

A SOLID MODELING PROGRAM FOR DESIGNERS

by

Patrick Scott Zeigler

Thesis submitted to the Faculty of the
Virginia Polytechnic and State University
in partial fulfillment of the requirements for the degree of

MASTER OF ARCHITECTURE

APPROVED :



John W. Wade, Chairman


Dennis Jones
Robert P. Schubert

February, 1988

Blacksburg, Virginia

A SOLID MODELING PROGRAM FOR DESIGNERS

by

Patrick Scott Zeigler

**Committee Chairman : John W. Wade
Architecture**

(ABSTRACT)

The personal computer is rapidly finding its way into the architectural working environment, at this time however it is used mostly as a drafting tool. This thesis is an investigation into programing that would allow the computer to become a design tool.

The issues that are dealt with in this thesis include an easy to use user interface that will not inhibit the design process, and develop a system that will allow design changes and additions to a model in a three dimensional sketch like ability.

Three dimensional models of paper, clay, wood and other materials have been used to create designs, and aid the designer in making decisions. This type of medium is difficult to make changes, and because of the scale of such materials it becomes difficult to work on interior spaces, thus more attention is usually placed on the exterior design. With the use of the computer these limitations may be eliminated, and the designer may create a design from any perspective or view point.

ACKNOWLEDGEMENTS

I would like to offer a special thanks to Robert Schubert, Dennis Jones and John Wade for their support and advice throughout this study. I would also like to recognize John and Dennis's work on the computer that has inspired me to produce this thesis.

A warm appreciation is extended to my parents for their support and sacrifices that made this study possible.

Finally, I am most grateful for my most precious loved ones.

TABLE OF CONTENTS :

INTRODUCTION.....	1
MODELING DESIGN PROCESS.....	5
GENERAL OVERVIEW OF SYSTEM.....	10
Setup.....	12
Translation Function.....	14
Drawing Function.....	17
Main Menu Functions.....	19
Main Menu Diagram.....	20
Dimension Functions.....	21
Solids Menu.....	24
Modeling Menu.....	32
Rotation Functions.....	34
Faceting Function.....	37
Gouraud Shading.....	40
Constant Shading.....	42
Translucency.....	44
Scaling.....	49
Centering.....	51
Origin Set.....	51
Viewport.....	54
Color.....	58
Remove Object.....	59

Windows.....	62
Files.....	64
CONCLUSIONS.....	66
BIBLIOGRAPHY.....	69
APPENDIX.....	70
VITA.....	322

LIST OF DIAGRAMS :

Modeling Design Diagram.....	7
General Overview Diagram	10
General Program Structure	11
Setup Diagram	13
Translation Function.....	14
Translation Key Pad	16
Drawing Function Diagram	18
Main Menu Diagram.....	20
Dimension Functions.....	21
Dimension Functions Diagram.....	23
Solids Menu Diagram.....	26
Cursor Function Diagram.....	27
Command List Diagram.....	30
Composite Solids Functions.....	31
Modeling Menu Branches.....	33
Path of X Axis Rotation.....	36
Path For Facet Functions	39
Path For Gouraud Shading Functions	41
Path For Constant Shading Functions	43
Translucency Functions Path.....	48
Path For Scale	50
Command List Example For Modeling Origin	52
Path For Modeling Origin	53
View Port Function Diagram.....	56

Color Funtion Path.....	58
Removal of Object.....	60
Point Evaluation	61
Windows.....	63
File Functions	65

INTRODUCTION:

In the last few years the introduction of the Personal computer has brought Computer Aided Design (CAD) to the small office and home. As the trend continues in the next few years we will have mainframe power available in the Personal Computer. Already many hardware developers have created plug-in boards with a very fast drawing speed, and call up graphic functions in firmware. With the introduction of these boards greater speed, and more flexibility in programing will be achieved. At this time there are a great many three dimensional CAD programs available, many are well suited for engineers, and draftsmen but there are few that take into account the needs of the designer. The designer's needs are much different than that of the draftsman or engineer. The designer needs a flexible program to use in case of changes, and a visual program with the ability to observe and compare. Because designers tend to be visual, visualization is primary. Aid from the computer will make the vision of the designer a reality in a matter of minutes. If the designer is not satisfied with a design then changes can be made neatly and quickly. The computer gives instant feedback, which can speed up the entire design process.

Three dimensional models are an essential tool for creating designs; however, there are few methods available that create 3D models quickly. Now with the introduction of CAD, models are created, transformed, and viewed from any perspective. CAD has an advantage in that the model can be viewed from the inside out as well as the outside in. This will aid the designer in creating new concepts and better integration of ideas. At the moment there are several limitations that exist before CAD can be used as a designing tool. The first and most important being that most designers are unfamiliar with the use of computers. Second most programs on the market today are very difficult to learn and work with. Once the programs become more orientated to designers and become easier to use, then more designers will turn to CAD for their designing needs.

The purpose of this thesis is to create a solid modeling program that can be used to create architectural designs, jewelry, furniture and 3D art forms on the computer.

By developing the design program the following objectives should be accomplished:

Create a 3D design program for my personal use.

Develop a driver for the Matrox SM-640 graphics board.

Develop skills to understand how to program other graphics boards on the market.

Develop skills in the C programming language.

Gain a better understanding of the computer operating system (DOS).

Learn general computing skills.

Gain a better understanding of Computer Aided Design.

Be more aware of available computer hardware.

The scope of the thesis includes only 3D solid modeling at this time. Elements that create a model are handled in the way one would construct a block type model. The programs methods of designing a model are similar to that of a sketch book. A designer sketches are not so concerned with exact dimensions, but more with the relationships of parts. Changes may be made easily so the designer can visualize a wider range of possibilities. The designer has the ability to program in exact dimensions; however, one of the strengths of the program is the ability to allow the designer to also make visual judgments, as in the sketch book. Therefore this program should not be seen as a drafting tool, but a sketch tool.

MODELING DESIGN PROCESS :

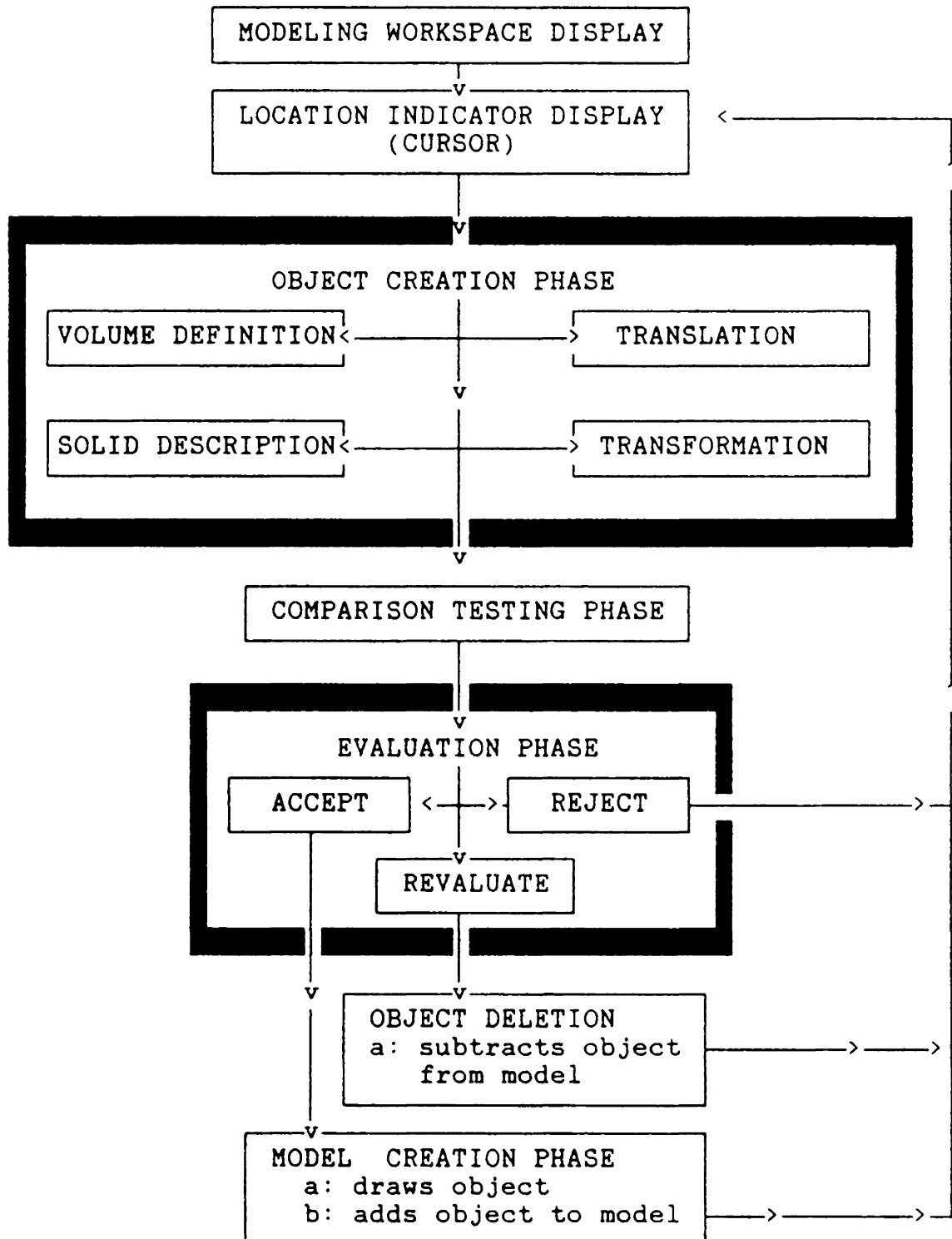
One of the main aspects of this program is its ability to allow the designer to test the relationships between elements before making a commitment to their existence in the work space. This is one of the computers strong points, but is not fully utilized in most existing CADD programs. Most other modeling mediums are not as flexible as well. During the design process these changes and comparisons are essential to allow the designer a full range of possibilities. If the drawing medium is not flexible, that inflexibility may influence the design process.

The program begins by presenting the designer with an empty work space, this work space may be increased or decreased in size at any time. Once the work space as been defined a location indicator called a cursor is shown at the origin 0.0, 0.0, 0.0 which is located at the center of the viewport. The designer is then automatically put into the object design phase, it is the main working environment. The object design phase includes:

Volume Definition	--Defines primitive, swept solid, extruded solids and composites
Translation	--Defines the location of an object
Solid Description	--Defines color, translucency, facets and shading
Transformation	--Defines Modeling transformations that effect the object
Reevaluate	--Deletes objects from the work space

These commands allow the designer to move, transform and define objects in three dimensional space before making a commitment to include that object in the work space. In this phase the object will appear only in wireframe. While in the Comparison Testing phase the object may be compared with existing objects in the model. From there the designer enter into the Evaluation Phase. At this point the designer may accept or reject the present configuration. If the configuration is unacceptable the designer may continue with the Object Creation Phase. If the model is satisfactory the Model Creation Phase is activated by pressing the "space bar" and the object with all of its attributes is drawn to the screen and added to the modeling file. The Reevaluate Phase allows the designer to go back and remove any object that has been added to the model.

MODELING DESIGN DIAGRAM :



GENERAL OVERVIEW OF SYSTEM :

The program begins by entering automatically into the main function. It then defines all global variables, and sets all graphic attributes, these attributes

include:

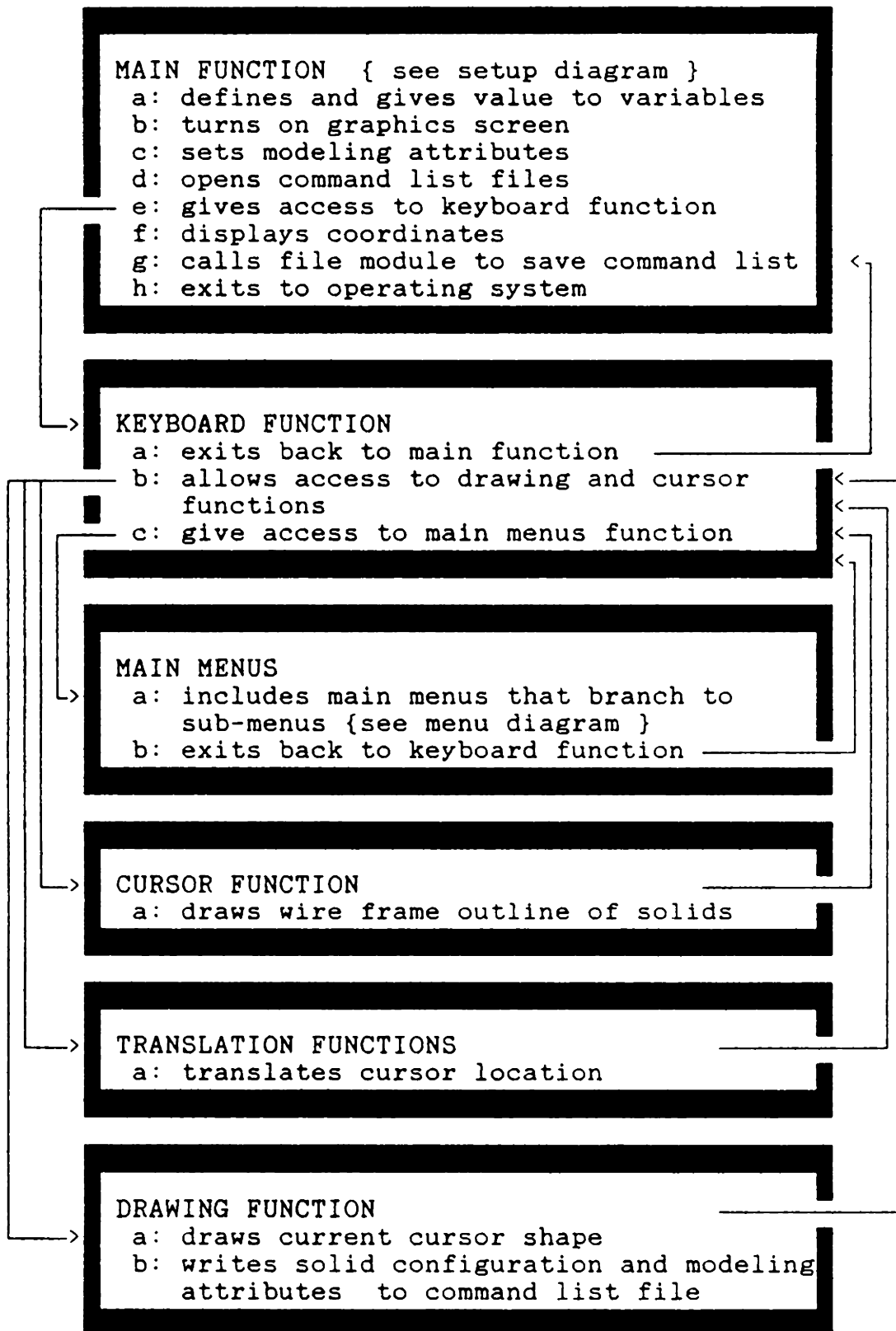
- window size
- viewport size
- viewing reference point
- clipping planes
- viewing distance
- lighting direction and intensity
- shading type
- color look up table

Once the attributes are set they are stored in a file called SETUP.D . SETUP.D is called up whenever there are any changes made to the setup attributes. The main function also creates COMMAND.D , this is the active work file, it will contain location of primitives, objects, modeling transformations, lighting attributes and color used in creating a model. SETUP.D and COMMAND.D together make up the Command List, which is used to recreate the model at another time.

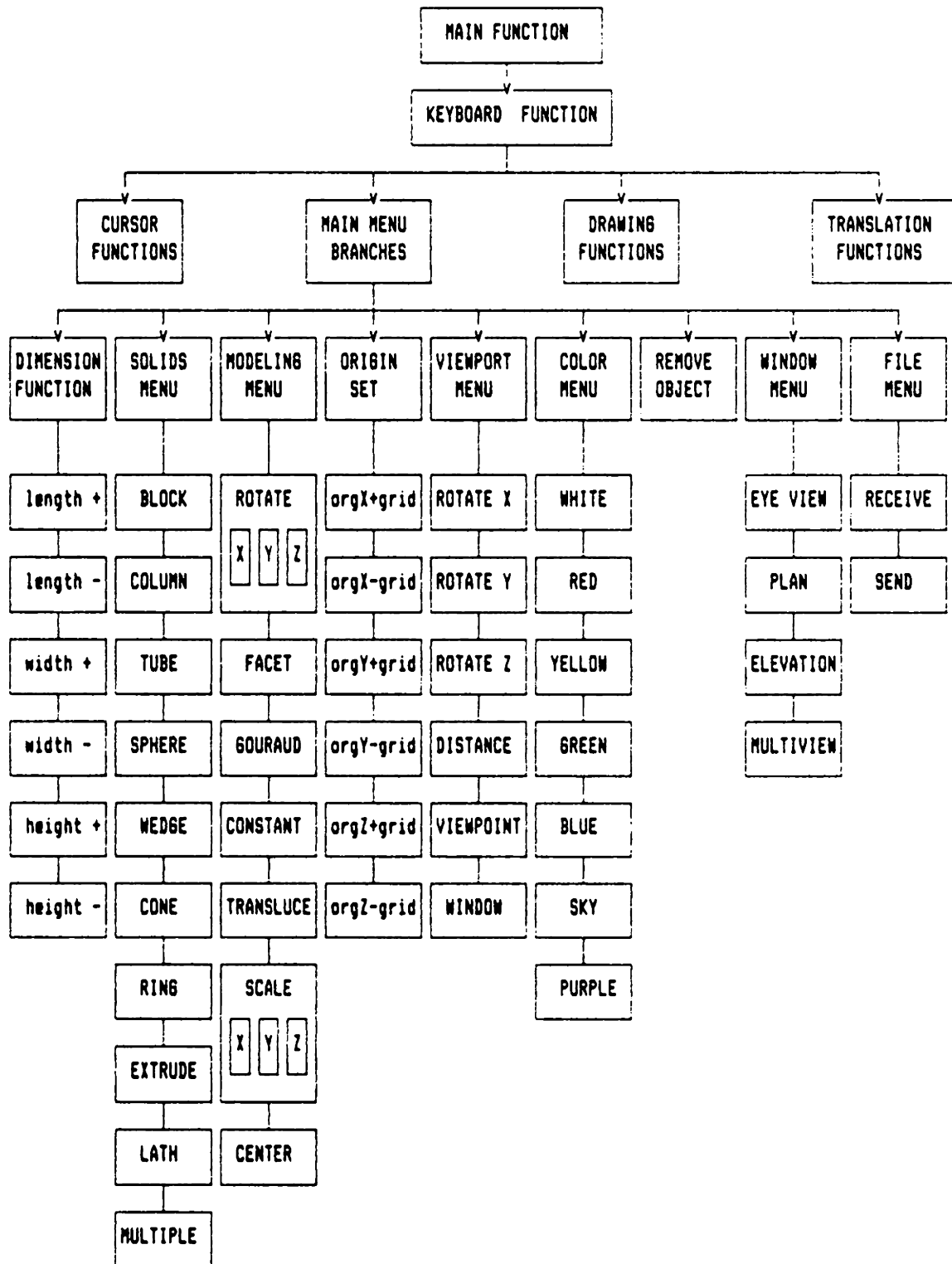
Once the main function creates the command list files it then opens up the keyboard function which gives access to cursor, drawing and main menu functions, the user will remain inside the main and keyboard function until the user exits the system.

