR, pointing toward the screen which is SCRD units away. The horizontal axis of the screen is given by the vector HDIR and the screen size is SCRZ. Finally, the distance between the eyes is given by EYED. EYED is only used in the 3-D interactive or movie making modes. One way to think of the geometry in Figure 2.1 is to think of REFP as the point midway between your eyes with VDIR a rod projecting forward with a screen attached to its far end. HDIR is then parallel to a line joining your eyes. The projection of the geometry onto the screen is what will appear on the display CRT. When using a 3-D mode, a point in space between REFP and the screen will appear to float in front of the CRT or motion picture screen.

For the interactive mode, the scalars EYED, SCRD, and SCRZ should be, roughly, in the ratio 1:12:4. This assumes an eye separation of 2 1/2
inches, 30 inches viewing distance and a 10 inch screen. This same ratio can be used in the movie modes also because in the 3-D case EYED is multiplied by 0.2 which gives reasonable dimensions for a small projection studio.

From the definition of HDIR, it is clear that it should be perpendicular to VDIR. If it is not, it will be adjusted so that it becomes perpendicular. In addition, HDIR is optional; if it is not given a vector is generated which is parallel to the X-Y plane.

2.3.6 The GPRINT Macro

This macro is primarily a debugging tool. It may be used to print a single geometric entity or all geometric entities. The macro is invoked by the statement:

GPRINT(OPTIONS);

OPTIONS may be either the keyword ALL or ENTITY=GVAR where GVAR is any geometric entity variable. Examples of the GPRINT macro are:

GPRINT(ALL);
GPRINT(ENTITY=PT(3));

2.4 Picture Generation Macros

The final group of macros to be described are the ones which actually generate the animated sequence.

2.4.1 The LEADER Macro

The LEADER macro should be used at the beginning and end of an animation sequence. This macro will initialize the system so that it is ready to generate animated sequences and, in addition, will produce a few frames of leader in the movie-making modes.

The macro is invoked by the statement:

LEADER;

This macro has no arguments.

2.4.2 The PICTURE Macro

Up to three OBJECT-VIEW pairs are supplied as arguments to this macro. The programming system will take each object, scan it to find all lines and points, remove from consideration any hidden items, and generate a display of the object using its associated view. Since three object-view pairs may be specified, the user may put up to three different objects on
one frame, or he may put up to three different views of the same object on a frame. However, the user should remember that hidden lines are removed independently from each object. This means that if one object hides another, the hidden lines will not be properly removed.

The macro is invoked by the statement:

\[ \text{PICTURE(GOBJECT1,GVIEW1,OPTION1,GOBJECT2,GVIEW2,} \]
\[ \text{OPTION2,GOBJECT3,GVIEW3,OPTION3);} \]

The arguments GOBJECT1,...,GOBJECT3 are geometric objects and GVIEW1,..., GVIEW3 are views. The argument OPTIONS1 applies to the GOBJECT1-GVIEW1 pair, and may contain multiple fields separated by ampersands. Possible fields are:

\text{NGC: This indicates that no geometry changes have been made} \]
\[ \text{since the last use of the PICTURE or LEADER macro.} \]

\text{RHL=CONVEX: Remove hidden lines under the assumption that the} \]
\text{object is a single convex body.} \]

\text{RHL=SOLID: Remove hidden lines under the assumption that the} \]
\text{object consists only of solid bodies; that is, the surfaces} \]
\text{define objects with an inside and an outside. The object} \]
\text{may also contain other miscellaneous points and lines and} \]
\text{these will be processed through the hidden line algorithms.} \]

\text{RHL=GENERAL: Remove hidden lines without making any simplifying} \]
\text{assumptions.} \]

Some examples of PICTURE macros are:

\[ \text{PICTURE(OB,VW);} \]
\[ \text{PICTURE(OB,VW,NGC \& RHL=SOLID);} \]
\[ \text{PICTURE(OB1,VW1,RHL=SOLID,OB2,VW2,RHL=GENERAL);} \]

\text{2.4.3 The REPEAT Macro} \]

In preparing an animation sequence for a motion picture, it is often required to have segments in the film with a static picture on the screen. This could be done by using the PICTURE macro repeatedly to generate the same picture a number of times. However, the PICTURE macro can imply a large amount of computation. Instead, the REPEAT macro may be used to repeat the last display generated by the PICTURE macro.

The macro is invoked by the statement:

\[ \text{REPEAT (NTIMES);} \]
The argument is the number of times the picture is to be repeated. Examples of this macro are:

\texttt{REPEAT (1);} \\
\texttt{REPEAT (I+J-1);} \\

2.5 Optimizing the Program

There are a number of things that a programmer can do to optimize a program in terms of both execution time and core requirements. The options described under the PICTURE macro represent important considerations in this respect. If hidden lines are being removed, NGC should be specified whenever possible. NGC will suppress the re-computation of surface equations. In addition, the simplest hidden line algorithm that will work on an object should be chosen. \texttt{RHL=CONVEX} uses a very fast, simple method. Unfortunately, most objects of interest are not convex. \texttt{RHL=SOLID} is much more complex than \texttt{RHL=CONVEX} but is still somewhat efficient. Finally, \texttt{RHL=GENERAL} is a slow algorithm and its use should be avoided, if possible.

2.5.1 Details of the \texttt{\%OPTIONS=} Statement

There are many fields, separated by ampersands, in addition to those described in Section 2.1.1 that may appear in the \texttt{\%OPTIONS=} statement. These fields will now be described.

\texttt{NODISP:} This field indicates that the Data Entry Macros will not be used in this procedure. This keyword, and the following two, will reduce compilation time by reducing the number of macros that must be scanned for.

\texttt{NOGEO:} This field indicates that the Geometric Definition Macros, except possibly \texttt{VIEW}, will not be used in this procedure.

\texttt{NOPIC:} This field indicates that the Picture Generation Macros will not be used in this procedure. The \texttt{VIEW} macro is available unless both \texttt{NOGEO} and \texttt{NOPIC} are specified.

Note: The preceding fields are valid only in main programs.

\texttt{RHL=} The right-hand side of the equal sign may contain any of \texttt{NONE}, \texttt{CONVEX}, \texttt{SOLID}, or \texttt{GENERAL} separated by commas. If \texttt{RHL=NONE} is specified, this will prevent the hidden line algorithms from becoming part of the final load module. The other values will cause only the necessary parts of the algorithm to be loaded.
NAREA=: This specifies the length (in bytes) of an AREA variable that is allocated to hold the geometric data structure. If a programmer underestimates this value, an error message will appear on the screen and the console operator will be able to increase the number and try again. A method of estimating this number is given in Appendix A.

NELEM=: This parameter specifies the length of a string where the display will be constructed. It, like NAREA, may be corrected by the console operator if it is incorrectly estimated in the program. It is more difficult to give a means of estimating NELEM. The following two hints may help: (1) the maximum number of bytes that a line segment can require is 8 bytes. (2) One thousand bytes can hold a quite complicated picture.

NENT=, NSET=, NAUXL=: These parameters specify values which are passed to procedure IDOPEN (see CGTM No. 80). The programmer will not normally have to use anything but the default values.

Examples of valid %OPTIONS= statements are:

%OPTIONS= 'MAIN & NAREA=5000 & NOPIC';
%OPTIONS= 'MAIN & RHL=CONVEX, SOLID & NELEM=750 & NAREA=10000 & NODSP';

2.5.2 Overlay Structures

The macros that have been described expand into one or more PL/1 statements. Many of these macro expansions contain calls to other procedures which are selected from program libraries which must be made available to the language programs, all of these procedures are themselves written in PL/1. Because of this, a programmer can easily find that he has a problem fitting his program into core. The use of program overlays is needed in this case.

Fortunately, a program written in this programming system divides itself quite naturally into three sections, (1) the entry of parameters from the console, (2) the definition of the geometry, and (3) the generation of an animation sequence. In addition, there is also an internal initialization section. These four sections can easily overlay each other. When overlaying of programs does become necessary, the user should study his expanded program to determine the names of procedures that can be overlain.
2.6 Future System Development

This system, as it now exists, contains all of the basic elements of an animation generation system. However, there are some things which could be done more easily if there were certain additions to the system. Some of these possible additions will now be discussed.

In the first place, additional macros are needed to give the programmer a greater ability to manipulate the data structure. In particular, the programmer should be able to delete an entity from the data structure. This addition offers no problem.

Another group of additions concerns new geometric data types in addition to points, lines, etc. A data type representing a curve would be very useful. Such a data type would represent a sequence of concatenated lines. As long as a curve was not allowed to be a surface boundary, this would be a straightforward addition. Other data types should be supplied which would allow the programmer to put text messages into an animation sequence. These could include both characters produced by the IDIOM character generator, and text composed of concatenated straight line segments. The latter character types could be of arbitrary size.

Finally, more work needs to be done on the hidden line eliminator to improve its efficiency. As explained in Appendix C, a totally hidden line is usually recognized as such quite easily, but a totally visible line is recognized as such only after considerable calculation. A successful attack on the problem of quickly recognizing totally visible lines could be very fruitful.
Figure 2.1. The Geometry of the VIEW Entity
CHAPTER III

RUNNING THE PROGRAM

The preceding chapter has described how a programmer writes a program in this programming system. This chapter will explain how to operate the program from the display console.

Although a program, at first examination, appears to generate the animation sequences only once, this is not the case. Actually, part of the expansion of the `\%INCLUDE TAIL;` statement is a GO TO which transfers control into part of the `\%INCLUDE HEAD2;` statement. Because of this, the animation sequence may be run through many times, perhaps with different parameters, during the time a program is running.