The cursor function draws on the screen a 3D cross-hair that locates the current X, Y, and Z locations. If a primitive or composite object is chosen the cursor will take the shape of that object, and draw a fully shaded representation of that object, once the drawing command is activated. The drawing function also opens the command list file and sends drawing attributes to the file along with the chosen object. After these commands are sent to the file a permanent work file is created, and the file is closed. The program then enters back into the keyboard function, and waits for another input command.

GENERAL OVERVIEW DIAGRAM:



GENERAL PROGRAM STRUCTURE :

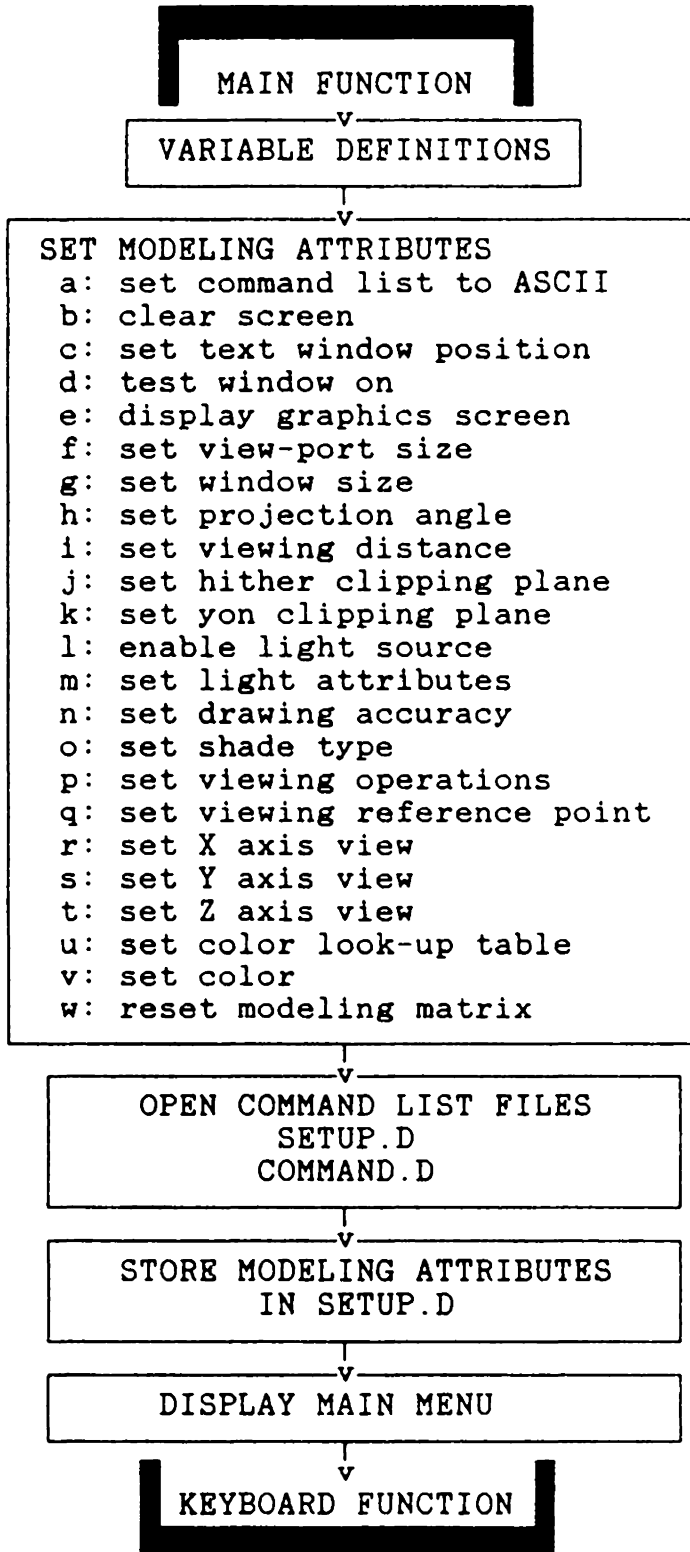


SETUP FUNCTIONS:

The first part of the Main Function consists of a function that defines all global variables used in the program. Once the variables have been defined, they are then assigned values, and set to the modeling attribute function, which sets the graphic screen to the assigned values. The program then creates two files SETUP.D and COMMAND.D . SETUP.D is responsible for all initial attributes and viewing commands. COMMAND.D is responsible to record all changes of the initial attributes and additions of objects with there locations. Together they are called the Command List. The Command List when passed through the program will recreate any drawing or model previously created on the screen.

After the Command List files have been set up the program enters into the menu setup function. This function displays the main menus accessible from the keyboard. At this point the program enters into the keyboard function which wait for the users input. The keyboard function is the most active function in the program, and can branch to most any other function in the program. The main branches include: menu branch, translation branch, and drawing branch.

SETUP DIAGRAM:

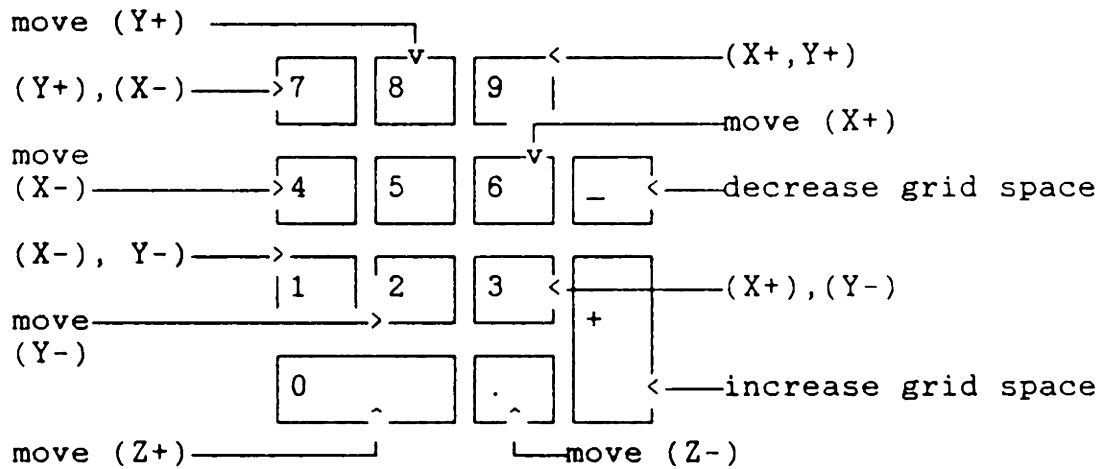


TRANSLATION FUNCTION:

One of the most important and most used functions is the Translation Function it allows the user to move an object freely on the screen, calculate distance from the last entry and stores X, Y and Z locations. The Translation function is actually a group of separate subfunctions that are accessed through the "num" key pad. Each key will activate a specific function that will move the object in a particular direction, they are as follows:

KEY	FUNCTION
.	moves Z axis negative
0	moves Z axis positive
1	moves X axis negative & Y axis negative
2	moves Y axis negative
3	moves X axis positive & Y axis negative
4	moves X axis negative
5	
6	moves X axis positive
7	moves X axis negative & Y axis positive
8	moves Y axis positive
9	moves Y axis positive & X axis positive
-	decrease grid space
+	increases grid space

TRANSLATION "ON" KEY PAD FUNCTIONS :

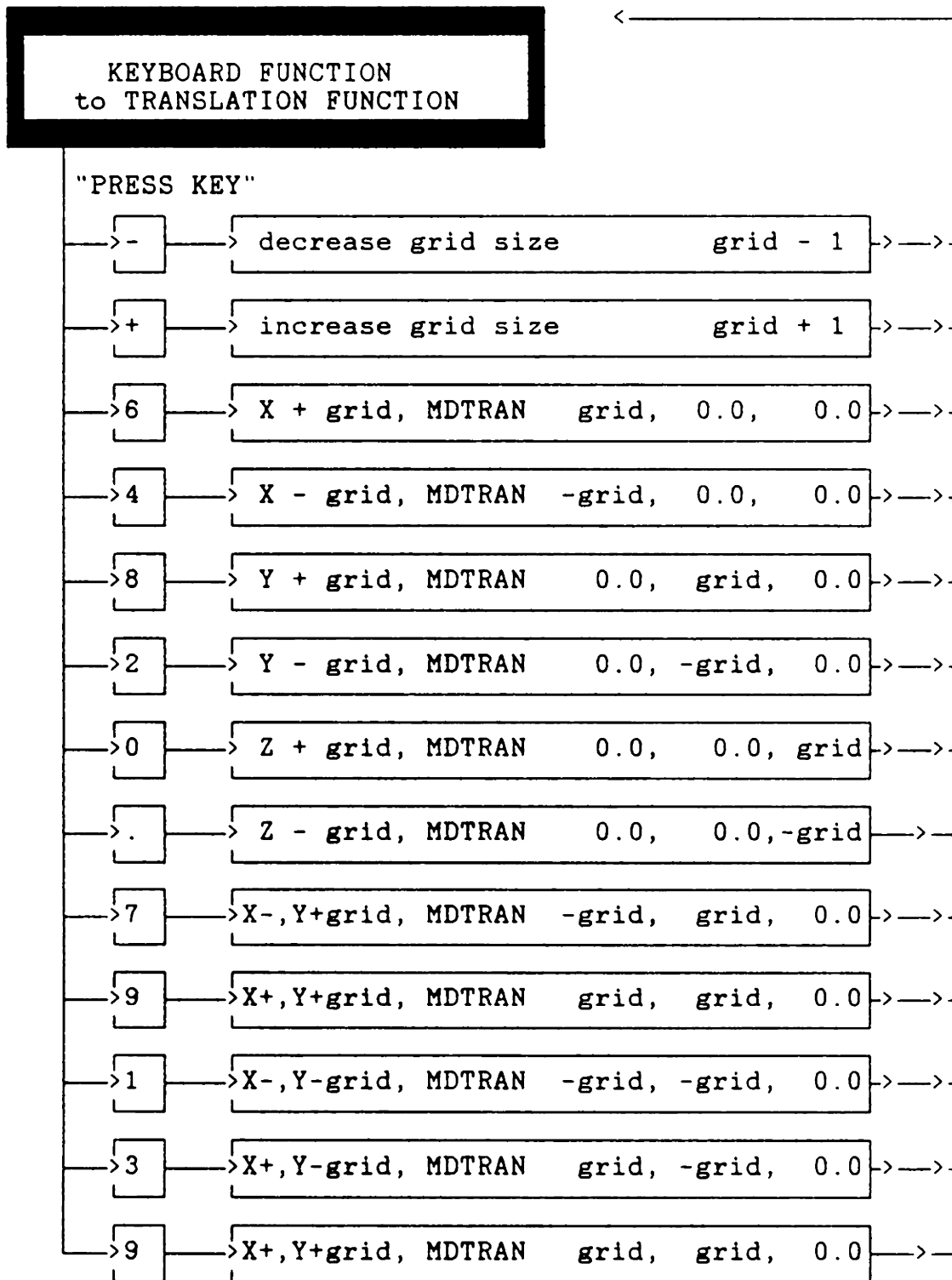


Each time a key is pressed it will automatically move the cursor or object in the chosen direction. At the bottom of the screen there is a constant read out of the present coordinate location, as well as the last modeling origin entered.

COORDINATE INFORMATION			ROTATION INFORMATION		
XD=0.00	YD=0.00	ZD=0.00	RX=0:0	RY=0:0	RZ=0:0
X=0.00	Y=0.00	Z=0.00	RX=0:0	RY=0:0	RZ=0:0

Once the proper coordinates have been located the "space" bar should be pressed, this will enter into the Drawing Function, and record the object and its location. The Command used for translation is MDTRAN rx, ry, rz. The rx, ry and rz represent relative X, Y and Z coordinates. For example : MDTRAN 50.0, 0.0, 0.0 will move the drawing point to 50.0, 0.0, 0.0. If it is entered twice the active point would be 100.0, 0.0, 0.0.

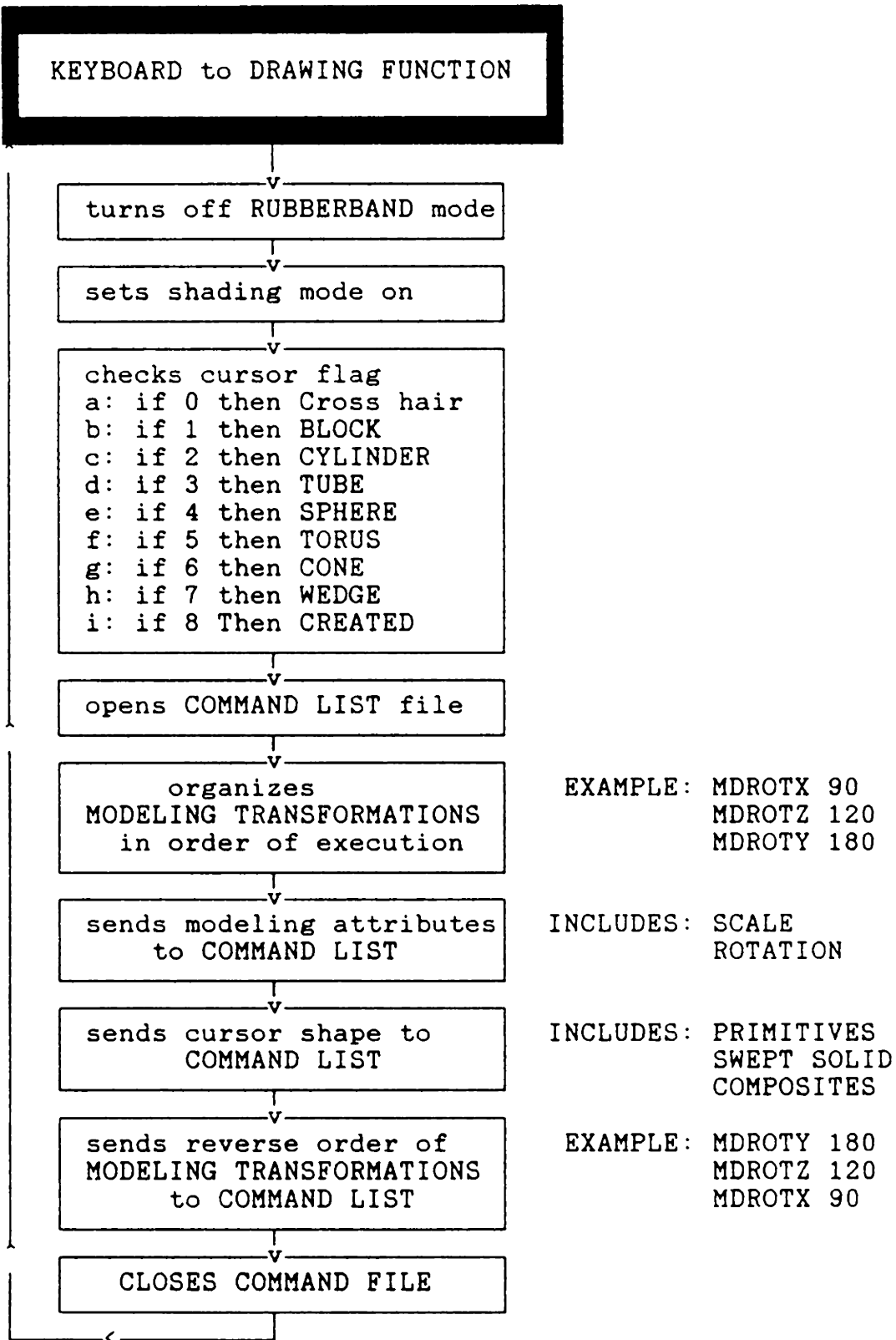
TRANSLATION FUNCTION DIAGRAM :



DRAWING FUNCTION :

The Drawing function is very closely tied to the Translation Function, Cursor Function and Main Menu functions. Whenever a change in status of the coordinates, cursor shape or modeling attributes are made, the drawing function records these changes in the Command List. After a Command List of objects and modeling attributes is created, the model can be recreated at another time. The Drawing Function and Main Menu functions act together to construct the Command list. The Drawing Function is activated by pressing the "Space" bar, once this is done the XOR drawing mode is turned off and the shading attributes are set. After this is accomplished the cursor Flag is checked to see which primitive or solid is to be drawn at the active coordinate location. Next the rotation attributes are arranged so that the proper order of execution will take place. Then COMMAND.D file is opened and the solid along with its modeling attributes is written to the file. If any rotation or scaling attributes are active the modeling matrix must be reset to the default values, This is because rotation and scaling effects the entire modeling matrix not just the object as it may appear when running the program.