When a program begins executing, the first thing that happens is the generation of the Mode Selection Display as shown in Figure 3.1. The console operator may use the light pen to select the mode in which he wishes to run. By pointing the light pen at one of the mode messages and pressing the light pen switch, the operator will cause the program to switch modes and move the designator box in the display.

In addition, the keyboard may be used to change the three parameters on the screen. The first two parameters are the values of NAREA and NELEM, as described in Section 2.5.1. The third parameter consists of four characters. These selection flags are compared to the selection flags of the individual lines and points. A line or point becomes part of the display according to the following tests:

1. If the selection flags of the Mode Selection Display are all blanks, then all lines and points (except those in 3 below) are displayed.
2. Otherwise, a line or point is displayed only if its selection flag appears as one of the four characters in the Mode Selection Display.
3. Lines or points with a selection flag of '${}' are never displayed. The primary purpose of this is to create non-convex surfaces out of convex patches and keep the join line invisible.

The purpose of the selection flags is the creation of color movies. If the lines in an object have selection flags of either 'A' or 'B' then the console operator could generate one film strip containing only 'A' images, and another strip containing only 'B' images. These two film strips could then be printed, through
color filters, onto the same film to produce a color film strip.

When an animation sequence has been completed, the Mode Selection Display always returns to the screen. In addition, if an error occurs, the Mode Selection Display will return to the screen with an error message. An example of this is shown in Figure 3.2.

After the console operator has completed making all the choices available to him in this display, he may go on to the next display by pushing the escape key on the keyboard. This second display, the Auxiliary Parameter Display, may take two possible modes depending on the current mode as shown in Figures 3.3 and 3.4. In the interactive mode, the operator is given the option of changing the length of a programmed delay that is performed between picture calculations. The purpose of this is to control the speed of the animation. In a movie-making mode, two parameters are presented for possible modification. The first requests the number of display regenerations which should be done for each frame. Values of one to six have been used. The second parameter is the number of frames of leader which will be generated by the LEADER macro. This display is also terminated by the escape key on the keyboard.

The final topic to be covered here is the function keyboard. The lower row of buttons consists of buttons Number 25 through 32. These buttons are used as follows:

25. In the interactive-single step mode this button will cause the picture to advance one frame.

26. In the interactive-single step mode this button will temporarily put the program into the continuous mode. The speed that the frames will advance is controlled by the delay parameter that was entered in the Auxiliary Parameter Display.

27. This button undoes the work of Button 26 and puts the program back into single step mode.

31. This will cause a return to the Mode Selection Display at the first convenient opportunity.

32. This is a panic button which will always return immediately to the Mode Selection Display except when already in that display, in which case the program terminates.
FIGURE 3.1  Mode Selection Display (Initial Entry)

FIGURE 3.2  Mode Selection Display (With Error Indication)
SECONDARY OPTIONS SELECTION DISPLAY

SECONDARY OPTIONS SELECTION DISPLAY

SECONDARY OPTIONS SELECTION DISPLAY

SECONDARY OPTIONS SELECTION DISPLAY

FIGURE 3.3 Auxiliary Parameter Display (Interactive Modes)

FIGURE 3.4 Auxiliary Parameter Display (Movie-making Modes)
CHAPTER IV

A PROGRAMMING EXAMPLE: THE STELLATED DODECAHEDRON

This chapter contains a listing of a complete program written in this system. The program first requests that the console operator verify or change some parameters. Then a geometric figure, a stellated dodecahedron, is defined and the camera is rotated about the figure.

Except for the initialized arrays at the beginning of the program, it is quite simple and self-explanatory. The arrays PX, PY, and PZ are the X, Y, and Z coordinates of all of the vertices in the figure. Arrays L1 and L2 define the lines in the figure. Thus, the first line is the one between point 5 and point 1. The first 90 lines are the edges of the stellated dodecahedron and the last 30 lines form a circumscribed icosahedron. The arrays S1, S2, and S3 define the triangular surfaces. For example, the first surface is bounded by lines 1, 35, and 31. The manner in which these arrays were generated will not be discussed.

Figures 4.1 and 4.2 show the two interactive displays as they appear on the CRT. Figures 4.3 and 4.4 are examples of the stellated dodecahedron in two positions without the circumscribed icosahedron, while Figures 4.5 and 4.6 show the same positions with the circumscribed icosahedron. These last two figures are confusing to the eye; however, if the object is viewed in 3-D or in motion, the confusion is gone. The expanded program, after the PL/1 pre-processor has completed its task, is shown in Appendix D.

Because of the large amount of calculation that must be done to eliminate hidden lines, this program takes approximately two seconds on the IBM 360/model 91 to generate the displays shown in Figures 4.3 and 4.4, while Figures 4.5 and 4.6 require almost six seconds. When hidden lines are not being removed, the program will run at a rate of many frames per second.
**** STELLED DODECAHEDRON GENERATOR ****/

EXAMPLE: PROCEDURE OPTIONS(MAIN);

/* THIS PROGRAM PRODUCES AN ANIMATED SEQUENCE SHOWING A STELLED DODECAHEDRON ROTATING IN SPACE. HIDDEN LINES IN THE OBJECT ARE REMOVED. A STELLED DODECAHEDRON IS THE SOLID FIGURE WHICH IS GENERATED BY EXTENDING THE FACES OF A REGULAR DODECAHEDRON. */

*INCLUDE 'HEAD1';
*OPTIONS = "MAIN & NELEM=750 & NAREA=11000 & RHL=SOLID ";
*GEOMETRY = " POINTS=PT(32) & LINES=LN(120) & SURFACES=SU(60) & OBJECTS=OB & VIEWS=VW ";

DECLARE FL ENTRY(CHARACTER(6),FLOAT BINARY);
DECLARE FX ENTRY(CHARACTER(6),FIXED BINARY);
DECLARE PXI(32) FLOAT BINARY STATIC INITIAL( 0.2764,-0.1056,-0.3416,-0.1056, 0.2764, 0.5528, 0.1708,-0.4472,
-0.4472, 0.1708, 0.4472,-0.1708,-0.5528,-0.1708, 0.4472, 0.3416,
 0.1056,-0.2764,-0.2764, 0.1056, 0.0000, 0.8944,-0.2764,-0.7236,
-0.7236, 0.2764,-0.7236,-0.2764,-0.8944,-0.2764, 0.7236, 0.0000);

DECLARE PLY(32) FLOAT BINARY STATIC INITIAL( 0.2008, 0.3249, 0.0000,-0.3249,-0.2008,-0.0000, 0.5257, 0.3249,
-0.3249,-0.5257, 0.3249, 0.5257, 0.0000,-0.5257,-0.3249, 0.0000,
 0.3249, 0.2008,-0.2008,-0.3249, 0.0000, 0.0000, 0.8507, 0.5257,
-0.5257,-0.8507, 0.5257, 0.8507, 0.0000,-0.8507,-0.5257, 0.0000);

DECLARE PZI(32) FLOAT BINARY STATIC INITIAL( -0.4472,-0.4472,-0.4472,-0.4472,-0.4472, 0.1056, 0.1056,-0.1056,
 0.1056,-0.1056,-0.1056,-0.1056,-0.1056,-0.1056,-0.1056,
-0.4472,-0.4472,-0.4472,-0.4472,-1.0000,-0.4472,-0.4472,-0.4472,
-0.4472,-0.4472, 0.4472, 0.4472, 0.4472, 0.4472, 0.4472, 1.0000);

DECLARE LI(120) FIXED BINARY STATIC INITIAL( 5, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 11, 7, 12, 8, 13, 9, 14, 10, 15,
 6, 7, 8, 9, 10, 16, 17, 18, 19, 20, 21, 21, 21, 22, 22, 22, 22,
 23, 23, 23, 23, 24, 24, 24, 24, 24, 25, 25, 25, 25, 26, 26, 26, 26,
 27, 27, 27, 27, 28, 28, 28, 28, 28, 29, 29, 29, 29, 30, 30, 30, 30,
 31, 31, 31, 31, 32, 32, 32, 32, 32, 21, 21, 21, 21, 22, 22, 23, 24,
 24, 25, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
 22, 27, 23, 28, 24, 29, 25, 30, 26, 31, 31, 31, 27, 28, 29, 30, 27, 28,
 29, 30, 30, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32);

DECLARE L2(120) FIXED BINARY STATIC INITIAL( 1, 2, 3, 4, 5, 11, 12, 13, 14, 15, 11, 7, 12, 8, 13, 9, 14, 10, 15,
 6, 16, 17, 18, 19, 20, 17, 18, 19, 20, 16, 1, 2, 3, 4, 5, 1, 11, 6, 15,
 1, 2, 12, 7, 11, 2, 3, 13, 8, 12, 3, 4, 14, 9, 13, 4, 5, 15, 10, 14,
 16, 17, 7, 11, 6, 17, 18, 8, 12, 7, 18, 19, 9, 13, 8, 19, 20, 10, 14, 9,
 20, 16, 6, 15, 10, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 23, 24, 25,
 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
 27, 23, 28, 24, 29, 25, 30, 26, 31, 22, 27, 28, 29, 30, 31, 32, 32, 32,
 32, 32, 32, 32, 32, 32, 32, 32);

DECLARE SI(60) FIXED BINARY STATIC INITIAL( 1, 2, 3, 4, 5, 1, 6, 11, 20, 10, 2, 7, 13, 12, 6, 3, 8, 15, 14, 7,
 4, 9, 17, 16, 8, 5, 10, 19, 18, 9, 11, 12, 22, 26, 21, 13, 14, 23, 27, 22,
 15, 16, 24, 28, 23, 17, 18, 25, 29, 24, 19, 20, 21, 30, 25, 26, 27, 28, 29, 30);