DRAWING FUNCTION DIAGRAM :



MAIN MENU FUNCTIONS:

From the Keyboard Function eight main menus or functions are accessible, and they represent eight specific groups of functions that are used in the creation of a model. Each group is represented in an abbreviated form at the top of the screen. {see fig below}

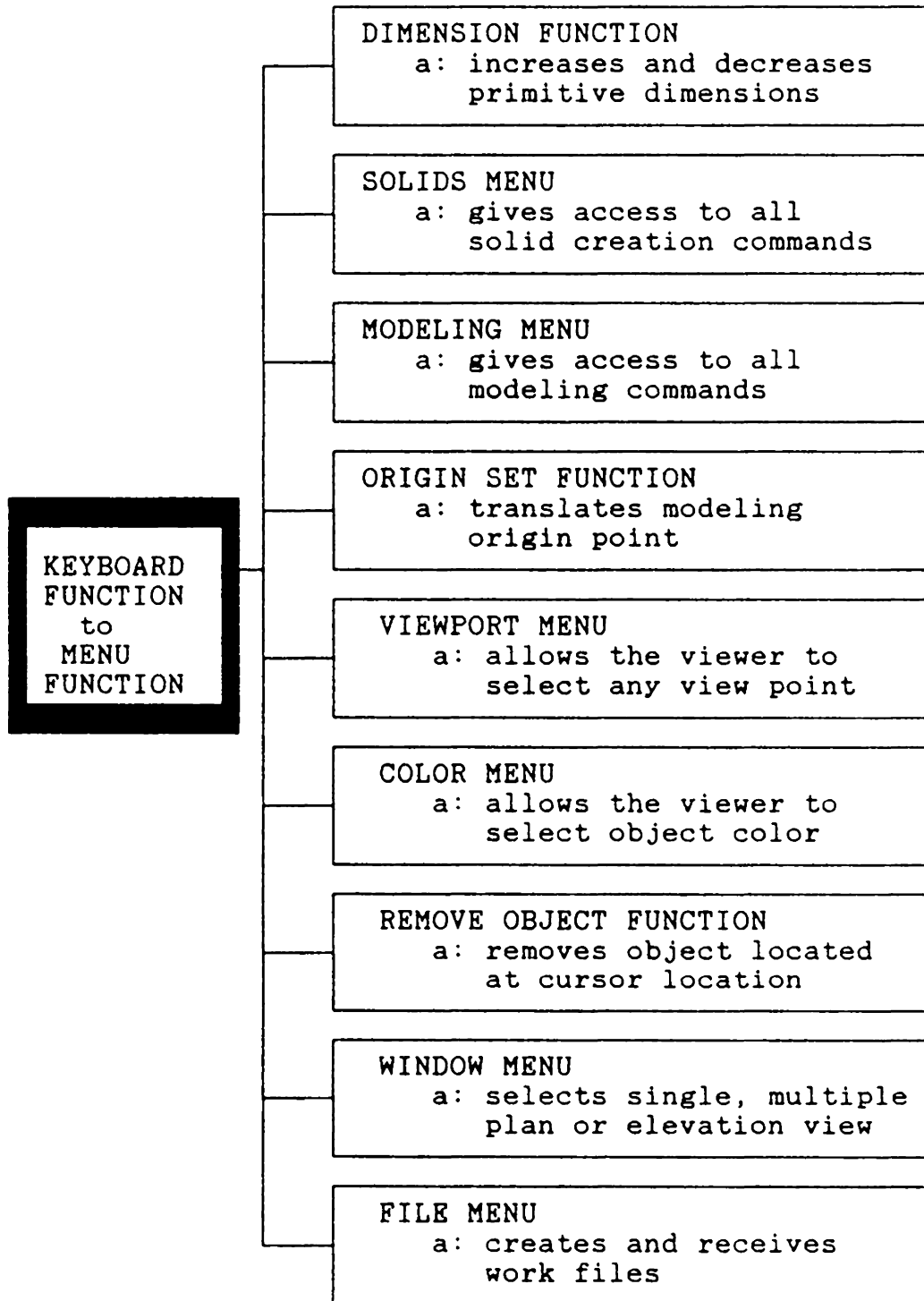
DIMN SOLD MODL ORGN VIEW COLR RMOV WNDW FILE ERAS

The menus and functions are arranged in an order of most frequent use. Each four letter abbreviation represents the following function or menu.

DIMN	--	DIMENSION FUNCTION
SOLD	--	SOLID MENU
MODL	--	MODELING MENU
ORGN	--	MODELING ORIGIN FUNCTION
VIEW	--	VIEW MENU
COLR	--	COLOR MENU
RMOV	--	REMOVE OBJECT FUNCTION
WNDW	--	SET PLAN, ELEVATION OR MULTIPLE WINDOW
FILE	--	SENDS OR RECEIVES WORKING FILES
ERAS	--	ERASES THE ENTIRE SCREEN

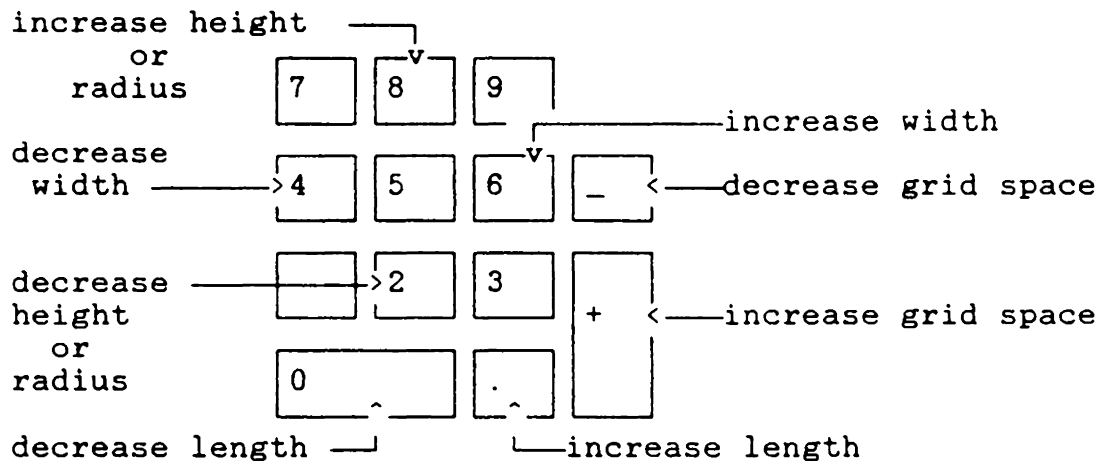
The main menus will then branch to more specific sub-menus. Many of the major functions such as translation and drawing are even accessible from within the sub menus. The sub-menus will always return back to the main menu by pressing "Esc".

MAIN MENU DIAGRAM :



DIMENSION FUNCTION:

The dimension function allows the user to increase or decrease the length, height or width of a block primitive and the radius of a sphere or cylindrical primitive. Once the function is entered by pressing "d" .The dimension function will be activated, then the num keys will control expansion or contraction along the width, length, height or radius. The amount of expansion is determined by the value of the grid spacing which can be adjusted by the plus and minus keys.

DIMENSION "ON" KEY PAD FUNCTIONS :

The dimension function works by adding or subtracting the grid value to the dimension value. Once the value are defined then they are used in the solid function to specify a height, length, width or radius, an example of this is as follows:

PSOLID 4 WIDTH HEIGHT LENGTH

PSOLID - command for solids

4 - sets solid mode to block

WIDTH - defines width value

HEIGHT - defines height value

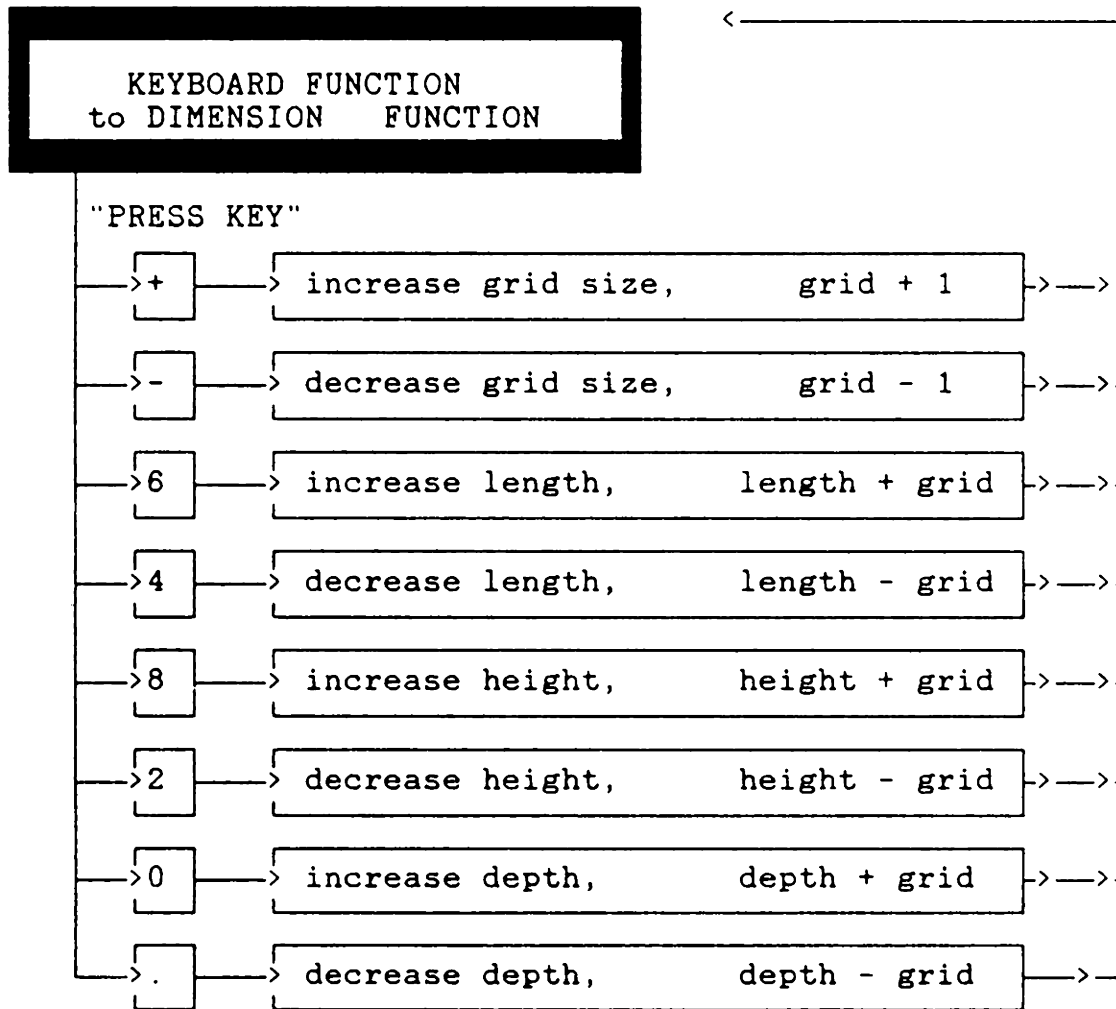
LENGTH - defines length value

In order to draw a block with a width of 100, height of 50 and length of 20. The command

PSOLID 4 100.0 50.0 20.0

must be created and sent to the Command List.

DIMENSION FUNCTION DIAGRAM :



SOLIDS MENU :

The Solids Menu gives the user access to a set of functions that will draw a set of primitives, swept solids or composite solids. The first function will display the selection of solids available to the user at the top of the screen like the figure below.

BLOC COLM TUBE SPHR WEDG PYRY RING EXTR LATH MULT

The solids are arranged in an order most frequently used, and a four letter abbreviation is used to symbolize each object, the abbreviations are represented as:

BLOC	--	BLOCK
COLM	--	CYLINDER, COLUMN
TUBE	--	TUBE
SPHR	--	SPHERE
WEDG	--	WEDGE
PYRY	--	CONE, PYRAMID
RING	--	TORUS, RING, WHEEL
EXTR	--	EXTRUDED SOLID
LATH	--	TURNED SOLID
MULT	--	COMPOSITE SOLIDS

In order to activate one of the solid commands press the first letter of the abbreviation on the keyboard. Once this is done a value is set to a variable that corresponds to the assigned solid, and the cursor will become that solid. The cursor will remain that solid until a change is made. If the "space" bar is pressed it will activate the drawing function. The selected solid will then be drawn to the screen and sent to the Command List along with other attributes that define the solid.

The command used to activate any primitive solid is PSOLID, then a mode zero to six is used to specify the type a solid. The following modes are used:

CYLINDER	= 0
CONE	= 1
SPHERE	= 2
TORUS	= 3
BLOCK	= 4
WEDGE	= 5
TUBE	= 6

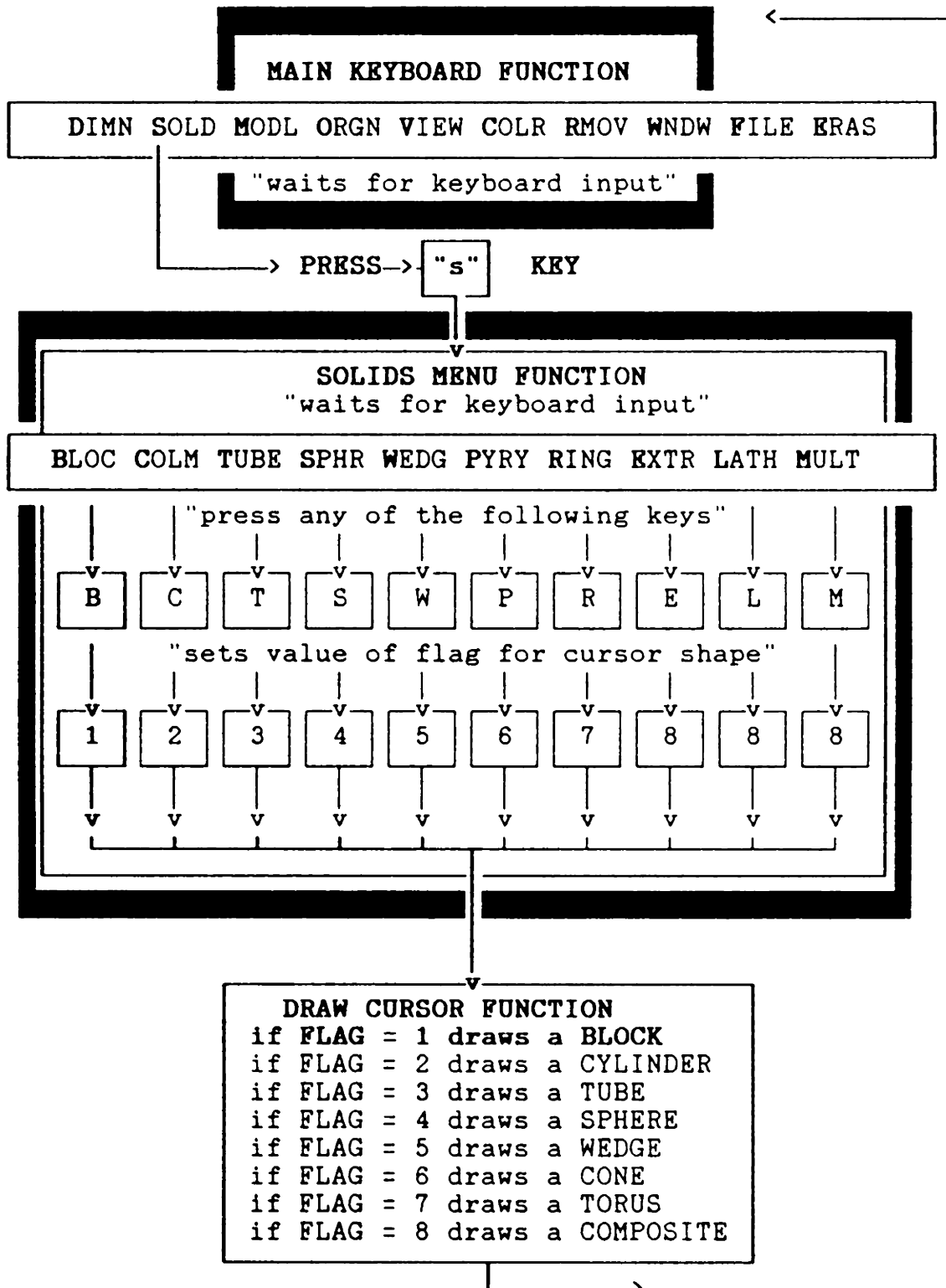
The mode number is written directly after the solid command, such as for a sphere. " PSOLID 2 " After the mode number comes the dimension information. Each primitive has its own format, they are as follows:

CYLINDER	- PSOLID 0, RADIUS, LENGTH
CONE	- PSOLID 1, RADIUS-TOP, RADIUS-BOTTOM, LENGTH
SPHERE	- PSOLID 2, RADIUS
TORUS	= PSOLID 3, RADIUS-INTERIOR, RADIUS-EXTERIOR
BLOCK	= PSOLID 4, WIDTH, HEIGHT, LENGTH
WEDGE	= PSOLID 5, WIDTH, HEIGHT, LENGTH
TUBE	= PSOLID 6, RADIUS-INTERIOR, RADIUS-EX, LENGTH

The mode number is activated through the solids menu. Once the selection is made a number is assigned to variable that dictates the cursor shape and the primitive that will be drawn.