DECLARE S2(60) FIXED BINARY STATIC INITIAL( 35, 31, 32, 23, 34, 37, 38, 39, 40, 36, 42, 43, 44, 45, 41, 47, 48, 49, 50, 46,
 52, 53, 54, 55, 51, 57, 58, 59, 60, 56, 64, 63, 62, 61, 65, 69, 68, 67, 66,
 74, 73, 72, 71, 75, 79, 78, 77, 76, 80, 84, 83, 82, 81, 85, 87, 88, 89, 90, 86);

DECLARE S3(60) FIXED BINARY STATIC INITIAL( 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 65, 64, 63, 62, 61, 70, 69, 68, 67, 66,
 75, 74, 73, 72, 71, 80, 79, 78, 77, 76, 85, 84, 83, 82, 81, 86, 87, 88, 89, 90);
DECLARE FG(120) CHARACTER(1) STATIC INITIAL(
   (30)(1)'A',(60)(1)'B',(30)(1)'C');
DECLARE DOBC CHARACTER(6) INITIAL('9.00');
DECLARE DSCR CHARACTER(6) INITIAL('10.00');
DECLARE EYES CHARACTER(6) INITIAL('0.70');
DECLARE SCRZ CHARACTER(6) INITIAL('2.50');
DECLARE FPIC CHARACTER(6) INITIAL('24');
DECLARE MPIC CHARACTER(6) INITIAL('240');
DECLARE NFPP CHARACTER(6) INITIAL('1');
DECLARE RSTP CHARACTER(6) INITIAL('1.00');
DECLARE (XDOBC,XDSCR,XEYES,XSCRZ,XRSTP) FLOAT BINARY;
DECLARE (XFPI,CMPIC,XNFP) FIXED BINARY;
DECLARE (V1,V2,V3) (3) FLOAT BINARY;
DECLARE (ARGS,C) FLOAT BINARY;
DECLARE (I,J) FIXED BINARY;
*INCLUDE HEAD2;

/* TWO INTERACTIVE DISPLAYS ARE CREATED TO ALLOW THE CONSOLE
OPERATOR TO ENTER PARAMETERS THROUGH THE KEYBOARD. THE FIRST
DISPLAY GIVES HIM THE OPTION OF CHANGING PARAMETERS ASSOCIATED
WITH CAMERA POSITIONING WHILE THE SECOND DISPLAY LETS HIM
CHANGE PARAMETERS WHICH CONTROL THE NUMBER OF FRAMES WHICH WILL
BE PRODUCED. */

DSP1: DSPINIT;
DSPTEXT(190,700,CPARM=VLRG,'ENTER CAMERA PARAMETERS');
DSPTEXT(260,600,VSPAC=-35,
   'DISTANCE TO CENTER OF OBJECT ',
   'DISTANCE TO THE SCREEN ',
   'EYE SEPARATION (STEREO ONLY) ',
   'SCREEN SIZE ');
DSPKBRD(68C,600,DOBC,DSCR,EYES,SCZ);
ON CONVERSION GO TO DSP1;
CALL FL(DOBC,XDOBC); CALL FL(DSCR,XSCP);
CALL FL(EYES,XEYES); CALL FL(SCZ,XCRZ);

DSP2: DSPINIT;
DSPTEXT(190,700,CPARM=VLRG,'ENTER MOTION PARAMETERS');
DSPTEXT(260,600,VSPAC=-35,
   'START/END FIXED PICTURES ',
   'NUMBER OF MOVING PICTURES ',
   'NUMBER OF FRAMES/PICTURE ',
   'ROTATION (DEGREES/PICTURE) ');
DSPKBRD(68C,600,FPIC,MPIC,NFPP,RSTP);
ON CONVERSION GO TO DSP2;
CALL FX(FPIC,XPIC); CALL FX(MPIC,XPIC);
CALL FX(NFPP,XNFPP); CALL FX(RSTP,XRSTP);

/* THE GEOMETRY IS NOW DEFINED. THE ODODECAHEDRON ITSELF HAS A
SELECTION FLAG OF 'A'. THE STELLATION STRUCTURE HAS A SELECTION
FLAG OF 'B'. FINALLY THE POINTS OF THE STELLATED ODODECAHEDRON
ARE JOINED TOGETHER WITH LINES WHICH FORM AN ICOSAHEDRON AND
HAVE A SELECTION FLAG OF 'C'. */
DO I=1 TO HBOUND(PT,1);
   V1(1)=PX(I); V1(2)=PY(I); V1(3)=PZ(I);
   POINT(PT(I),,V1);
END;
DO I=1 TO HBOUND(LN,1);
  LINE(LN(I),SFLAG=FG(I),PT(L1(I)),PT(L2(I)));
  IF FG(I)="C" THEN OBJECT(OB,EXPAND,LN(I));
END;
DO I=1 TO HBOUND(SU,1);
  SURFACE(SU(I),LN(S1(I)),LN(S2(I)),LN(S3(I)));
  OBJECT(OB,EXPAND,SU(I));
END;

/* THE ANIMATED SEQUENCE IS NOW PRODUCED. THE PARAMETERS WHICH
   WERE ENTERED THROUGH THE KEYBOARD ARE USED AT THIS TIME. */
LEADER;
DO I=1 TO XMPIC;
  ARG=(I-1)*XRSTP; S=SIND(ARG); C=COSD(ARG);
  V2(1)=-C*C; V2(2)=S*C*(1+S); V2(3)=S*(C*C-S);
  V3(1)=S*C; V3(2)=C*C-S*S*S; V3(3)=-S*C*(1+S);
  V1=-XDBBC*V2;
  VIEW(WV,1,V1,V2,V3,XYES,XDSCR,XSCRZ);
  PICTURE(OB,WV,NGC&RHL=SOLID);
  IF (I=1) OR (I*XMPIC) THEN J=XPIC*XNFPP; ELSE J=0;
  REPEAT(J*XNFPP=1);
END;
LEADER;

/* THIS INTERNAL PROCEDURE IS USED TO CONVERT CHARACTER STRINGS
   TO FIXED OR FLOAT BINARY NUMBERS. THE REASON FOR THE UNUSUAL
   CODE IS TO FORCE ALL POSSIBLE ERRORS TO RAISE THE CONVERSION
   CONDITION. */
FL: PROCEDURE(XCH,XFL);
  DECLARE XCH CHARACTER(6);
  DECLARE XFL FLOAT BINARY;
  DECLARE XFX FIXED BINARY;
  DECLARE XTM CHARACTER(6) INITIAL("XXXXX X");
  DECLARE FLAG FIXED BINARY;
    FLAG=0;
    GO TO XXS1;
FX:  ENTRY(XCH,XFX);
    FLAG=1;
XXS1:  SUBSTR(XTM,1,6)=XCH;
    IF FLAG=0 THEN GET STRING(XTM) LIST(XFL);
    ELSE GET STRING(XTM) LIST(XFX);
    RETURN;
END FL;

%INCLUDE TAIL;

END EXAMPLE;
ENTER CAMERA PARAMETERS

DISTANCE TO CENTER OF OBJECT 5.00
DISTANCE TO THE SCREEN 10.00
EYE SEPARATION (STANDARD ONLY) 0.70
SCREEN SIZE 2.50

FIGURE 4.1 First Interactive Display

ENTER MOTION PARAMETERS

START/END FIXED PICTURES 24
NUMBER OF MOTIONS PICTURES 240
NUMBER OF FRAMES/PUPC 4
ROTATION (DEGREES/PIC) 1.00

FIGURE 4.2 Second Interactive Display
FIGURE 4.3  The Stellated Dodecahedron (I)
(without circumscribed Icosahedron)

FIGURE 4.4  The Stellated Dodecahedron (II)
(without circumscribed Icosahedron)
FIGURE 4.5 The Stellated Dodecahedron (I) 
(with circumscribed Icosahedron)

FIGURE 4.6 The Stellated Dodecahedron (II) 
(with circumscribed Icosahedron)
APPENDIX A

THE GEOMETRIC DATA STRUCTURE

The Geometric Data Structure used by this programming system consists of a number of different CONTROLLED variables which may be allocated in a single AREA variable. Each time the POINT, LINE, SURFACE, OBJECT, or VIEW macro is used to define a new geometric entity, a CONTROLLED variable is allocated and initialized. The corresponding geometric variable will then contain a pointer to the CONTROLLED variable.

The declarations for the complete set of CONTROLLED variables is shown in Figure A.1. All of the entities (except the auxiliary surface and node entities) start out with four characters of identification and flags, and follow with a pointer called MLGEOML. MLGEOML is the link in the master list of all entities. Each entity points to the next entity until the last entity which contains a NULL value in MLGEOML. In addition, most entities also contain pointer variables called MLUSESL and MLUSEDL. These variables are the heads of rings of entities as shown in the example in Figure A.2. The MLUSESL ring contains all entities which the entity uses to complete its specification. Thus, the MLUSESL ring of a line entity contains its end points. Similarly, the MLUSEDL ring contains all entities that the given entity is used by. Thus, the MLUSEDL ring of a point will contain the lines of which it is an end point (and possibly the objects which contain the point directly). These rings are always linked together by means of NODE entities. Other items in the geometric entities are described in Figure A.1.