CREATE SOLIDS FUNCTIONS:

EXAMPLE BLOCK :



CURSOR FUNCTION DIAGRAM :

KEYBOARD to CURSOR FUNCTION

a: gives cursor shape

b: flag is set in solids function

if cursor flag equals zero
then cursor remains a CROSS HAIR

if cursor flag equals one
then cursor becomes a BLOCK

if cursor flag equals two
then cursor becomes a CYLINDER

if cursor flag equals three
then cursor becomes a TUBE

if cursor flag equals four
then cursor becomes a SPHERE

if cursor flag equals five
then cursor becomes a TORUS

if cursor flag equals six
then cursor becomes a CONE

if cursor flag equals seven
then cursor becomes a WEDGE

if cursor flag equals eight
then cursor is CREATED by the user

v

KEYBOARD

Because swept, extruded and composite solids require more information another format is used, for swept and extruded solids the Command SSOLID is used, the extruded solid uses mode 1, and swept solid uses mode 2. Then following the mode number is a closed 2D vector that is either extruded along a sweep vector, or rotated around the Y axis in a counter-clockwise direction. In order to draw a prism shape, the command format is as follows:

SSOLID 0, NUMBER-OF-VERTICES, X Y POINT SET

EXAMPLE: for prism with points 0.0, 0.0, 0.0, 100.0, -100.0, 100.0, 0.0, 0.0

```
SSOLID 0, 4
      0.0, 0.0
      0.0, 100.0
    -100.0, 100.0
      0.0, 0.0
```

For Composite Solids the Command List Definition or Solid Definition commands may be used. They are as follows:

COMMAND LIST DEFINITIONS

```
CLBEG num - begins Command List num = {0...256}
CLEND      - ends current Command List
CLRUN num  - runs list num
```

SOLID DEFINITION COMMANDS

```
SOLBEG num - begins Solid List num = {0...256}
SOLEND      - ends current Solid List
SOLRUN num  - runs list num
```

The advantage to using one over the other is the the Command List is faster and may include any other command to define an object, while Solid Definition will allow subtraction and intersection of solids {not yet implemented}. An example of a composite solid may be a frame that has four block primitives intersected at ends. The following is an example:

```
CLBEG  1          --- begin Command List 1
MDTRAN  -8.0,  0.0, 0.0  --- move to -8,0,0
PSOLID  4,  2.0, 14.0, 4.0 --- draw block (1)
MDTRAN  8.0,  8.0, 0.0  --- move 8,8,0
PSOLID  4, 18.0,  2.0, 4.0 --- draw block (2)
MDTRAN  8.0, -8.0, 0.0  --- move 8,-8,0
PSOLID  4,  2.0, 14.0, 4.0 --- draw block (3)
MDTRAN -8.0, -8.0, 0.0  --- move -8,-8,0
PSOLID  4, 18.0,  2.0, 4.0 --- draw block (4)
CLEND          --- end List
CLRUN  1          --- run List 1
```

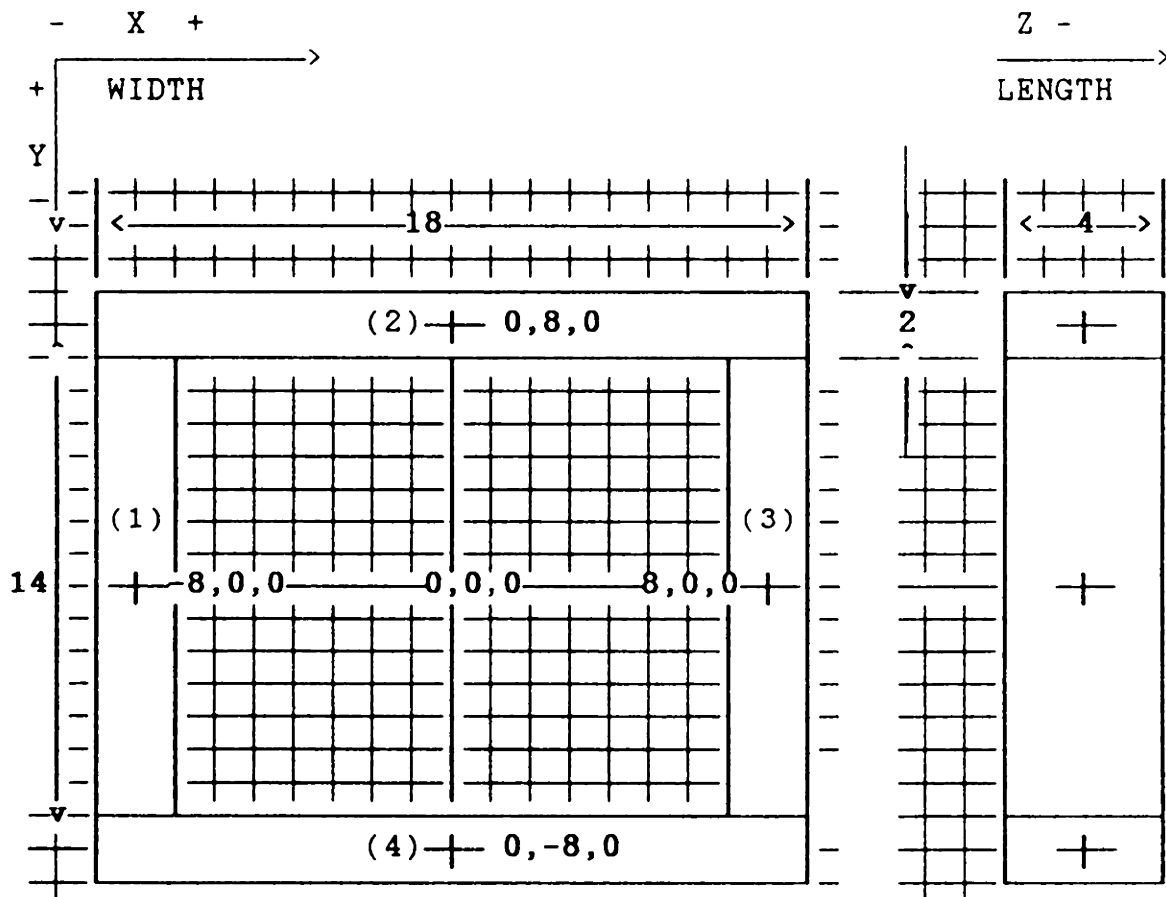
```

CLBEG  1          --- begin Command List 1
MDTRAN  -8.0,  0.0,  0.0  --- move to -8,0,0
PSOLID  4,  2.0, 14.0,  4.0  --- draw block (1)
MDTRAN  8.0,  8.0,  0.0  --- move 8,8,0
PSOLID  4, 18.0,  2.0,  4.0  --- draw block (2)
MDTRAN  8.0, -8.0,  0.0  --- move 8,-8,0
PSOLID  4,  2.0, 14.0,  4.0  --- draw block (3)
MDTRAN -8.0, -8.0,  0.0  --- move -8,-8,0
PSOLID  4, 18.0,  2.0,  4.0  --- draw block (4)
CLEND          --- end List
CLRUN  1          --- run List 1

```

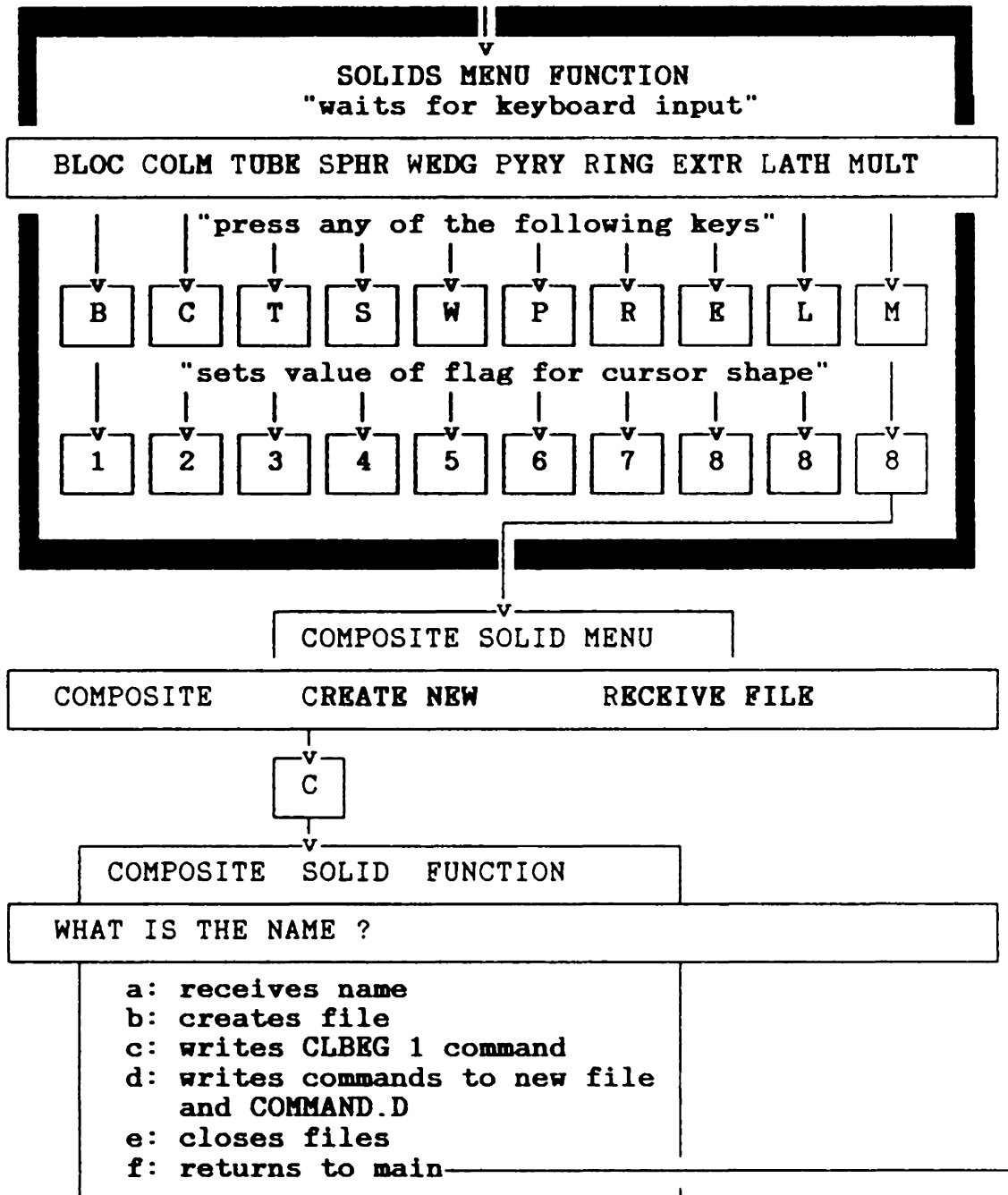
FRONT VIEW

RIGHT SIDE VIEW



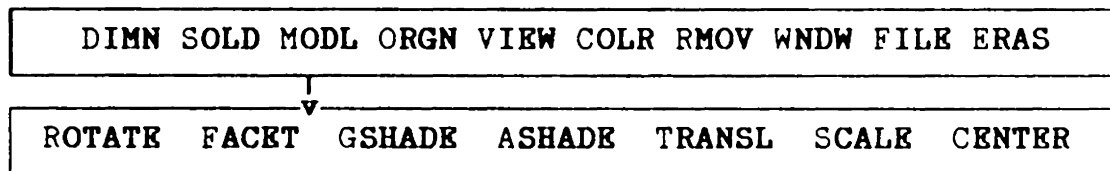
CREATE COMPOSITE SOLIDS FUNCTIONS:

EXAMPLE :



MODELING MENU :

The Modeling Menu gives access to a set of functions that will effect certain attributes associated with a primitive or object. These attributes are presented at the top of the screen after the user presses the key "m" for the modeling menu. The modeling menu screen display will be drawn at the top of the screen, like the figure below, the set of abbreviations that are displayed will correspond to a set of modeling functions.

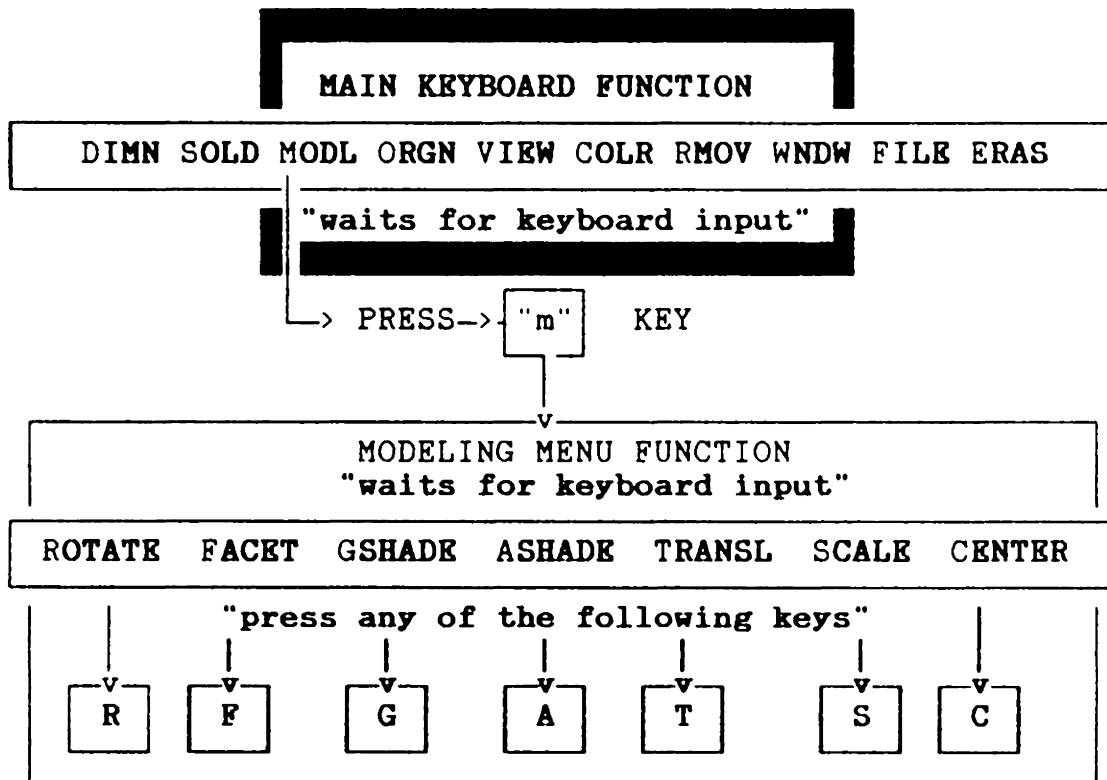


ROTATE -- OBJECT ROTATION ON THE X, Y OR Z AXIS
 FACET -- FACETS SOLID PRIMITIVES TO A GREATER OR LESSER DEGREE OF ACCURACY
 GSHADE -- SETS TO GOURAUD SHADING
 ASHADE -- SETS TO CONSTANT SHADING
 TRANSL -- SETS TRANSLUCENCY INDEX
 SCALE -- SCALES OBJECTS ON THE X, Y OR Z AXIS
 CENTER -- RESETS AND CENTERS MODELING MATRIX

Each of these functions also has its own command they are as follows:

ROTATE -- MDROTX num, MDROTY num, MDROTZ num
 FACET -- DISPAT 4 num {0...7}
 GSHADE -- DISPAT 0 1
 ASHADE -- DISPAT 0 0
 TRANSL -- DISPAT 1 num {0...15}
 SCALE -- MDSCAL sx, sy, sz

MODELING MENU BRANCHES:



This set of modeling functions will give access to another set of functions related to the heading, or is only a single function. In order to activate a command press the key of the first letter of the word or abbreviation. If the word represents a function it will be activated, if it branches to another menu it will present the user with a new set of functions.