The advantages of a ring structure similar to the one shown in Figure A.2 is its generality and completeness. If one starts with a pointer to any geometric entity, it is possible to determine exactly how that entity fits into the overall geometry of the object. For example, if one starts with a line, it is a basically simple problem to find its end points or to find which surfaces it bounds. The use of node entities to connect everything together is important to the generality of the structure. Thus, the use of node entities means that a single point may be an end point of an arbitrary number of lines.
A set of macro statements which would have generated the structure shown in Figure A.2 are:

POINT (Pt1, V1);
POINT (Pt2, V2);
POINT (Pt3, V3);
POINT (Pt4, V4);
LINE (Ln1, Pt1, Pt2);
LINE (Ln2, Pt2, Pt3);
LINE (Ln3, Pt3, Pt4);
SURFACE (Su1, Ln1, Ln2, Ln3);
OBJECT (Ob1, Su1, Ln1, Pt1);

Note that the surface in this example is not correctly defined because the boundary does not form a closed polygonal curve. An attempt to use it in one of the hidden line algorithms will cause unpredictable results. The definition of the surface would be correct if the symbol Pt4 were replaced by Pt1 everywhere it occurs.

From Figures A.1 and A.2, it is possible to compute the number of bytes of storage needed in the AREA variable to contain a specific data structure. The number of bytes needed for each geometric entity is shown below.

<table>
<thead>
<tr>
<th>Type of Entity</th>
<th>Number of Bytes Needed in the AREA Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>24 + 8 N</td>
</tr>
<tr>
<td>Line</td>
<td>16 + 8 N</td>
</tr>
<tr>
<td>Surface (MINSIZE)</td>
<td>40 + 8 N</td>
</tr>
<tr>
<td>Surface</td>
<td>48 + 8 N + 4 M</td>
</tr>
<tr>
<td>Object</td>
<td>16 + 8 N</td>
</tr>
<tr>
<td>View</td>
<td>72</td>
</tr>
</tbody>
</table>

In this table, N is the number of containing entities and M is the number of surface vertices rounded up to an even number.
DECLARE 1 MLPNTX BASED(MLGEO1), /* POINT ENTITY */
  2 MLIDENT CHARACTER(4), /* POINT IDENTIFICATION */
  2 MLGEOML POINTER, /* MASTER GEOMETRY LIST */
  2 MLUSEDL POINTER, /* USED ENTITY RING */
  2 MLPTXYZ(3) FLOAT BINARY; /* COORDINATES OF POINT */

DECLARE 1 MLINRX BASED(MLGEO1), /* LINE ENTITY */
  2 MLIDENT CHARACTER(4), /* LINE IDENTIFICATION */
  2 MLGEOML POINTER, /* MASTER GEOMETRY LIST */
  2 MLUSEDL POINTER, /* USED ENTITY RING */
  2 MLUSESL POINTER; /* USES ENTITY RING */

DECLARE 1 MLSURX BASED(MLGEO1), /* SURFACE ENTITY */
  2 MLIDENT CHARACTER(4), /* SURFACE IDENTIFICATION */
  2 MLGEOML POINTER, /* MASTER GEOMETRY LIST */
  2 MLUSEDL POINTER, /* USED ENTITY RING */
  2 MLUSESL POINTER, /* USES ENTITY RING */
  2 MLTEMPL POINTER, /* TEMPORARY SURFACE LIST */
  2 MLEQUTX(4) FLOAT BINARY, /* SURFACE EQUATION */
  2 MLAUXLX POINTER; /* AUXILIARY DATA POINTER */

DECLARE 1 MLSURY BASED(MLGEO1), /* AUXILIARY SURFACE ENTITY */
  2 MLNARRY FIXED BINARY, /* VERTEX COUNT + 1 */
  2 MLLARRAY(MLGEOM2 REFER(MLSURY,MLNARRY)) POINTER; /* POINTERS TO VERTICES */

DECLARE 1 MLOBJX BASED(MLGEO1), /* OBJECT ENTITY */
  2 MLIDENT CHARACTER(4), /* OBJECT IDENTIFICATION */
  2 MLGEOML POINTER, /* MASTER GEOMETRY LIST */
  2 MLUSEDL POINTER, /* USED ENTITY RING */
  2 MLUSESL POINTER; /* USES ENTITY RING */

DECLARE 1 MLVEWX BASED(MLGEO1), /* VIEW ENTITY */
  2 MLIDENT CHARACTER(4), /* VIEW IDENTIFICATION */
  2 MLGEOML POINTER, /* MASTER GEOMETRY LIST */
  2 MLREFPX(3) FLOAT BINARY, /* REFERENCE POINT */
  2 MLVDIRX(3) FLOAT BINARY, /* VIEW DIRECTION */
  2 MLHDIRX(3) FLOAT BINARY, /* SCREEN HORIZONTAL DIR. */
  2 MLUDIRX(3) FLOAT BINARY, /* SCREEN UPWARD DIR. */
  2 MLVEYDX FLOAT BINARY, /* EYE SEPARATION */
  2 MLSCRDX FLOAT BINARY, /* DISTANCE TO SCREEN */
  2 MLSCRZX FLOAT BINARY; /* SCREEN SIZE */

DECLARE 1 MLNODX BASED(MLGEO1), /* NODE ENTITY */
  2 MLUSEDL POINTER, /* USED ENTITY CONNECTOR */
  2 MLUSESL POINTER; /* USES ENTITY CONNECTOR */

FIGURE A.1: Declarations of the Geometric Entities
FIGURE A.2: An Example of a Data Structure
APPENDIX B

THE PROJECTION MATRIX

One of the basic operations which must be done repeatedly is the operation of transforming a point \((X,Y,Z)\) in the model space into screen coordinates \((\mu,\nu)\). In the programming system the data supplied to the \texttt{VIEW} macro is first used to construct a 3x4 matrix \(M\). This matrix has the property that the coordinate transformation may be carried out by:

\[
\begin{pmatrix}
\mu \\
\nu \\
1
\end{pmatrix} = M
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix} \quad (B.1)
\]

Thus, the problem of transforming a point in model space to screen coordinates is reduced to a matrix multiplication and two divisions. The remainder of this appendix will describe a method of constructing the matrix \(M\).

As shown in Figure B.1, the data supplied to the \texttt{VIEW} macro (or easily computed from this data) are:

- \(R\) - A reference point which is conceptually a point midway between the eyes of the viewer.
- \(V\) - A unit vector in the direction of the view.
- \(H\) - A unit vector defining the horizontal direction of the projection screen.
- \(U\) - A unit vector defining the upward direction on the projection screen.
- \(d\) - The distance from the projection screen to \(R\).
- \(e\) - The distance between the eyes of the viewer.
- \(f\) - The size of the projection screen.

In addition, \(E\) is an eye point, \(P\) is a point in model space, and \(Q\) is the projection of \(P\) onto the projection plane.
From Figure B.1, it is evident that $E$ and $Q$ may be given by:

$$
E = R + \frac{e}{2} H \\
Q = R + d V + \xi H + \eta U
$$

(B.2)

Where $(\xi, \eta)$ is related to $(\mu, \nu)$ by:

$$
\xi = \frac{f}{1024} (\mu-512) \\
\eta = \frac{f}{1024} (\nu-512)
$$

(B.3)

Since $E$, $P$, and $Q$ lie on a straight line, the vectors $(P-E)$ and $(Q-E)$ are parallel. Because these vectors are parallel, they are proportional. Therefore:

$$
\lambda (Q-E) = (P-E)
$$

(B.4)

Substituting equations B.2 into the left hand side of equation B.4 results in:

$$
\lambda \left[ (\xi - \frac{e}{2}) H + \eta U + d V \right] = (P-E)
$$

(B.5)

Equation B.5 may be rewritten in the form:

$$
\lambda M_1 \begin{pmatrix}
\xi - \frac{e}{2} \\
\eta \\
d
\end{pmatrix} = (P-E)
$$

(B.6)

$$
M_1 = (H\ U\ V)
$$

Where the columns of $M_1$ are $H$, $U$, and $V$. Now observe that the $3x3$ matrix $M_1$ is orthogonal. Therefore, its inverse is equal to its transpose. Thus,
multiplication of equation B.6 by the inverse of \( M_1 \) gives:

\[
\lambda \left( \begin{array}{c}
\frac{\mu}{2} - \frac{f + e}{2} \\
\eta \\
d
\end{array} \right) = M_2 (P-E) \\
M_2 = \begin{pmatrix}
H^T \\
U^T \\
V^T
\end{pmatrix}
\] (B.7)

where the rows of \( M_2 \) are \( H^T, U^T, \) and \( V^T \). Now substitute equations B.3 into equations B.7 to obtain:

\[
\lambda \left( \begin{array}{c}
\frac{f}{1024} \\
\frac{\mu}{2} \\
\eta \\
\frac{f}{1024} \\
d
\end{array} \right) = M_2 (P-E) \\
\] (B.8)

To simplify future computation, replace the indeterminant \( \lambda \) by \( 1024 \lambda / f \). When this is done, equation B.8 becomes:

\[
\lambda \left( \begin{array}{c}
\mu - \frac{512(f+e)}{f} \\
-512 \\
\frac{1024d}{f}
\end{array} \right) = M_2 (P-E) \\
\] (B.9)

The column vector on the left hand side of equation B.9 may be written as:

\[
\begin{pmatrix}
1 & 0 & -\frac{512(f+e)}{f} \\
0 & 1 & -512 \\
0 & 0 & \frac{1024d}{f}
\end{pmatrix}
\begin{pmatrix}
\mu \\
\nu \\
1
\end{pmatrix}
\] (B.10)
Now multiply equation B.9, on the left, by the inverse of the 3x3 matrix in equation B.10. The result of this is:

\[
\lambda \begin{pmatrix} \mu \\ v \\ 1 \end{pmatrix} = M_3 (P-E)
\]

\[
M_3 = \begin{pmatrix}
1 & 0 & \left(\frac{f+e}{2d}\right) \\
0 & 1 & \left(\frac{f}{2d}\right) \\
0 & 0 & \left(\frac{f}{1024d}\right)
\end{pmatrix}
M_2
\]  
(B.11)

Finally, the first of equations B.11 may be written as equation B.1 where

\[
M = (M_3 \mid (-M_3E))
\]  
(B.12)

Equations B.12 states that the first three columns of \( M \) are the same as those of \( M_3 \) and the last column of \( M \) is the column vector \(-M_3E\).
FIGURE B.1: Projecting a Point onto the Display Screen
APPENDIX C

SOME DETAILS ON PICTURE GENERATION AND THE REMOVAL OF HIDDEN LINES

First of all, it must be stated that no great novelty is claimed for the hidden line algorithm. It is a simple, straightforward algorithm whose utility is due to the fact that it is incorporated into a complete movie-making facility.