ROTATION FUNCTIONS:

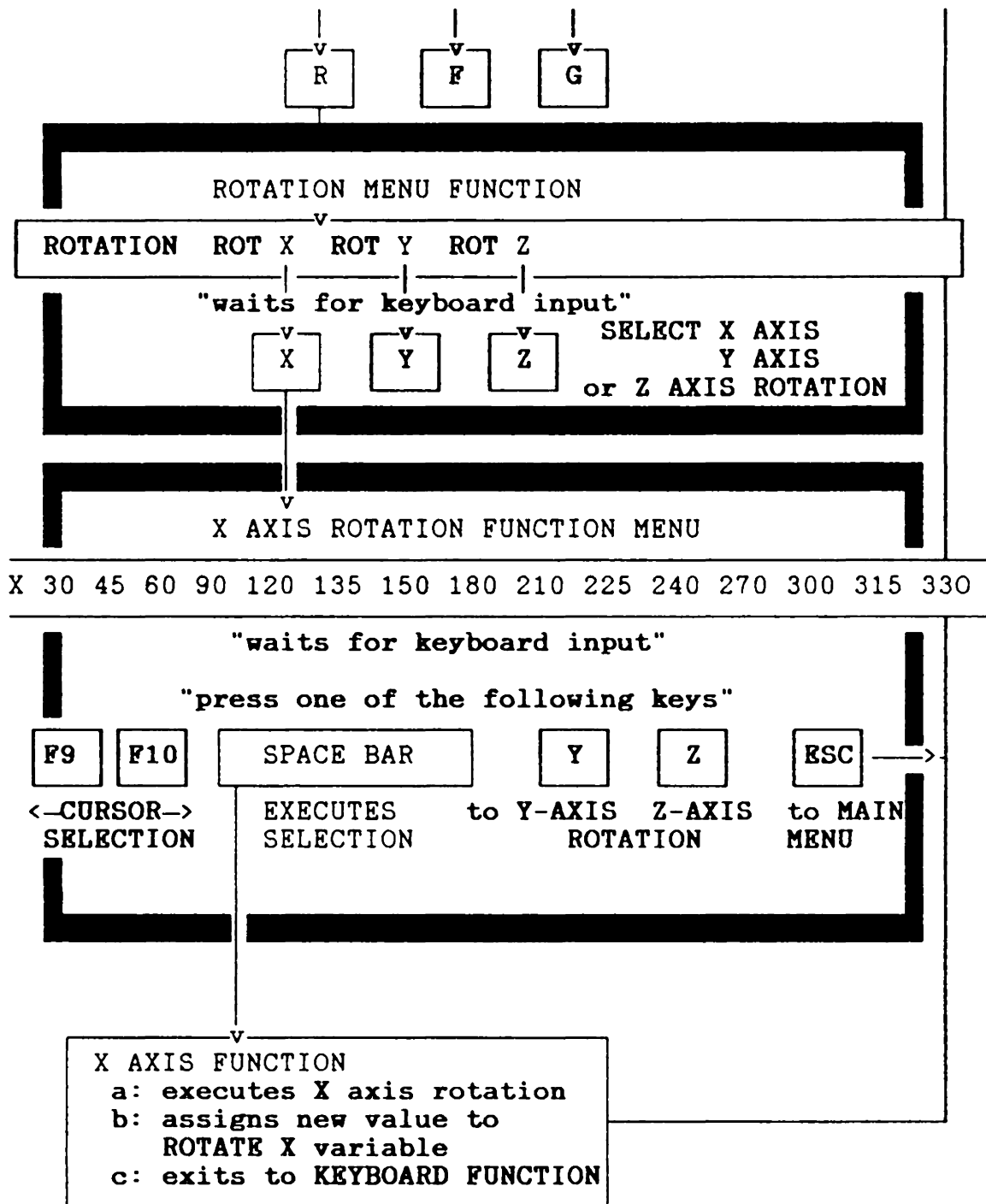
The Rotation Functions branch to sub-functions Rotate X axis, Rotate Y axis and Rotate Z axis. Each of the different axis rotation functions are displayed at the top of the screen, so that the user may choose by pressing either the " X" key for X axis rotation, the "Y" key for Y axis rotation, or the " Z" for Z axis rotation. Once the choice is made the program will enter into that specific rotation function. Once entered into one of three rotation functions another menu will be displayed. That menu will consist of a series of numbers that represent different angles that the object will be rotated on. The " F9" and " F10" keys will activate a separate menu cursor that will rotate the object in real time. This effect will allow the designer to make a visual choice. If the designer already knows which angle is to be used the cursor can be placed on the first letter of the menu. This will activate a query

function that will receive any angle that is desired. Each axis rotation function can be accessed from the other simply by pressing the letters X, Y or Z. This will eliminate the need to return back to the main rotation function. Also, translation and grid space commands are available within each function. This will give the user the ability to create models freely that would require multiple rotations of a single element, such as a spiral stairway. In order to exit back to the main menu the the "Esc" should be pressed.

Once a rotation values been set they are stored in a variable, each axes has its own variable. If the variable is greater than zero then the rotation commands will be written with the solid object it defines, and then another command will also be written to rotate the modeling matrix back to zero. Here is an example block rotated at 45 degrees on the Z- axis :

```
MDTRAN 0.0, 0.0, 0.0      -- locates block at origin
MDORG  0.0, 0.0, 0.0      -- set modeling origin
MDROTZ 45                  -- rotates modeling matrix 45
LOCATE 0.0, 0.0, 0.0      -- locates centroid at origin
PSOLID 4, 10.0, 70.0, 50.0-- draws block
MDROTZ 315                 -- sets modeling matrix to 0
```

PATH OF X-AXIS ROTATION:



FACETING FUNCTION:

The Faceting Function has the ability to draw curved surfaces or solids to a greater or lesser precision. In this graphic system curved and circular objects are drawn with a series of lines or planes that represent that object; therefore, a round column can be represented as a six sided column in gouraud shading and may be drawn very quickly with little accuracy, or may be drawn slowly having sixty sides with a greater amount of accuracy. Another useful aspect of the Faceting function is the ability to draw an eight, six or twelve sided column just by adjusting the faceting angle. If this is the desired outcome then the Constant Shading Function should be used. The Faceting Function and shading functions work together to create different attributes associated with the manner in which the object is drawn. The Faceting Function is activated by the modeling menu branch, and access is achieved by pressing the "m" key, then the "f" key. Once the faceting function has been entered it will present its own menu, that menu consist of a series of numbers that represent the faceting angle. If the "F9" or "F10" keys are pressed they activate a menu cursor that will select the desired angle.

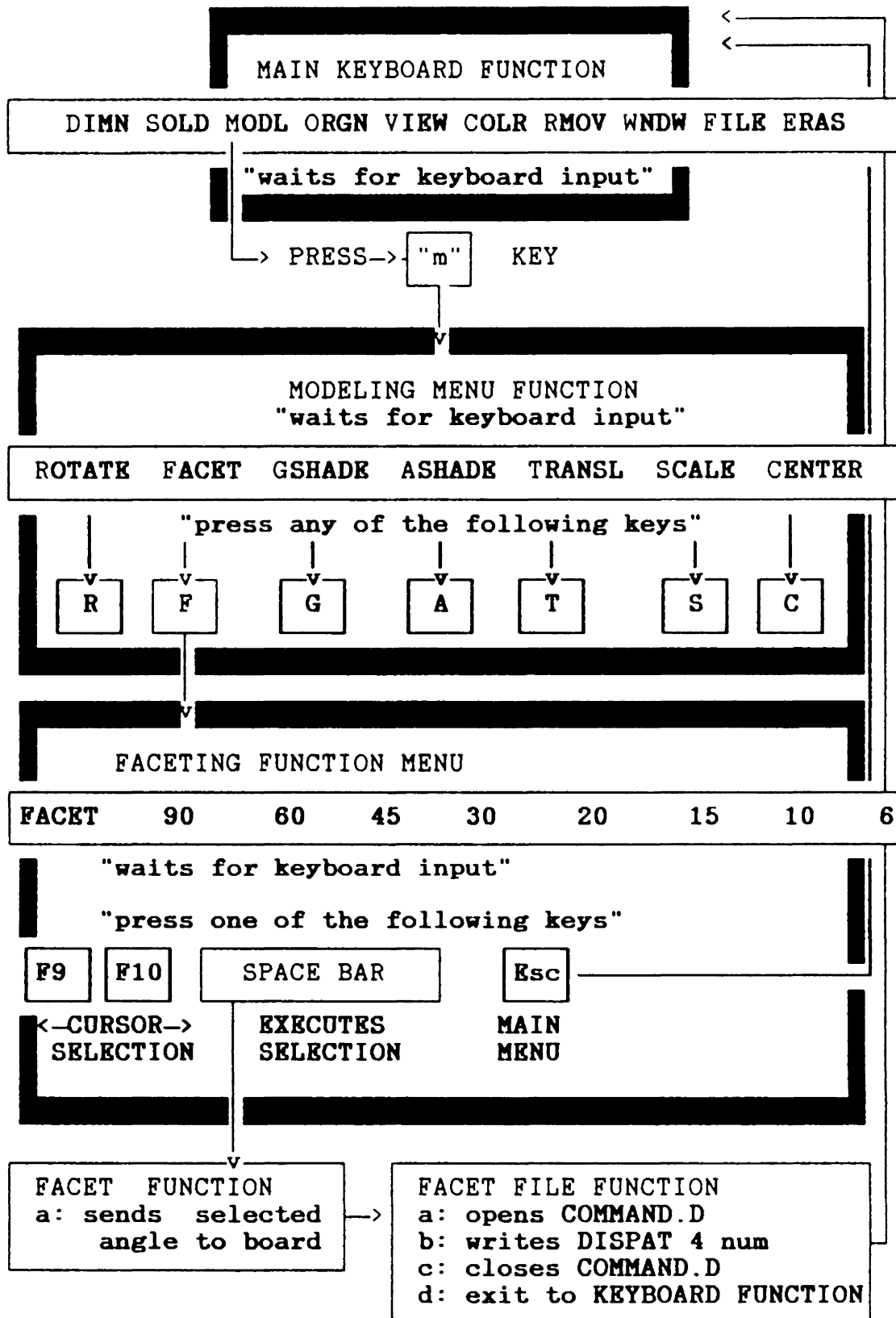
The command for faceting is Dispat 4 num, num may be any number from 0...7. The higher the number the more accurate the model and the slower the drawing speed. The number will also give a particular number of sides to a circular edge. They are as follows :

NUM	ANGLE	NUMBER OF SIDES TO CIRCULAR EDGE
0	90	4
1	60	6
2	45	8
3	30	12
4	20	18
5	15	24
6	10	36
7	6	60

An example command list for an eight sided column would be :

```
DISPAT 4, 2      -- sets to eight sided  circular edge
DISPAT 0, 0      -- sets to constant shade
LOCATE 0.0, 0.0, 0.0  -- locates solid at the origin
PSOLID 0, 35.0, 130.0 -- defines solid as a column
```

PATH FOR FACETING FUNCTION:

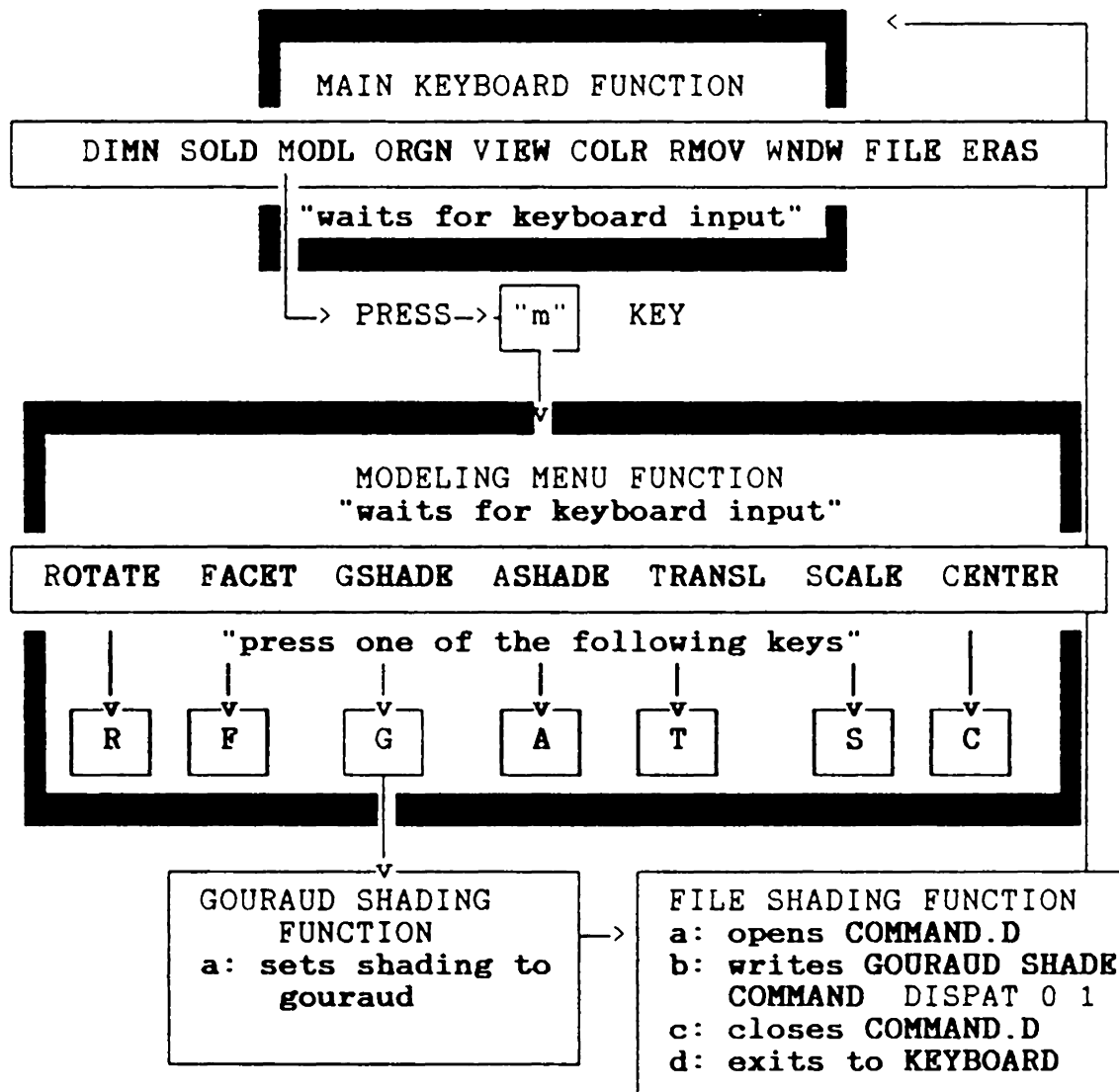


GOURAUD SHADING FUNCTION:

The Gouraud Shading function is represented by the abbreviation GSHADE. This type of shading performs a blending of all the surface planes so that the object has a very smooth surface appearance. The Shading Function is very useful in creating round smooth surfaces. It is also used in conjunction with the Faceting Function when drawing time is important, example: a sphere drawn with gouraud shading at a faceting of fifteen degrees will appear smooth and round, and may be drawn quickly. At the same time a sphere drawn at six degree facets will have a more accurate spherical shape, but it will take over twice the amount of time to draw. When a model that has several hundred circular objects in it is being redrawn, it may be an advantage to use a greater faceting angle in the design development stage and to use a higher accuracy image for presentation purposes. In order to activate gouraud shading the user must first enter into the Modeling Menu then press the "g" key, this will automatically set the shading mode and all other objects will be drawn in gouraud shading until a change is made.

The Gouraud shading command is DISPAT 0 1. Once placed in the Command list all following objects will be drawn in Gouraud Shading.

PATH FOR GOURAUD SHADING FUNCTION:

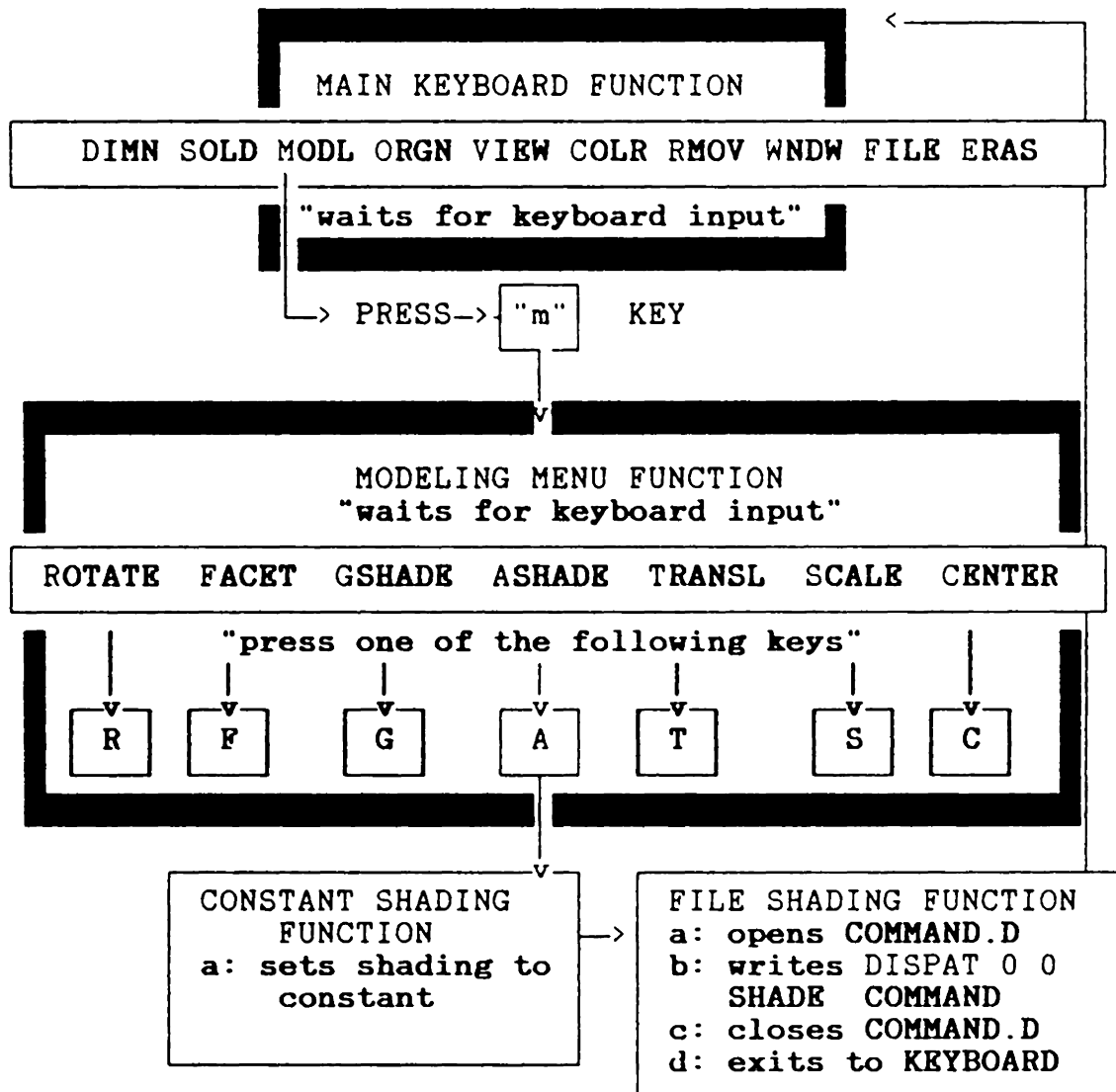


CONSTANT SHADING :

The Constant Shading or Angular Shading function is represented by the Abbreviation ASHADE. This type of shading shades each surface patch separately, with this shading technique all angles facets are made visible. By using Constant Shading the sphere primitive can represent a tetrahedron, octohedron, decahedron, dodecahedron and more simply by adjusting the faceting angle. Cones, columns, rings and tubes can also be faceted to make a wide range of primitives that have four to sixty-four equal sides. In order to activate constant shading the user must first enter into the Modeling Menu by pressing the "m" key then press the "a" key.

The Constant shading command is DISPAT 0 0. Once placed in the Command list all following objects will be drawn in Constant Shading.

PATH FOR CONSTANT SHADING FUNCTION:



TRANSLUCENCY FUNCTION :

The translucency function transforms a solid opaque object into one that has a different level of translucency. There are five levels of translucency that may be activated, they are : 100% opaque, 50% opaque, 25% opaque and 12.5% opaque. Each level then has different patterns of dithering that the translucency mask is applied to. The 50% level is the simplest and has two levels. Below is a diagram showing two complementary masks on a 2 X 2 pixel area of an image.*

X	O
O	X

LEVEL 1

O	X
X	O

LEVEL 2

O = DO NOT CHANGE

X = DRAW THE PIXEL

If level 1 is applied then a 50% translucent object is drawn. Level 2 will also draw a 50% mask and is the compliment of level 1. The advantage to this is that the two can then be combined. For example if a red translucent object is in front of a yellow translucent object, the viewer would see orange where the two overlap. If both use level 1 then one would mask the other and the first drawn object would only be seen.

The 25% mask works on the same method, but has four different patterns, they are diagramed as follows:

O = PIXEL IS NOT DRAWN X = PIXEL IS DRAWN

LEVEL 3

X	O	X	O	X	O	X	O
O	O	O	O	O	O	O	O
X	O	X	O	X	O	X	O
O	O	O	O	O	O	O	O

LEVEL 4

O	X	O	X	O	X	O	X
O	O	O	O	O	O	O	O
O	X	O	X	O	X	O	X
O	O	O	O	O	O	O	O

LEVEL 5

O	O	O	O	O	O	O	O
X	O	X	O	X	O	X	O
O	O	O	O	O	O	O	O
X	O	X	O	X	O	X	O

LEVEL 6

O	O	O	O	O	O	O	O
O	X	O	X	O	X	O	X
O	O	O	O	O	O	O	O
O	X	O	X	O	X	O	X

The 25% mask allows four translucent objects to be seen overlapping each other. One pixel is drawn for every 2x2 pixel area. The values are from level 3-6.

The 12.5% translucency mask values are from 7-14 this mask allows eight different planes to viewed through each other, two pixels are drawn for every 4 x 4 pixel area.

LEVEL 7 - 14

the number level selected will draw the
corresponding pixel number.

7	12	14	9	7	12	14	9
11	8	10	13	11	8	10	13
14	9	7	12	14	9	7	12
10	13	11	8	10	13	11	8

Another advantage to translucency is that the color table can be increased by blending colors. The 50% mask allows the blending of two colors in the color table while the 25% mask would allow four colors to be blended while the 12.5% eight.

In order to activate this command the modeling menu must first be entered by pressing "m" then by pressing the "t" key the translucency menu will be shown. The different levels of translucency will be displayed at the top of the screen.

TRANSL 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

By pressing the "F9" and "F10" keys the menu cursor will move to the desired level, or by pressing the desired "num" key, once selected the "space" bar will activate that command. The translucency index will not change unless another selection is made.

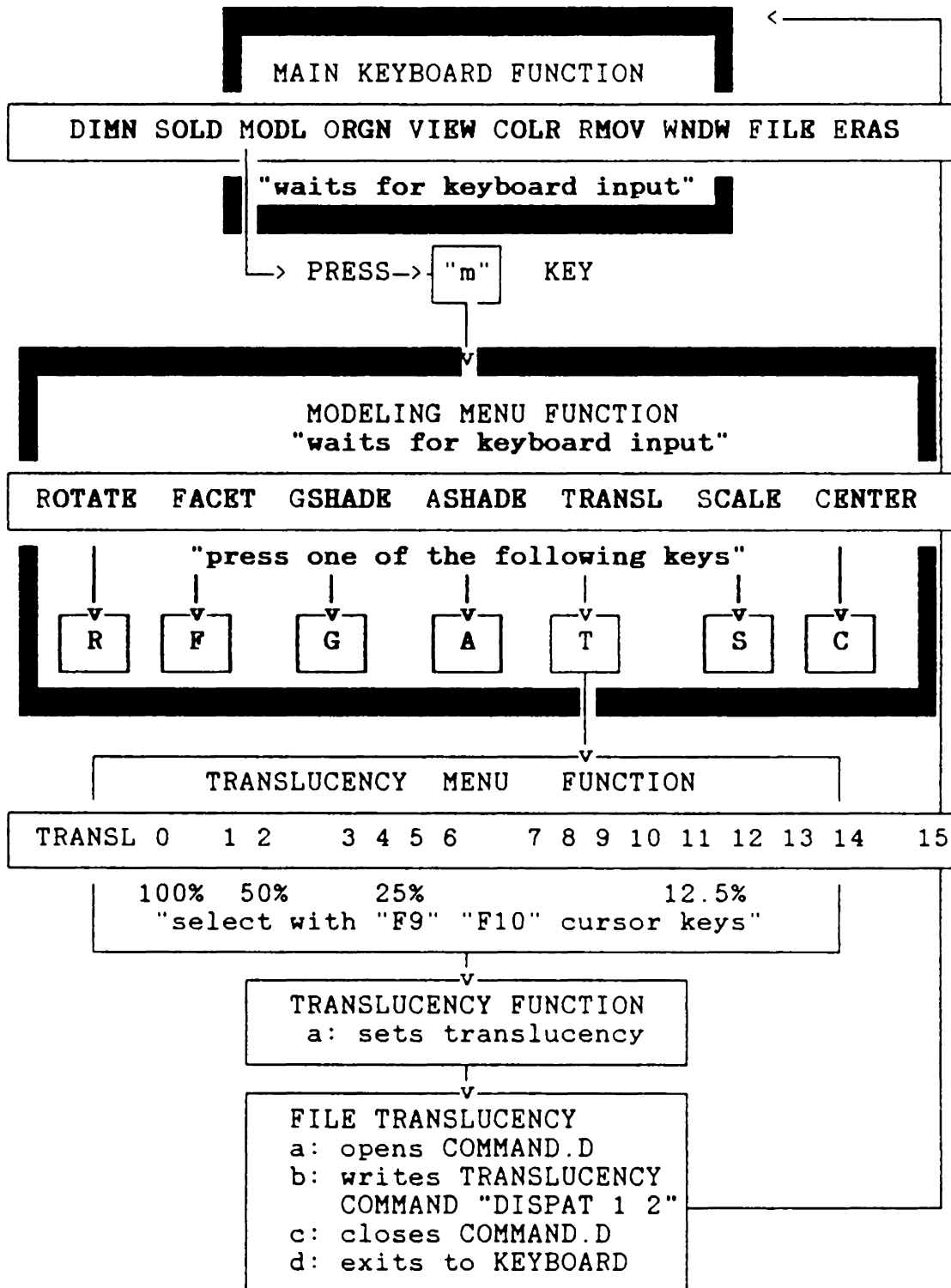
The COMMAND LIST format for translucency is as follows:

DISPAT 1 (level)

example for a translucency of 50%

DISPAT 1 2

PATH FOR TRANSLUCENCY FUNCTION:



SCALING FUNCTION :

The scale functions include scaling on the X, Y and Z axis. Each can be performed all at once or separately. When all three axes are scaled the object will retain the same proportions as the original object. If one axis such as the X axis is scaled then only the dimensions on that axis will be scaled. This is useful in creating new primitives, example: a sphere or a column can be scaled on one axis to become more elliptical in cross section. Scaling differs from the Dimension Function in that the everything on the axis being scaled is changed after the function is activated, while the dimension function acts only on the primitive being drawn. The scaling values will remain on until the function is called again.

The Scale Function is activated by entering into the Modeling Menu by pressing "m" then by pressing "s". Once in the function a scaling menu will be displayed

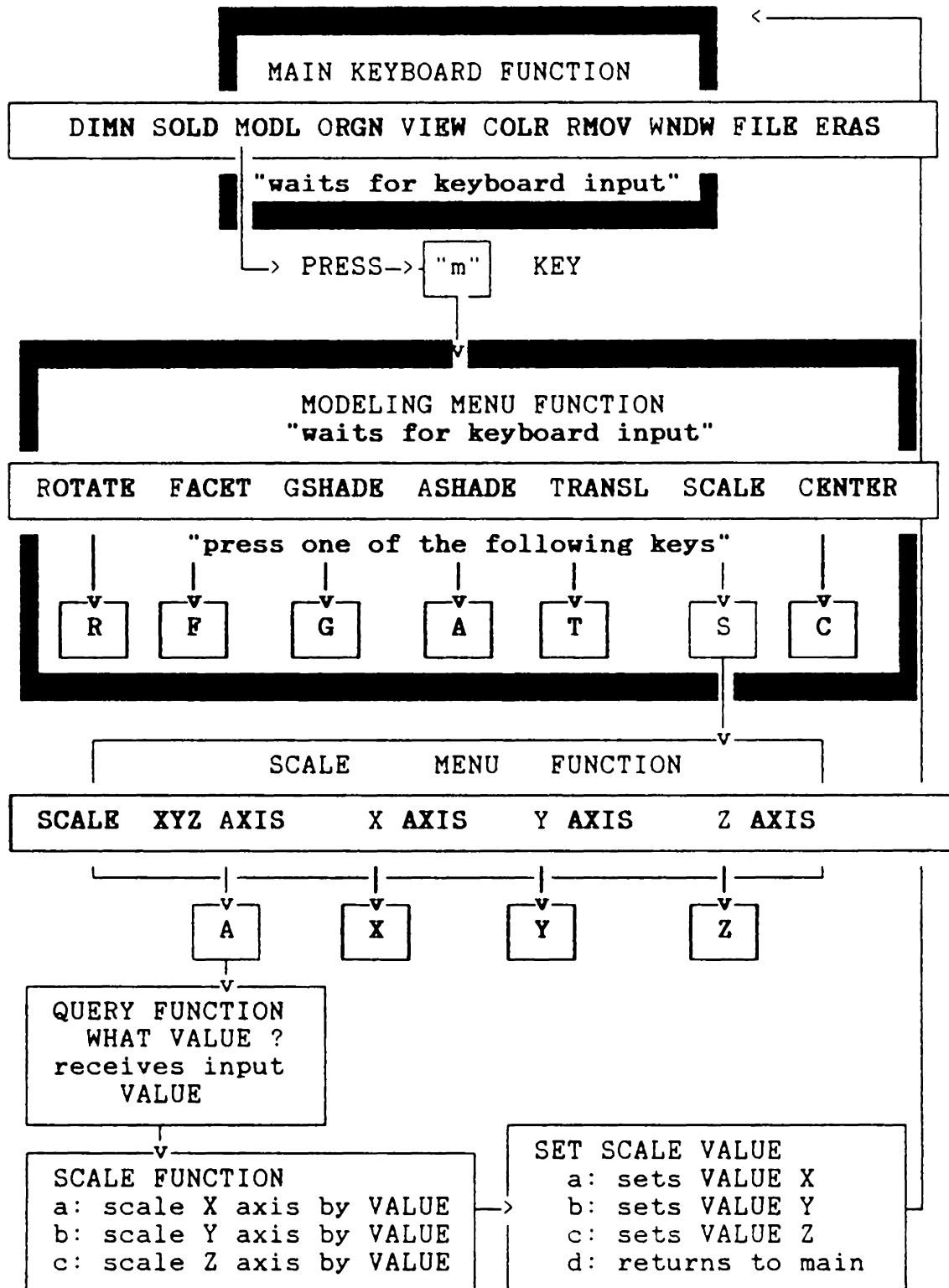
SCALE	XYZ AXIS	X AXIS	Y AXIS	Z AXIS

Once the selection is made by either the menu cursor or by pressing the highlighted key, a query will ask the user for the scaling value.

The COMMAND LIST format is as follows:
MDSCAL 0.5, 0.5, 0.5

This command will scale the X, Y and Z axis by 0.5.

PATH FOR SCALE XYZ AXIS FUNCTION:



CENTERING FUNCTION :

The Centering function acts as a reset button it will set Scaling and Rotation values to there default values. It will also set the cursor back to the origin. The command MDIDEN is used to reset the modeling matrix. Once it is called all translation, rotation and scaling value are set to their default values.

ORIGIN SET :

The Origin Set Function is closely associated with rotation and scaling functions. This function allows the User to set a particular point in which to rotate an object about, or be able rotate an object on end. Without this function objects will only be able rotate around there centroid; however, by being able to move the point of rotation, complex composite solids may be constructed quite easily. An excellent example would again be the construction of a spiral stairway. The same object may be rotated around the same Y axis while only increasing the Y value. In order to activate MDORG press the "o" key, this will save the current location, and enter into access to the Translation functions. The user may then move the cursor to desired point. Once this is accomplished the "space" bar should be pressed, this will record the new point and the difference between the two points. An example of a rotated block on end is as follows:

COMMAND LIST EXAMPLE FOR MDORG:

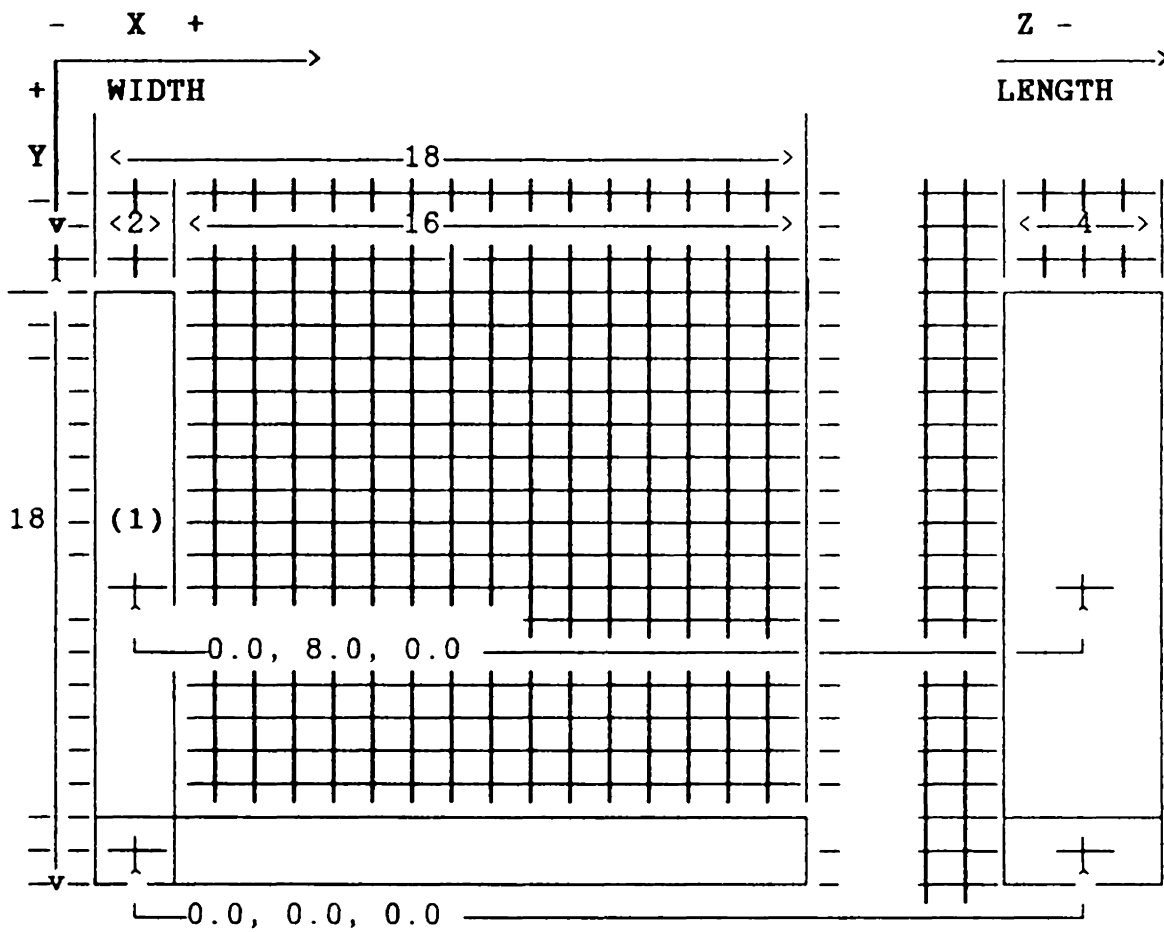
```

MDTRAN 0.0, 0.0, 0.0
LOCATE 0.0, 0.0, 0.0
PSOLID 4, 2.0, 18.0, 4.0
MDTRAN 0.0, 8.0, 0.0
MDORG 0.0, 0.0, 0.0
MDROTZ 270
PSOLID 4, 2.0, 18, 4.0
MDROTZ 90

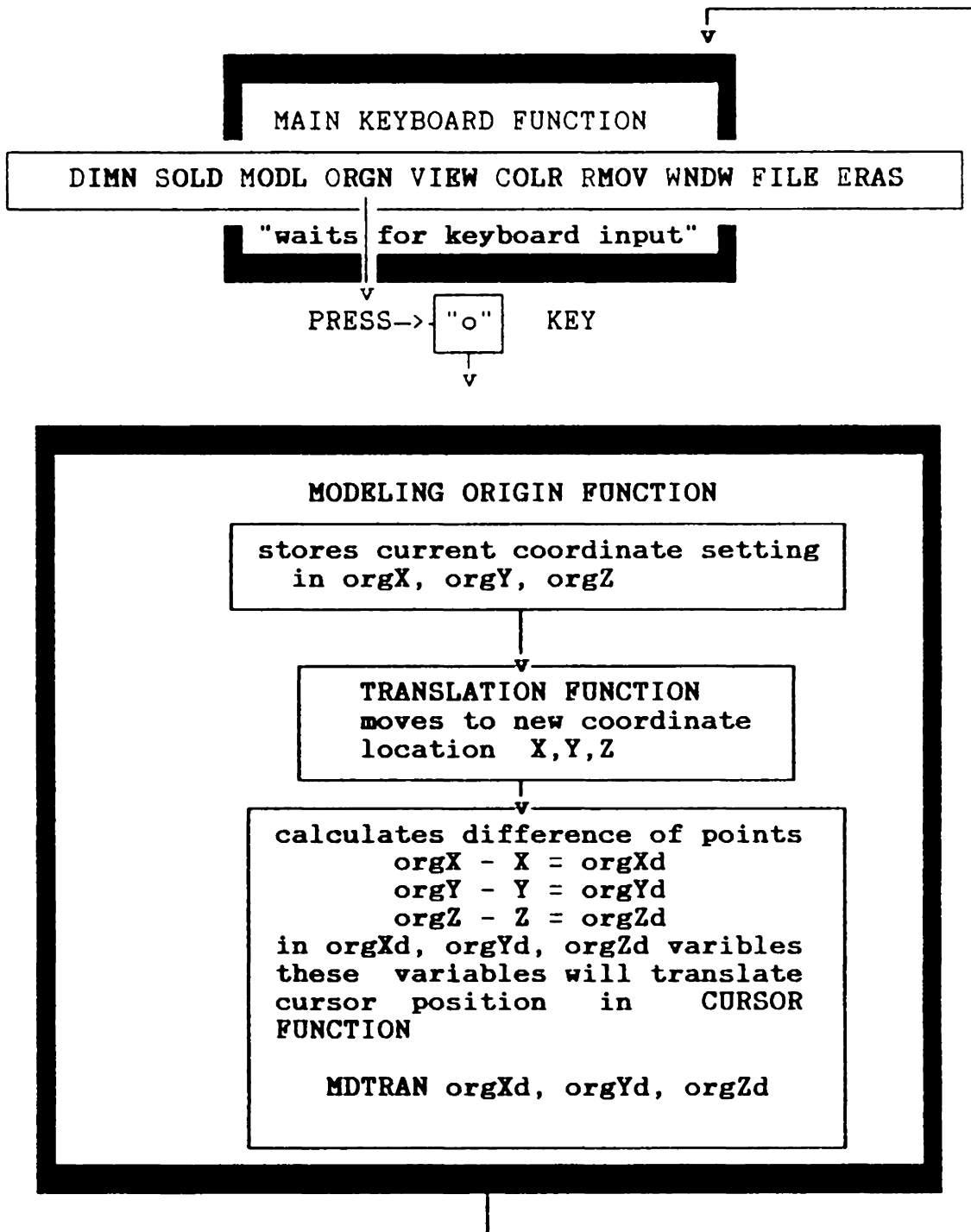
```