The expansion of the PICTURE macro contains calls to the procedure MLPROC to process a geometric object and produce display orders for the IDIOM. An argument in the calling sequence to MLPROC specifies if hidden line removal is to occur and if so, which algorithm is to be used. In addition, if hidden lines are being removed, a call to the procedure MLFXUP will precede the calls to MLPROC.

Procedure MLFXUP scans the entire data structure and recomputes the equations of any surfaces which have been changed since the last time MLFXUP was called. It also fills in any auxiliary surface entities.

Procedure MLPROC is the heart of the picture generation scheme. Briefly, the manner in which it processes the data available to it is as follows:

1. The transformation matrix of Appendix B is generated.
2. If hidden lines are to be removed, MLPRC1 is called to form a list of all surfaces in the object.
3. For each line or point in the object, the following operations are performed:
   a. If it is a line and RHL=CONVEX or SOLID, then MLPRC2 is called to check for topological invisibility.
   b. If RHL=SOLID or GENERAL, then MLPRC3 is called to check for geometric invisibility.

If the line or point passes all tests, it is put into the display buffer.

Procedure MLPRC1 forms a list of all surfaces in the object and marks them as being front surfaces or back surfaces. A surface is a front surface if the eye point is on the side of the surface that was designated as the outside in the SURFACE macro.

Procedure MLPRC2 will make a line invisible if it is a boundary of two back surfaces or if it is a boundary of a back surface and a front surface with the back surface closer to the eye point.
The method that is used in this second case is illustrated in Figures C.1 and C.2. In Figure C.1, the common edge of the two surfaces is invisible while in Figure C.2 it will be visible if no other surface hides it. Let F and B be the outward surface normals of the front and back surfaces respectively. Let E be a vector along the edge which takes on the counter-clockwise direction of the front surface. When this is done, the triple scalar product \((F \times B \cdot E)\) is negative in Figure C.1 and positive in Figure C.2.

Procedure MLPRC3 takes a line segment supplied to it and exhaustively checks it against all surfaces. If RHL=SOLID, it must only check it against front surfaces. The line segment may be either accepted as visible, rejected as invisible, or split into two segments. When a line segment is split, MLPROC will again call MLPRC3 with each of the individual pieces.

The algorithm that is used to compare a single line segment against a single surface is shown in Figure C.3. It works by breaking the line segment up so that type 6 through 8 segments are reduced to shorter lines of type 1 through 5. The flow chart in Figure C.3 isolates two problems whose solution is not immediately obvious. These problems are:

1. In boxes labeled 2 and 4 it is necessary to decide if a point is within the boundary of the surface as viewed from the eye point.
2. In the box labeled 6 it is necessary to compute the intersection of the surface boundary and a line segment as viewed from the eye point.

The means used to solve these two problems will now be described:

Figure C.4 illustrates the first of these problems. Form an infinite pyramid with its apex at E, the eye point, extending out toward the vertex points of the surface, \(V_1, V_2, \ldots\). The problem is to determine if a point \(P\) lies inside this pyramid. Consider the fourth order determinant:

\[
\begin{vmatrix}
E & V_1 & V_2 & P \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{vmatrix}
\]

where each column is formed by using the three components of the point and a one in the fourth position. It is well known that the absolute value of determinant C.1 is six times the volume of the tetrahedron with vertices at \(E, V_1, V_2,\) and \(P\). In addition, the sign of this determinant gives the orientation of the four points. If one moves the point \(P\) from one side of the plane determined by \(E, V_1,\) and \(V_2\) to the other, then the sign of the determinant changes. This problem is, therefore,
solved by examining the three determinants (for a triangular surface):

\[
\begin{vmatrix}
E & V_1 & V_2 & P \\
1 & 1 & 1 & 1 \\
\end{vmatrix}, \quad 
\begin{vmatrix}
E & V_2 & V_3 & P \\
1 & 1 & 1 & 1 \\
\end{vmatrix}, \quad 
\begin{vmatrix}
E & V_3 & V_1 & P \\
1 & 1 & 1 & 1 \\
\end{vmatrix}
\] (C.2)

and checking to see if they are all of the same sign. Note that it is at this stage that the convexity of the surface is used. The problem is much more difficult for non-convex surfaces.

The second problem to be solved is illustrated in Figure C.5. The problem is to determine the apparent intersection, Q, of a surface edge, from \( V_1 \) to \( V_2 \), with a line, from \( P_1 \) to \( P_2 \) as seen from the eye point E. A further condition is imposed because we seek only those points above the line joining \( V_1 \) and \( V_2 \) in Figure C.5.

First note that a point on the line segment can be represented parametrically as:

\[
P_1 + t (P_2 - P_1)
\] (C.3)

where the parameter \( t \) must satisfy \( 0 \leq t \leq 1 \) to be on the line segment. A point on the plane determined by E, \( V_1 \), and \( V_2 \) can be represented as:

\[
E + \alpha (V_1 - E) + \beta (V_2 - E)
\] (C.4)

To restrict the points to those between the lines joining \( V_1 \) and \( V_2 \) to E, we must impose the conditions that \( 0 \leq \alpha \) and \( 0 \leq \beta \). In addition, to get points above the line joining \( V_1 \) and \( V_2 \) we must have \( 1 \leq \alpha + \beta \). Therefore, the final determination of Q, if it exists, can be done by equating the expressions in equations C.3 and C.4 to get:

\[
\alpha (V_1 - E) + \beta (V_2 - E) + t (P_1 - P_2) = (P_1 - E)
\] (C.5)

Equation C.5 gives three equations in three unknowns. If the solutions satisfy:

\[
\begin{align*}
0 & \leq t \leq 1 \\
0 & \leq \alpha, \ 0 \leq \beta \\
1 & \leq \alpha + \beta
\end{align*}
\] (C.6)

Then a valid point Q exists and it can be found using the computed value of t in equation C.3.
FIGURE C.1: An Invisible Edge

FIGURE C.2: A Possibly Visible Edge
FIGURE C.3: The Basic Hidden Line Algorithm
FIGURE C.4: Is a Point Within an Apparent Surface Boundary

FIGURE C.5: Finding the Apparent Intersection of a Line and Surface Boundary
APPENDIX D

A PRE-PROCESSED PROGRAM

In Chapter IV, a listing of a complete program was shown. On the following pages this same program is shown after it has passed through the PL/1 pre-processor. It should be noted that the output of the pre-processor has been extensively re-formatted. The actual output of the pre-processor is almost unreadable.
EXAMPLE: PROCEDURE OPTIONS(MAIN);
/*
* THIS PROGRAM PRODUCES AN ANIMATED SEQUENCE SHOWING A STELLATED
DODECAHEDRON ROTATING IN SPACE. HIDDEN LINES IN THE OBJECT ARE
REMOVED. A STELLATED DODECAHEDRON IS THE SOLID FIGURE WHICH IS
GENERATED BY EXTENDING THE FACES OF A REGULAR DODECAHEDRON. */

DECLARE FL ENTRY(CHARACTER(6),FLOAT BINARY);
DECLARE FX ENTRY(CHARACTER(6),FIXED BINARY);
DECLARE PX(32) FLOAT BINARY STATIC INITIAL(
  0.2764,-0.1056,-0.3416,-0.1056, 0.2764, 0.5528, 0.1708,-0.4472,
  -0.4472, 0.1708, 0.4472,-0.1708,-0.5528,-0.1708, 0.4472, 0.3416,
  0.1056,-0.2764, 0.2764, 0.1056, 0.0000, 0.8944, 0.2764,-0.7236,
  -0.7236, 0.2764, 0.7236,-0.2764,-0.8944,-0.2764, 0.7236, 0.0000);
DECLARE PY(32) FLOAT BINARY STATIC INITIAL(
  0.2008, 0.3249, 0.0000,-0.3249,-0.2008, 0.0000, 0.5257, 0.3249,
  -0.3249,-0.5257,-0.3249, 0.5257, 0.0000,-0.5257,-0.3249, 0.0000,
  0.3249,-0.2008,-0.3249, 0.2008, 0.0000, 0.8507, 0.5257, 0.3249,
  -0.5257,-0.8507,-0.5257, 0.8507, 0.0000,-0.8507,-0.5257, 0.0000);
DECLARE PZ(32) FLOAT BINARY STATIC INITIAL(
  0.4472,-0.4472, 0.4472,-0.4472,-0.4472, 0.1056, 0.1056,-0.1056,
  0.1056,-0.4472, 0.0000,-0.4472,-0.4472, 0.0000, 0.4472,-0.4472,
  -0.4472, 0.4472, 0.4472, 0.4472, 0.4472, 0.4472, 0.4472, 0.0000);
DECLARE LI(120) FIXED BINARY STATIC INITIAL(
  5, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 11, 7, 12, 8, 13, 9, 14, 10, 15,
  6, 7, 8, 9, 10, 16, 17, 18, 19, 20, 21, 21, 22, 22, 22, 22,
  23, 23, 23, 23, 24, 24, 24, 24, 25, 25, 25, 25, 25, 26, 26, 26, 26,
  27, 27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 30, 30, 30,
  31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 21, 21, 21, 21, 22, 22,
  23, 23, 23, 24, 24, 25, 25, 26, 26, 31, 31, 31, 28, 29, 30, 27, 28,
  29, 30, 31, 31);
DECLARE L2(120) FIXED BINARY STATIC INITIAL(
  1, 2, 3, 4, 5, 11, 12, 13, 14, 15, 11, 7, 12, 8, 13, 9, 14, 10, 15,
  6, 16, 17, 18, 19, 20, 17, 18, 19, 20, 16, 1, 2, 3, 4, 5, 5, 11, 6, 15,
  1, 2, 12, 7, 11, 2, 3, 13, 8, 12, 3, 4, 14, 9, 13, 4, 5, 15, 10, 14,
  16, 17, 7, 11, 6, 17, 18, 8, 12, 7, 18, 19, 9, 13, 8, 19, 20, 10, 14, 9,
  20, 16, 6, 15, 10, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 23, 24, 25, 26, 22,
  27, 23, 28, 24, 29, 25, 30, 26, 31, 32, 22, 27, 28, 29, 30, 31, 32, 32, 32,
  32, 32, 32);
DECLARE S1(60) FIXED BINARY STATIC INITIAL(
  1, 2, 3, 4, 5, 1, 6, 11, 20, 10, 2, 7, 13, 12, 6, 3, 8, 15, 14, 7,
  4, 9, 17, 16, 8, 5, 10, 19, 18, 9, 11, 12, 22, 26, 21, 13, 14, 29, 27, 22,
  15, 16, 24, 28, 23, 17, 18, 25, 29, 24, 19, 20, 21, 30, 25, 26, 27, 26, 29, 30);
DECLARE S2(60) FIXED BINARY STATIC INITIAL(
  35, 31, 32, 33, 34, 37, 38, 39, 40, 36, 42, 43, 44, 45, 41, 47, 48, 49, 50, 46,
  52, 53, 54, 55, 51, 57, 58, 59, 60, 56, 64, 63, 62, 61, 65, 69, 68, 67, 66, 70,
  74, 73, 72, 71, 75, 79, 78, 77, 76, 80, 84, 83, 82, 81, 85, 87, 88, 89, 90, 86);
DECLARE S3(60) FIXED BINARY STATIC INITIAL(
  31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
  51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 65, 64, 63, 62, 61, 70, 69, 68, 67, 66,
  75, 74, 73, 72, 71, 80, 79, 78, 77, 76, 85, 84, 83, 82, 81, 86, 87, 88, 89, 90);
DECLARE FG(120) CHARACTER(1) STATIC INITIAL((30)(1)'A', (60)(1)'B', (30)(1)'C'))
DECLARE DOBC CHARACTER(6) INITIAL(' 9.00');
DECLARE DSCR CHARACTER(6) INITIAL(' 10.00');
DECLARE EYES CHARACTER(6) INITIAL(' 0.70');
DECLARE SCRZ CHARACTER(6) INITIAL(' 2.50');
DECLARE FPIC CHARACTER(6) INITIAL(' 24');
DECLARE MPIC CHARACTER(6) INITIAL(' 240');
DECLARE NFPP CHARACTER(6) INITIAL(' 1');
DECLARE PSTP CHARACTER(6) INITIAL(' 1.00');
DECLARE (XDDBC, XDSCR, XEYES, XSCRZ, XRSTP) FLOAT BINARY;
DECLARE (XFPIC, XMPLIC, XNFPP) FIXED BINARY;
DECLARE (V1, V2, V3) (3) FLOAT BINARY;
DECLARE (ARG, S, C) FLOAT BINARY;
DECLARE (I, J) FIXED BINARY;

DECLARE 1 MLCBLK CONTROLLED EXTERNAL,
  2 MLELEMM FIXED BINARY, /* DISPLAY ELEMENT SIZE */
  2 MLAREAN FIXED BINARY, /* GEOMETRIC AREA SIZE */
  2 MLERRFG FIXED BINARY, /* ERROR FLAG */
  2 MLDSP1 FIXED BINARY, /* DSP ELEMENT ID COUNT */
  2 MLGPNTR POINTER, /* POINTER TO GEOMETRY LIST */
  2 MLIDMSN FIXED BINARY, /* 0=2-D, 1=3-D */
  2 MLINTMM FIXED BINARY, /* 0=INTERAC., 1=MOVIE M. */
  2 MLSCON FIXED BINARY, /* 0=SINGLE ST., 1=CONTIN. */
  2 MLPULSE FIXED BINARY, /* 0=SINGLE PL., 1=DOUBLE PL. */
  2 MLFRAME FIXED BINARY, /* FRAME COUNT FOR MM MODE */
  2 MLDelay FIXED BINARY, /* DELAY FOR INTERAC. MODE */
  2 MLLEADER FIXED BINARY, /* FRAME COUNT FOR LEADER */
  2 MLKATTN CHARACTER(12), /* KRUN MODE ATTENTIONS */
  2 MLSATTN CHARACTER(12), /* STOP MODE ATTENTIONS */
  2 MLFLAGS CHARACTER(4), /* SELECTION FLAGS */
  2 MLFXRMN FIXED BINARY, /* 0=FIXED DSP., 1=RUNNING */
  2 MLFIRST FIXED BINARY, /* FIRST FRAME FLAG */
  2 MLITES BIT(32); /* FUNCTION BUTTON STATUS */
DECLARE MLELEM CHARACTER( * ) VARYING CONTROLLED EXTERNAL;
DECLARE MLLE2 CHARACTER( * ) VARYING CONTROLLED EXTERNAL;
DECLARE MLAREA AREA( * ) CONTROLLED EXTERNAL;
DECLARE 1 MLGVAR EXTERNAL, 2 (PT(32), LN(120), SU(60), OB, VW ) POINTER;

ML0001: CALL MLINIT( 25, 1, 0, 750, 11000, ML0002 );
IF ML0002 = 0 THEN
  RETURN;
ON CONDITION(IDERROR) SNAP
  BEGIN;
    MLCBLK. MLERRFG=1;
  GO TO ML0001;
END;
ON CONDITION(MLERROR) GO TO ML0001;
ON AREA
  BEGIN;
    MLCBLK. MLERRFG=4;
  GO TO ML0001;
END;
PT=NULL;
LN=NULL;
SU=NULL;
DR=NULL;
VW=NULL;

/* TWO INTERACTIVE DISPLAYS ARE CREATED TO ALLOW THE CONSOLE
OPERATOR TO ENTER PARAMETERS THROUGH THE KEYBOARD. THE FIRST
DISPLAY GIVES HIM THE OPTION OF CHANGING PARAMETERS ASSOCIATED
WITH CAMERA POSITIONING WHILE THE SECOND DISPLAY LETS HIM
CHANGE PARAMETERS WHICH CONTROL THE NUMBER OF FRAMES WHICH WILL
BE PRODUCED. */

DSP1:
  DO;
  CALL IDREM('STOP');
  CALL IDRATN(0,MLATTND);
  IF MLATTND.MSTRING='NONE' THEN
      GO TO ML0003;
  MLBLK,MLSPI=0;
  CALL IDDATTN(MLBLK,MLKATTN1,MLBLK,MLSATN);
  CALL IDLITES('XOR',MLBLK,MLITES);
  CALL IDAPPLY(0,-1,IDMDOX,'DELT');
  END;
  DO;
  MLELEM='';
  CALL IDPOS('ABSL',190,700,MLELEM,ML0004);
  CALL IDTEXT('VLRG','ENTER CAMERA PARAMETERS!',MLELEM,
              ML0004);
  IF ML0004=0 THEN
      GO TO ML0005;
  MLBLK,MLSPI=MLBLK,MLSPI+1;
  CALL IDEPUT(MLBLK,MLSPI,'NORMBRT',MLELEM);
  END;
  DO;
  MLELEM='';
  CALL IDPOS('ABSL',260,600,MLELEM,ML0004);
  CALL IDTEXT('MEDM','DISTANCE TO CENTER OF OBJECT ',MLELEM,
              ML0004);
  CALL IDPOS('XAYR',260,-35,MLELEM,ML0004);
  CALL IDTEXT('MEDM','DISTANCE TO THE SCREEN ',MLELEM,
              ML0004);
  CALL IDPOS('XAYR',260,-35,MLELEM,ML0004);
  CALL IDTEXT('MEDM','EYE SEPARATION (STEREO ONLY)',MLELEM,
              ML0004);
  CALL IDPOS('XAYR',260,-35,MLELEM,ML0004);
  CALL IDTEXT('MEDM','SCREEN SIZE ',MLELEM,
              ML0004);
  IF ML0004=0 THEN
      GO TO ML0005;
  MLBLK,MLSPI=MLBLK,MLSPI+1;
  CALL IDEPUT(MLBLK,MLSPI,'NORMBRT',MLELEM);
  END;