FRONT VIEW

RIGHT SIDE VIEW



PATH FOR MODELING ORIGIN FUNCTION:



VIEWPORT FUNCTIONS :

The viewport function include those functions that allow viewing transformations, these function do not alter the model in any way, they only allow the viewer to have different view points of the model. The program will allow the user to view the model from any angle, from an outer or inner perspective. This will allow a walk through perspective of a model. The functions that are included are as follows:

```

ROTATE X  -- rotates the model view around the X axis
ROTATE Y  -- rotates the model view around the Y axis
ROTATE Z  -- rotates the model view around the Z axis
DISTANCE --- determines the distance from the view point
VIEWPOINT -- determines the point that is being viewed
WINDOW ---- zoom for orthographic projections

```

These functions work within the file SETUP.D if any changes are made to the viewpoint of the model the entire Command List must be run with the new changes in the setup file. Each function will create a Command in the SETUP.D file, They are as follows:

```

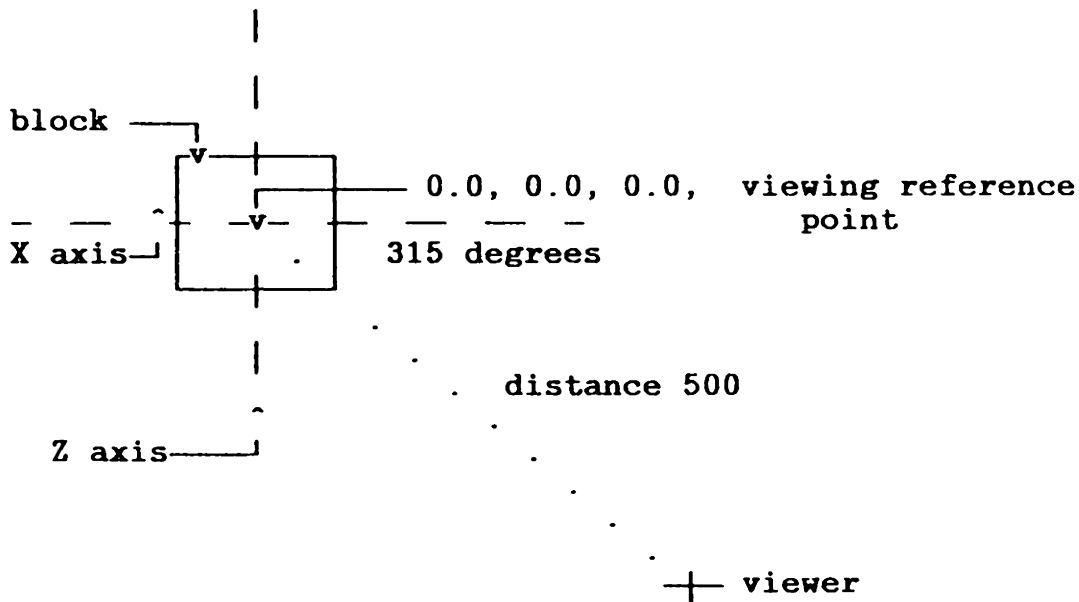
ROTATE X  -- VWROTX angle
ROTATE Y  -- VWROTY angle
ROTATE Z  -- VWROTZ angle
DISTANCE --- DISTAN dist
VIEWPOINT - VWRPT  x, y, z
WINDOW ---- WINDOW x1, x2, y1, y2

```

A COMMAND LIST example for viewing a block at 315

degrees on the Y axis is as follows :

```
VWPORT 0, 640, 0, 480 ----sets the viewport to full screen
WINDOW -640.0, 640, -480, 480 ----- sets window size
PROJECT 50 ----- sets projection angle
DISTAN 500 ----- sets distance from reference point
VWRPT 0.0, 0.0, 0.0----- sets location of reference point
VWROTX 0 ----- sets 0 degrees rotation on x axis
VWROTY 315 ----- sets 315 degrees rotation on Y axis
VWROTZ 0 ----- sets 0 degrees rotation on Z axis
PSOLID 4, 30.0, 30.0, 30.0 ----- defines solid block size
```



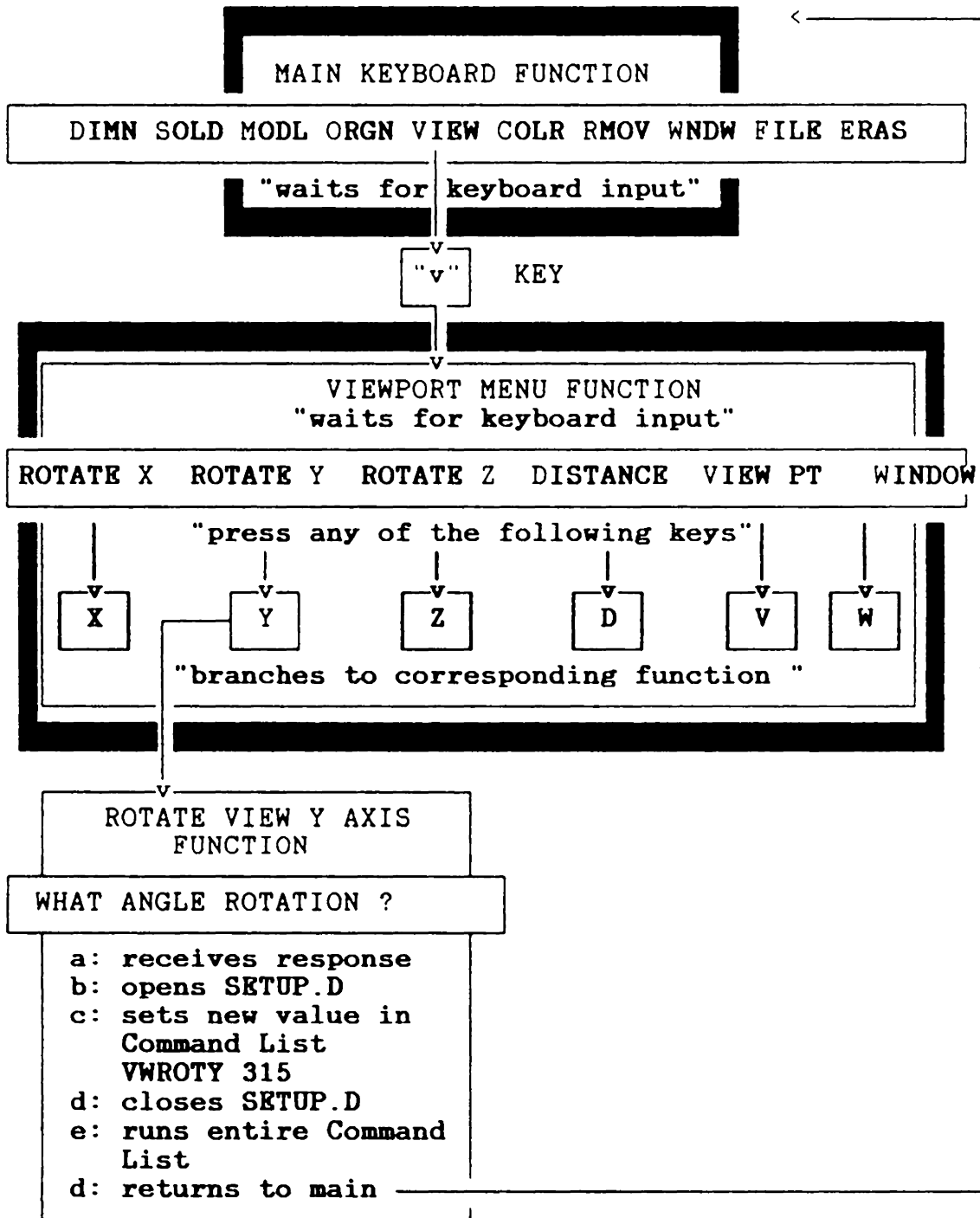
The Viewing Functions are accessed by pressing the "v" key and entering into the View Menu Function. Once in the View Menu the following display will be given:

ROTATE X	ROTATE Y	ROTATE Z	DISTANCE	VIEW PT	WINDOW

To activate a given function the user then must press the highlighted letter.

VIEWPORT FUNCTION DIAGRAM :

EXAMPLE ROTATE VIEW 315 DEGREES ON THE Y AXIS:



COLOR FUNCTION :

The Color Functions are a set of functions that control the color of an object. They are accessed through the Color Menu Function which is activated by pressing the "c" key. Once activated it will display the color menu.

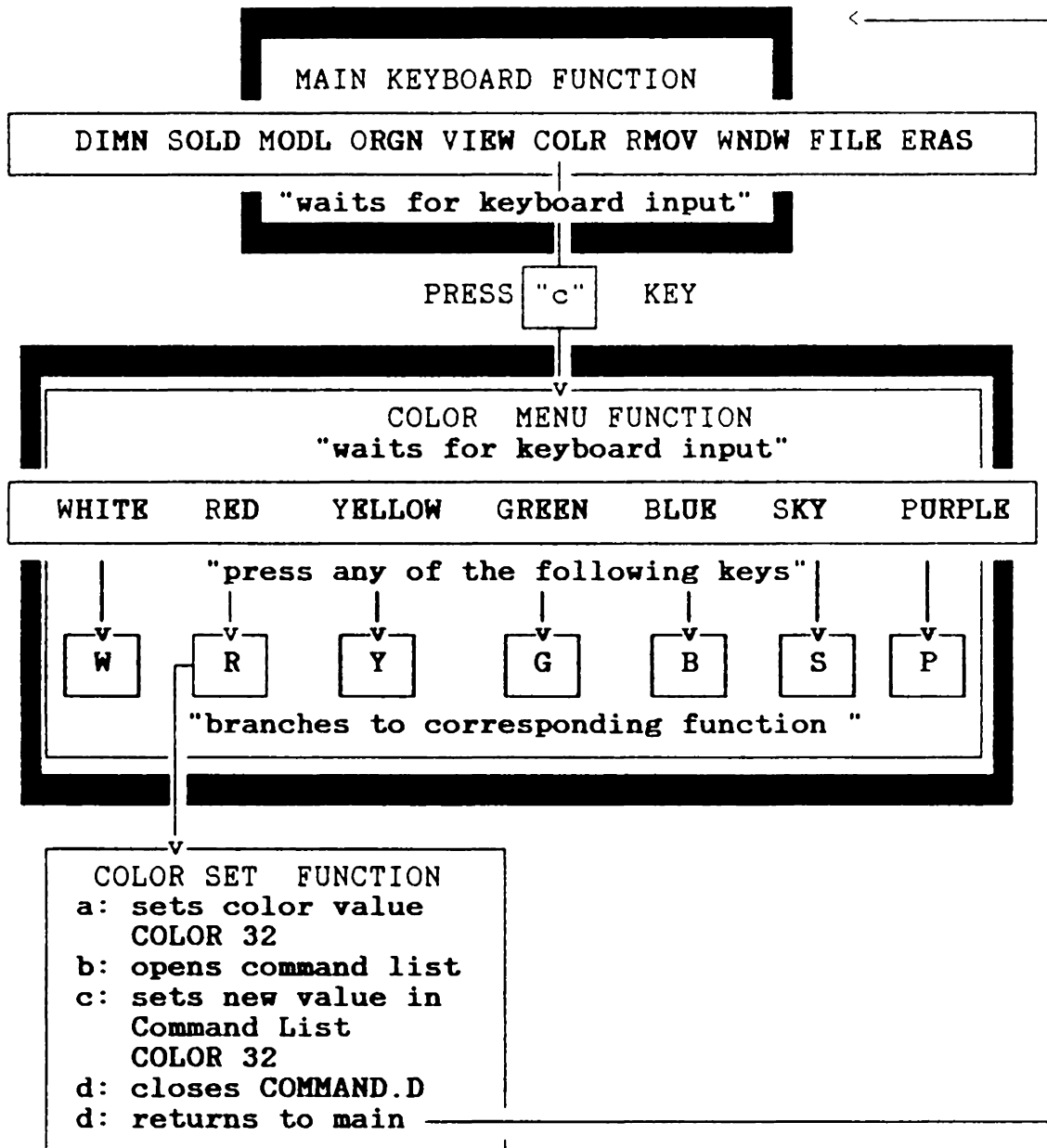
WHITE	RED	YELLOW	GREEN	BLUE	SKY	PURPLE

To select the desired color the user then may press the first letter of that color. All objects drawn after the selection will be drawn in that color until another selection is made. After the selection has been made the color command COLOR num is added to the command list. Once added to the command list the object will be recreated in the selected color. Each color represent 32 shades of that color. This will give a very realistic shading of an object. The maximum colors available on the screen are 256, however this can be increased by using the transparency function and blend two to four colors together. Each color is coded with a number which activates the graphics board to that specific color. The list is as follows:

WHITE	224
RED	32
YELLOW	128
GREEN	64
BLUE	96
SKY	218
PURPLE	252

COLOR FUNCTION DIAGRAM :

EXAMPLE ADD COLOR RED:



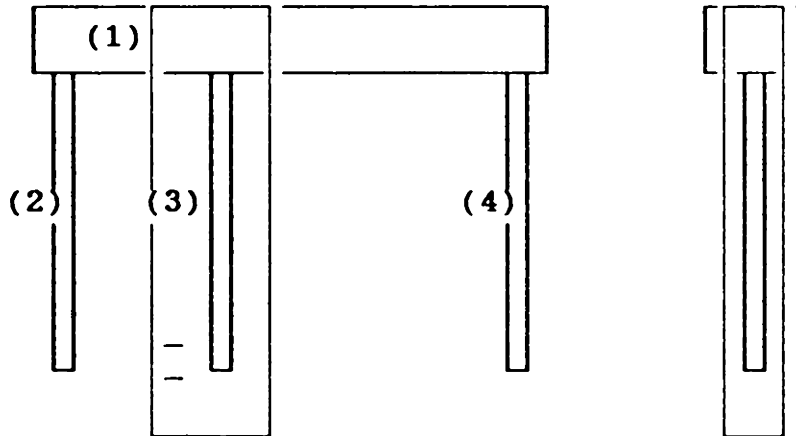
REMOVE FUNCTION:

The Remove Function purpose is to allow the user to delete unwanted objects from the screen. This is a very useful function especially in the design stage when many changes may be made. The Remove Function is activated by pressing the "r" key. Once the key is pressed the cursor will take the shape of a block, this block may then be moved to any location on the screen. When the "r" key is pressed a second time all objects that are inside of the block will be deleted from the screen and the command list. In order to provide a more accurate deleting capability the blocks dimensions may be changed by using the Dimension Function. The function works simply by comparing the coordinates of the deleting block to that of the objects on the screen. Those objects that are within the blocks boundaries will not be redrawn to the screen. All curved shapes such as spheres and cones will be given a rectilinear boundary definition to speed up calculations. For even faster removal the function may be set to locate only the centroid or closest object to the cursor location. An example of boundary definition removal is as follows:

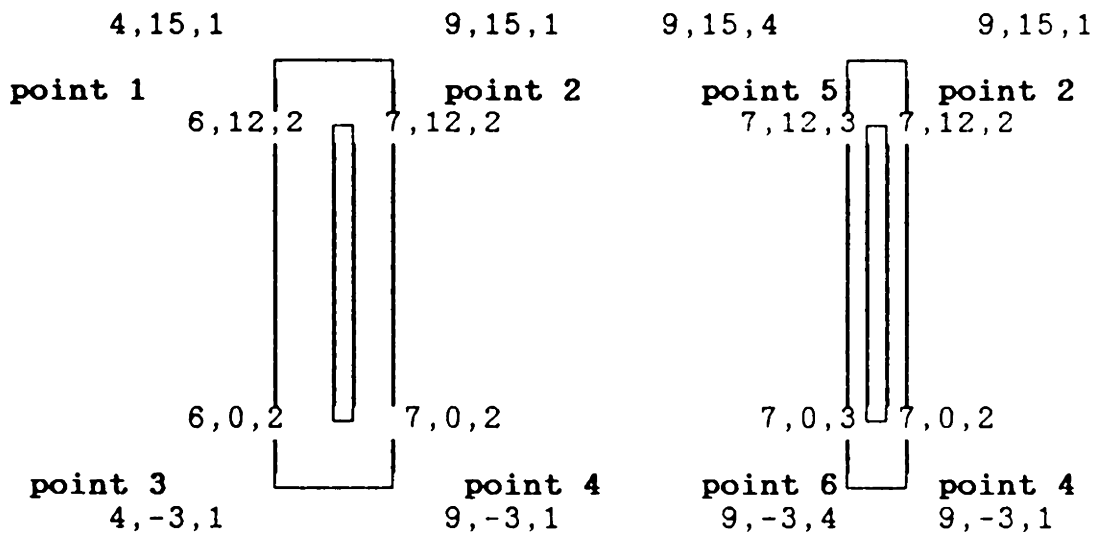
REMOVAL OF OBJECT (3) :

FRONT

RT. SIDE



POINT CHECK: each corresponding point must lie within the boxes boundaries



POINT EVALUATION : each point must test true for the object to be removed.