   - D-4 -
DO;
MLELEM='';
CALL IDP05('ABSL',680,600-35,MLELEM,ML0004);
CALL IDT05('MEDM',DSRC,MLELEM,ML0004);
CALL IDEPUL(MLCBLK,MLDSPI+2,'TTYBBRIT',MLELEM);
MLELEM='';
CALL IDP05('ABSL',680,600-70,MLELEM,ML0004);
CALL IDT05('MEH',EYES,MLELEM,ML0004);
CALL IDEPUL(MLCBLK,MLDSPI+3,'TTYBBRIT',MLELEM);
MLELEM='';
CALL IDP05('ABSL',680,600-105,MLELEM,ML0004);
CALL IDT05('MEDM',SCRZ,MLELEM,ML0004);
CALL IDEPUL(MLCBLK,MLDSPI+4,'TTYBBRIT',MLELEM);
MLELEM='';
CALL IDP05('ABSL',680,600,MLELEM,ML0004);
CALL IDT05('MEDM',DOBC,MLELEM,ML0004);
CALL IDEPUL(MLCBLK,MLDSPI+1,'TTYBBRIT',MLELEM);
CALL IDEATTN('TTYP','STOP');
CALL IDRGEN('STR');
CALL IDATTN(-1,MLATTND);
CALL IDDATTN('TTYP');
CALL IDLITES('LOR',MLCBLK,MLITES);
CALL IDEATTN(MLCBLK,MLKATTN,'KRUN');
CALL IDEATTN(MLCBLK,MLSATTN,'STOP');
  IF MLATTND.MLSTRING='GOOF' THEN
    GO TO ML0006;
  IF MLATTND.MLSTRING='PFM' THEN
    GO TO ML0007;
CALL IDEGET(MLCBLK,MLDSPI+2,MLELEM,ML0004,ML0004);
CALL IDXTYB(MLLElem,DSRC,ML0004);
CALL IDEGET(MLCBLK,MLDSPI+3,MLELEM,ML0004,ML0004);
CALL IDXTYB(MLLElem,EYES,ML0004);
CALL IDEGET(MLCBLK,MLDSPI+4,MLELEM,ML0004,ML0004);
CALL IDXTYB(MLLElem,SCRZ,ML0004);
CALL IDEGET(MLCBLK,MLDSPI+1,MLELEM,ML0004,ML0004);
CALL IDXTYB(MLLElem,DOBC,ML0004);
END;
ON CONVERSION GO TO DSP1;
CALL FL(DOBC,DOBC);
CALL FL(DSRC,DSRC);
CALL FL(EYES,EYES);
CALL FL(SCRZ,SCRZ);
DSP?
DO;
CALL IDRGEN('STOP');
ML0008:
  IF MLATTND.MLSTRING='NONE' THEN
    GO TO ML00C8;
MLCBLK,MLDSPI=0;
CALL IDDATTN(MLCBLK,MLKATTN|MLCBLK,MLSATTN);
CALL IDLITES('XOR',MLCBLK,MLITES);
CALL IDAPPLY(0,-1,IDMODX,'DELT');
END;
DO;
MLELEM='';
CALL IDPOS('ABSL',190,700,MLELEM,ML0004);
CALL IDTEXT('VLRG','ENTER MOTION PARAMETERS',MLELEM,ML0004);
  IF ML0004=0 THEN
    GO TO ML0005;
  MLCBLK,MLDSPI=MLCBLK,MLDSPI+1;
CALL IDEPUT(MLCBLK,MLDSPI,'NORMBRIT',MLELEM);
END;
DO;
MLELEM='';
CALL IDPOS('ABSL',260,600,MLELEM,ML0004);
CALL IDTEXT('MEDM','START/END FIXED PICTURES',MLELEM,ML0004);
CALL IDPOS('XARY',260,-35,MLELEM,ML0004);
CALL IDTEXT('MEDM','NUMBER OF MOVING PICTURES',MLELEM,ML0004);
CALL IDPOS('XARY',260,-35,MLELEM,ML0004);
CALL IDTEXT('MEDM','NUMBER OF FRAMES/PICTURE',MLELEM,ML0004);
CALL IDPOS('XARY',260,-35,MLELEM,ML0004);
CALL IDTEXT('MEDM','ROTATION (DEGREES/PICTURE)',MLELEM,ML0004);
  IF ML0004=0 THEN
    GO TO ML0005;
  MLCBLK,MLDSPI=MLCBLK,MLDSPI+1;
CALL IDEPUT(MLCBLK,MLDSPI,'NORMBRIT',MLELEM);
END;
DO;
MLELEM='';
CALL IDPOS('ABSL',680,600-35,MLELEM,ML0004);
CALL IDTEXT('MEDM',MPIC,MLELEM,ML0004);
CALL IDEPUT(MLCBLK,MLDSPI+2,'TTYBBRIT',MLELEM);
MLELEM='';
CALL IDPOS('ABSL',680,600-70,MLELEM,ML0004);
CALL IDTEXT('MEDM',NFP,MLELEM,ML0004);
CALL IDEPUT(MLCBLK,MLDSPI+3,'TTYBBRIT',MLELEM);
MLELEM='';
CALL IDPOS('ABSL',680,600-105,MLELEM,ML0004);
CALL IDTEXT('MEDM',RSTP,MLELEM,ML0004);
CALL IDEPUT(MLCBLK,MLDSPI+4,'TTYBBRIT',MLELEM);
MLELEM='';
CALL IDPOS('ABSL',680,600,MLELEM,ML0004);
CALL IDTEXT('MEDM',FPIC,MLELEM,ML0004);
CALL IDEPUT(MLCBLK,MLDSPI+1,'TTYBBRIT',MLELEM);
CALL IDEATTN('TTYP','STOP');
CALL IDRGEN('STRT');
CALL IDEATTN(-1,MLATTND);
CALL IDDATTN('TTYP');
CALL IDLITES('LOR',MLCBLK,MLITES);
CALL IDEATTN(MLCBLK,MLKATTN,'KRUN');
CALL IDEATTN(MLCBLK,MLSATTN,'STOP');
IF MLATTND. MLSTRING='GOOF' THEN
  GO TO ML0006;
IF MLATTND. MLSTRING='PFLM' THEN
  GO TO ML0007;
CALL IDEGET(MLCLBLK. MLDSPI+2, MLELEM, ML0004, ML0004);
CALL IDXTTYB(MLELEM, MPIC, ML0004);
CALL IDEGET(MLCLBLK. MLDSPI+3, MLELEM, ML0004, ML0004);
CALL IDXTTYB(MLELEM, NFPP, ML0004);
CALL IDEGET(MLCLBLK. MLDSPI+4, MLELEM, ML0004, ML0004);
CALL IDXTTYB(MLELEM, RSTP, ML0004);
CALL IDEGET(MLCLBLK. MLDSPI+1, MLELEM, ML0004, ML0004);
END;
ON CONVERSION GO TO DSP2;
CALL FX(FPIC, XFPIC);
CALL FX(MPIC, XMPIC);
CALL FX(NFPP, XNFPP);
CALL FLR(RSTP, XRSTP);

/* THE GEOMETRY IS NOW DEFINED. THE DODECAHEDRON ITSELF HAS A 
SELECTION FLAG OF 'A'. THE STELLATION STRUCTURE HAS A SELECTION 
FLAG OF 'B'. FINALLY THE POINTS OF THE STELLATED DODECAHEDRON 
ARE JOINED TOGETHER WITH LINES WHICH FORM AN ICOSAHEDRON AND 
HAVE A SELECTION FLAG OF 'C'. */
DO I=1 TO HBOUND(PT, 1);
  V1(I)=PX(I);
  V1(I)=PY(I);
  V1(I)=PZ(I);
CALL MLPNTA(MLCLBLK. MLGPNT, PT(I), ', ', V1);
END;
DO I=1 TO HBOUND(LN, 1);
CALL MLLTN(MLCLBLK. MLGPNT, LN(I), FG(I), PT(L(l)(1)), 
        PT(L(2)(I)I));
  IF FG(I)='C' THEN
    CALL MLOBJA(MLCLBLK. MLGPNT, OB, O, LN(I), NULL, NULL, NULL, 
                 NULL, NULL, NULL, NULL, NULL, NULL, NULL);
END;
DO I=1 TO HBOUND(SU, 1);
CALL MLSURA(MLCLBLK. MLGPNT, SU(I), 0, LN(S1(I)), LN(S2(I)), 
        LN(S3(I)), NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);
CALL MLOBJA(MLCLBLK. MLGPNT, OB, O, SU(I), NULL, NULL, NULL, NULL, 
           NULL, NULL, NULL, NULL, NULL, NULL);  
END;

/* THE ANIMATED SEQUENCE IS NOW PRODUCED. THE PARAMETERS WHICH 
WERE ENTERED THROUGH THE KEYBOARD ARE USED AT THIS TIME. */
DO;
  CALL IDREGEN('STOP');
ML0009:
  CALL IDRATTN(0, MLATTND);
  IF MLATTND. MLSTRING='NONE' THEN
    GO TO ML0009;
  IF MLCLBLK. MLINTMM==O THEN

-D-7-
DO;
CALL MLEADR(MLCBLK,MLDIMS,MLCBLK,MLEADER,
MLCBLK,MLPULSE,MLCBLK,MLFLAGS,ML0004);
IF ML0004=0 THEN
GO TO ML0007;
END;
MLCBLK,MLFIRST=0;
CALL IDAPPLY(0,-1,IDMODX,'DELT');
END;
DO I=1 TO XMPIC;
APG=(I-1)*XRSTP;
S=SIND(ARG);
C=COSD(ARG);
V2(1)=-C*C;
V2(2)=S*C*(1+S);
V2(3)=S*(C*C-S);
V3(1)=S*C;
V3(2)=C*C-S*S*S;
V3(3)=-S*C*(1+S);
V1=-XDBBC*V2;
CALL MLVEWA(MLCBLK,MLGPNT, VW,1,V1, V2, V3, XEYES, XDSRCR, XSCRZ);
IF MLCBLK,MLFIRST=0 THEN
DO;
MLCBLK,MLFIRST=1;
CALL MLFXUP(MLCBLK,MLGPNT);
END;
MLELEM='';
CALL MLPROC(2,MLCBLK,MLDIMS,OB,VW,MLELEM);
ML0004=0;
IF MLCBLK,MLINTMM+MLCBLK,MLPULSE=2 THEN;
CALL IDSHUTR('OPEN');
IF MLCBLK,MLDIMS=1 THEN
DO;
MLELE2='';
CALL MLPROC(2,-1,OB,VW,MLELE2);
CALL IDEPUT(1001,'NORMBRITINCLSTOP',MLELEM);
CALL IDEPUT(-1001,'NORMBRITIMITSTRT',MLELE2);
IF MLCBLK,MLINTMM=1 THEN
DO;
CALL IDRATTN(-1,MLATTND);
IF MLCBLK,MLPULSE=1 THEN
CALL IDSHUTR('CLOSG');
IF MLATTND,MLSTRING='GOOF' THEN
GO TO ML0006;
IF MLATTND,MLSTRING='PFLM' THEN
DO;
IF MLATTND,MLARRAY(2)=32 THEN
GO TO ML0007;
ELSE
ML0004=1;
END;
IF MLCBLK.MLPULSE=1 THEN
    CALL IDSHUTR('OPEN');
    CALL IDRGEN('SWST');
END;
ELSE
    CALL IDEPUT(1001,'NORMBRITINCSTRT',MLELEM);
    IF MLCBLK.MLSSCON+MLCBLK.MLFXRNM-2*MLCBLK.MLINTMM>0 THEN
        MLC012=MLCBLK.MLDELAY;
    ELSE
        MLC012=-1;
    END;
    CALL IDRATTN(MLC012,MLATTND);
    IF MLCBLK.MLINTMM+MLCBLK.MLPULSE=2 THEN
        CALL IDSHUTR('CLOS');
    IFMLATTND.MLSTRING='GOOF' THEN
        GO TO MLC006;
    IFMLATTND.MLSTRING='PFL' THEN
        DO;
            IF MLATTND.MLARRAY(2)=25 THEN
                GO TO MLC011;
            IF MLATTND.MLARRAY(2)=26 THEN
                DO;
                    MLCBLK.MLFXRNM=1;
                    GO TO MLC011;
                END;
            IF MLATTND.MLARRAY(2)=27 THEN
                DO;
                    MLCBLK.MLFXRNM=0;
                    CALL IDRGEN('STRT');
                    GO TO MLC010;
                END;
                MLCO4=1;
            END;
        END;
        IF MLCO4=1 THEN
            GO TO MLC007;
    END;
    IF (I=1)||(I=XPIC) THEN
        J=XPIC*XNFPP;
    ELSE
        J=0;
    END;
    DO MLC016=1 TO J+XNFPP-1;
    IFMLCBLK.MLINTMM=1 THEN
        DO;
            IF MLCBLK.MLPULSE=1 THEN
                CALL IDSHUTR('OPEN');
            IF MLCBLK.MLDIMSN=1 THEN
                DO;
                    CALL IDRGEN('SWST');
                    CALL IDRATTN(-1,MLATTND);
                END;
        END;
    END;
END;
IF MLCBLK.MLPULSE=1 THEN
CALL IDSHUTR('CLOS');
IF MLATTND.MLISTRING='GOOF' THEN
GO TO ML0006;
IF MLATTND.MLISTRING='PFLM' THEN DO;
IF MLATTND.MARRAY(2)=32 THEN
GO TO ML0007;
ELSE
ML0004=1;
END;
IF MLCBLK.MLPULSE=1 THEN
CALL IDSHUTR('OPEN');
ML0015='SWST';
END;
END;
ML0013:
CALL IDRGEN(ML0015);
IF MLCBLK.MLSSCON+MLCBLK.MLFXRNM-2*MLCBLK.MLINTMM>0 THEN
ML0012=MLCBLK.MLDELAY;
ELSE
ML0012=-1;
CALL IDRATTN(ML0012,MLATTND);
IF MLCBLK.MLINTMM+MLCBLK.MLPULSE=2 THEN
CALL IDSHUTR('CLOS');
IF MLATTND.MLISTRING='GOOF' THEN
GO TO ML0006;
IF MLATTND.MLISTRING='PFLM' THEN DO;
IF MLATTND.MARRAY(2)=25 THEN
GO TO ML0014;
IF MLATTND.MARRAY(2)=26 THEN DO;
MLCBLK.MLFXRNM=1;
GO TO ML0014;
END;
IF MLATTND.MARRAY(2)=27 THEN DO;
MLCBLK.MLFXRNM=0;
GO TO ML0013;
END;
ML0004=1;
END;
IF ML0004=1 THEN
GO TO ML0007;
ML0014:
END;
END;
DO;
CALL IDRGEN('STOP');
ML0017:
CALL IDRATTN(0,MLATTND);
IF MLATTND.MLISTRING='NONE' THEN
GO TO ML0017;
IF MLCBLK.MLINTMM=0 THEN DO; CALL MLEADR(MLCBLK,MLDMSN,MLCBLK,MLEADER, MLCBLK,MLPULSE,MLCBLK,MLFLAGS,ML0004); IF ML0004=0 THEN GO TO ML0007; END; MLCBLK,MLFIRST=0; CALL IDAPPLY(0,-1,IDMODX,'DELT'); END;

/* THIS INTERNAL PROCEDURE IS USED TO CONVERT CHARACTER STRINGS TO FIXED OR FLOAT BINARY NUMBERS. THE REASON FOR THE UNUSUAL CODE IS TO FORCE ALL POSSIBLE ERRORS TO RAISE THE CONVERSION CONDITION. */
FL: PROCEDURE(XCH,XFL);
DECLARE XCH CHARACTER(6);
DECLARE XFL FLOAT BINARY;
DECLARE XFX FIXED BINARY;
DECLARE XTM CHARACTER(8) INITIAL('XXXXXXXX X');
DECLARE FLAG FIXED BINARY;
    FLAG=0;
    GO TO XXX1;
XXX1: SUBSTR(XTM,1,6)=XCH;
    IF FLAG=0 THEN
        GET STRING(XTM) LIST(XFL);
    ELSE
        GET STRING(XTM) LIST(XFX);
    RETURN;
END FL;

GO TO ML0001;
ML0005: MLCBLK,MLERRFG=3;
GO TO ML0007;
ML0006: MLCBLK,MLERRFG=2;
GO TO ML0007;
ML0007: SIGNAL CONDITION(MLERROR);
DECLARE IDPOS ENTRY(CHARACTER(*),FIXED BINARY,FIXED BINARY, CHARACTER(*) VARYING,FIXED BINARY);
DECLARE IDTEXT ENTRY(CHARACTER(*),CHARACTER(*),CHARACTER(*) VARYING, FIXED BINARY);
DECLARE IDEPUT ENTRY(FIXED BINARY,CHARACTER(*),CHARACTER(*) VARYING);
DECLARE IDEGET ENTRY(FIXED BINARY,CHARACTER(*) VARYING,FIXED BINARY, FIXED BINARY);
DECLARE IDMODX ENTRY(FIXED BINARY,CHARACTER(*));
DECLARE IDAPPLY ENTRY(FIXED BINARY,FIXED BINARY,ENTRY,1);
DECLARE IDXTYPE ENTRY(CHARACTER(*), VARYING, CHARACTER(*), FIXED BINARY);
DECLARE IDEATTN ENTRY(CHARACTER(*), CHARACTER(4));
DECLARE IDDATTN ENTRY(CHARACTER(*));
DECLARE IDRATTN ENTRY(FIXED BINARY(31,0), 1, 2 CHARACTER(4), 2 (5) FIXED BINARY);
DECLARE 1 MLATTND,
   2 MLSTRING CHARACTER(4),
   2 MLARRAY(5) FIXED BINARY;
DECLARE IDRGEN ENTRY(CHARACTER(4));
DECLARE IDLITES ENTRY(CHARACTER(4), BIT(*));
DECLARE IDSHUTR ENTRY(CHARACTER(4));
DECLARE MLINIT ENTRY(FIXED BINARY, FIXED BINARY, FIXED BINARY, FIXED BINARY, FIXED BINARY, FIXED BINARY);
DECLARE MLO002 FIXED BINARY;
DECLARE MLREADR ENTRY(FIXED BINARY, FIXED BINARY, FIXED BINARY, FIXED BINARY, CHARACTER(*), FIXED BINARY);
DECLARE MLFXUP ENTRY(POINTER);
DECLARE MLPROC ENTRY(FIXED BINARY, FIXED BINARY, POINTER, POINTER, POINTER, CHARACTER(*) VARYING);
DECLARE MLPNTA ENTRY(POINTER, POINTER, CHARACTER(1), (3) FLOAT BINARY);
DECLARE MLINSA ENTRY(POINTER, POINTER, CHARACTER(1), POINTER, POINTER);
DECLARE MLSURA ENTRY(POINTER, POINTER, FIXED BINARY, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER);
DECLARE MLOBJA ENTRY(POINTER, POINTER, FIXED BINARY, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTER, POINTE