POINT LOCATION		BOX	TEST	OBJECT
POINT 1 FRONT LEFT TOP	X	4	<	6
	Y	15	>	12
	Z	1	<	2
POINT 2 FRONT RIGHT TOP	X	9	>	7
	Y	15	>	12
	Z	1	<	2
POINT 3 FRONT LEFT BOTTOM	X	4	<	6
	Y	-3	<	0
	Z	1	<	2
POINT 4 FRONT RIGHT BOTTOM	X	9	>	7
	Y	-3	<	0
	Z	1	<	2
POINT 5 BACK RIGHT TOP	X	9	>	7
	Y	15	>	12
	Z	4	>	3
POINT 6 BACK RIGHT BOTTOM	X	9	>	7
	Y	-3	<	0
	Z	4	>	3
POINT 7 BACK LEFT TOP	X	4	<	6
	Y	15	>	12
	Z	4	>	3
POINT 8 BACK LEFT BOTTOM	X	9	>	6
	Y	-3	<	0
	Z	4	>	3

WINDOW FUNCTIONS :

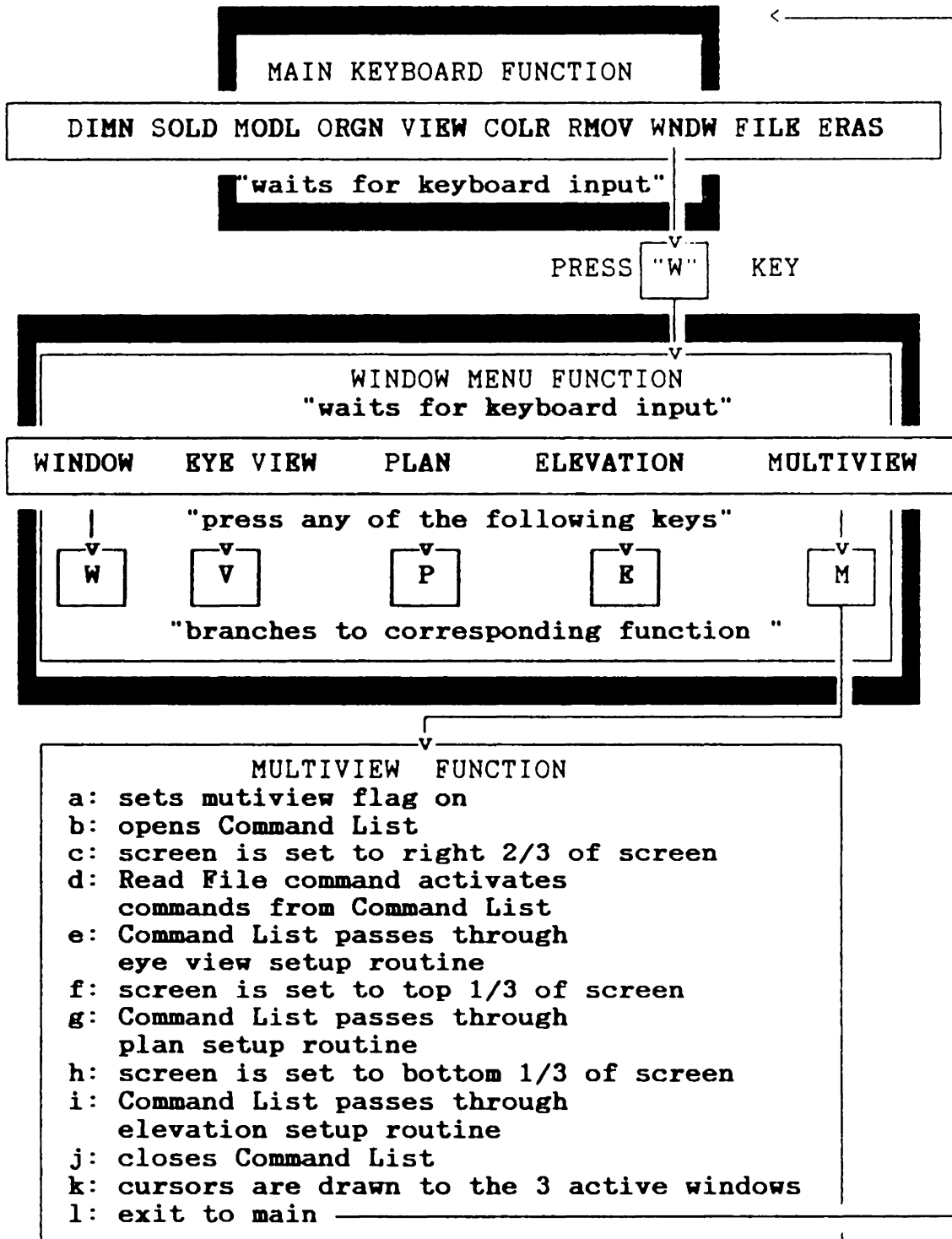
The Window Functions are a set of functions that allow the user to move from orthographic projections to perspective views, also is the ability to set the number of viewing windows from one to three. The purpose of these functions is to give the user multiple views of a model. When in the design stage it is very critical that the designer be able to view an object from several perspectives. While in a perspective view it is nearly impossible to determine the space relationship between two objects without a undistorted form of measurement such as a plan or elevation. The multiple window does just that, it gives a plan, elevation and a perspective view of the model. A cursor is visible in all three window simultaneously.

The function are activated by pressing the "w" key, once this is done the Window Menu will be displayed at the top of the screen.

WINDOW	EYE VIEW	PLAN	ELEVATION	MULTIVIEW

WINDOWS FUNCTION DIAGRAM :

EXAMPLE MULTIVIEW:



FILE FUNCTIONS :

The Filing Functions are the designers way of storing Command Lists of models. The functions consist of two, one to create a new file and one to display an existing file. To activate these functions the user must press the "f" key. Once this is done the menu will query the user, if a new file will be created or an existing file should be displayed. A "c" or "r" key should be pressed. After a key is selected another query will ask for a name. If a file list is needed press the "f" key, all existing work files will be presented with a brief description. The file names may also be filed with a suffix code and put into separate categories. The Create new file works simply by writing the existing COMMAND.D List to the newly created file. The receive function is very similar replaces the existing COMMAND.D file with the selected file.

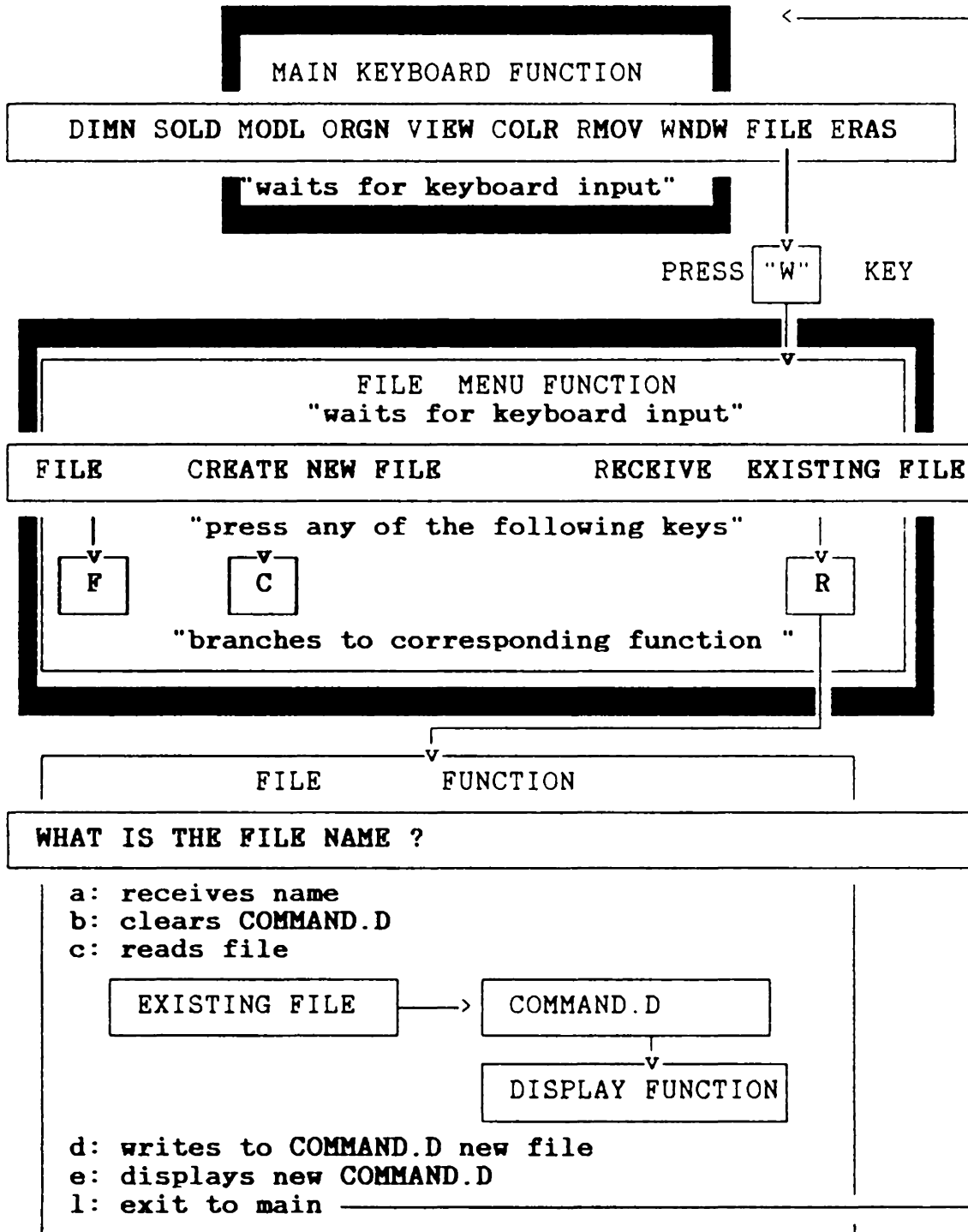
FILE	CREATE NEW FILE	RECEIVE	EXISTING FILE

WHAT IS THE FILE NAME ?

ANY COMMENTS ABOUT THE FILE ?

FILE FUNCTION DIAGRAM :

EXAMPLE RECEIVE FILE:



CONCLUSIONS :

After undertaking such an extensive project a great amount of learning and experience has taken place. One of the biggest lessons has been, that while working with the computer a whole new world has begun to unfold. The computer has become the teacher, a very demanding one that demands a clear structured way of thinking. The computer language itself is a part of this structure. The structure of the language itself seems to open up new ways of thinking, and a greater awareness of the working process of our own mind. After all it is man communicating to a machine, and man creating the structure in which to communicate.

This last year has been my first exposure to computers and their capabilities, and my conclusions are quite simple, its simply one of the tools of the future that will be in almost every working environment, especially in the designers work space. Those of us who choose not to be involved with computer, are actually selecting a very limited future.

If I look at where this program will be in the future I would predict that there will be libraries of subroutines available, which this program could utilize, examples would be complex light shading, and reflection of

surfaces. Ray tracing routines which are done on super computers will be more widely available and a program such as this one would be able to utilize such routines. The menu structure and how they accessed be comes a very important issue, because the menus structure may greatly effect the design process. Future revisions will examine and refine the relationship between design process and the program interface. This program is now dependent on a very sophisticated graphics board, I selected this option because at present solid modeling on standard systems is much too slow. Most of the design work needs to done in wire-frame, then transferred into shaded objects; However, with the use of more powerful hardware solid modeling becomes possible. In a few years this will change, personal computer speeds will go up and prices should come down. Already in one year, computing speed has been greatly improved, and in a few more years personal computers may have the speed of a supercomputer. The main hardware improvements may probably always be speed, memory and resolution. The Software improvements will focus on what this thesis has investigated, how we are able to

communicate with the computer in terms of design. I believe this thesis has dealt with many design issues, and the end result is a program that may be used to create a 3D model in a way that the designer may understand; however, I must add this is just the beginning. This thesis is more an expression and investigation of a design process rather than an end product.

BIBLIOGRAPHY

- Martin, D., Prata, S., and Waite M., **C Primer Plus**, Howard W.Sams & Company, Indianapolis, Indiana, 1987
- Lafore, R., **Microsoft C Programming for the IBM**, Howard W.Sams & Company, Indianapolis, Indiana, 1987
- Prata, S., **Advanced C Primer ++**, Howard W.Sams & Company, Indianapolis, Indiana, 1987
- Wolverton, V., **Running MSDOS**, Microsoft Press, Redmond Washington, 1985
- Matrox Electrical Systems Limited, **Space Machine 640 Users Manual**, Matrox Electrical Systems LTD, 1987

APPENDIX :

SOURCE CODE IN C FOR SOLID MODELING PROGRAM

```

/* MAIN FUNCTION-KEYBOARD-SETSCREEN-MAINVAR-PLAOTVAR-MAINVARS-WINDOWS-CCLKARI-RANIDSK-MCASE-ABS-MCENTER-
CCLKARS-CVRP T-CCLKARF-ORGSET*/
#include "std.h" /*CALLS EXTERNAL FILES*/
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
#include "keys.h"
#define LIM 40
#define MAXLIN 81
#define WHITE 255
int vx1, vx2, vy1, vy2; /* VARIABLES FOR FULL SCREEN VIEWPORT*/
int vx1p, vx2p, vy1p, vy2p; /* STORES UPPER RIGHT WINDOW */
int vx1e, vx2e, vy1e, vy2e; /* STORES LOWER RIGHT WINDOW */
int vx1v, vx2v, vy1v, vy2v; /* STORES LEFT WINDOW */

int rycnt, rzcnt, rxcnt; /* VAR FOR ROTATION ORDER*/
int rcnt, rcnty, rcntz;
int rxt, ryt, rzt; /* STORES TOTAL ROTATION VALUES */
int vx, vy, vz; /* HOLDS VIEW WINDOW ROTATION VALUES */
int rz, rz2, ry, ry2, rx, rx2; /* HOLDS ROTATION VALUES */
int tw0, tw1, twy0, twy1, twe0, twe1; /* STORES TEXT WINDOW SIZE*/

int returnval; /* RETURNS INTEGER VALUE FROM STRING*/
int firstcall; /* USED IN TA1A.-FILEPRIME(), SOLM.-SSHEMU2();BUILDS SWEPT SOLID COMMAND
LIST */

int vectotal; /* TA1A.,MAINVAR; TA2V GETFILE(),SSOLIDR(),CPLAN(),CELEVAT(),SHOP. -- COUNTS
NUMBER OF VECTORS IN SWEPT SOLIDS */

int rxd, ryd, rzd; /* TA1A.,CPRIME2(),FCNTX(); STORES DIFFERENCE OF GIVEN ANGLE */
int sscount; /* TA2V.,GETFILE(),SSOLIDR(),CPLAN(),CELEVAT(); CONTS NUMBER OF PASSES
THROUGH SSOLID FUNCTIONS */

int ssnum; /* TA2V.,SSOLIDR(),CPLAN(),CELEVATE(),; SHOP.,LATHE(),EXTUDE(); DETERMINES
SSOLID TYPE */

int total; /* TA2V.,GETFILEO(),SSOLIDR, CPLAN(),CELEVAT();COUNTS NUMBER OF VECTORS
PASSED THROUGH FUNCTION*/

int ssct; /* TA2V.,GETFILEO(),SSOLIDR, CPLAN(),CELEVAT();COUNTS NUMBER OF VECTORS
PASSED THROUGH FUNCTION*/

int vtot; /* TA2V.,GETFILEO(),SSOLIDR, CPLAN(),CELEVAT();COUNTS NUMBER OF VECTORS
PASSED THROUGH FUNCTION*/

int cl, cli; /* TAFILES.,setup(); HOLDS CLEAR SCREEN VARIABLES */
int tw; /* TAFILES., setup(); text window visible */
int proj; /* TAFILES., setup(); SETS PROJECTION ANGLE */
int lten1, lten2; /* TAFILES., setup(); ENABLE LIGHT SOURCE */
int wdw; /* SETS SCREEN TO SINGLE OR MULTIPLE */
int bcol; /* SETS BACKGROUND COLOR */
int ltnum, ltmax; /* SETS LIGHT NUMBER AND SHADES AVAILABLE */
int col; /* SETS COLOR */

```

```

int curs;          /* DEFINES CURSOR TYPE */
int planon;        /* HOLDS VIEW , PLAN, ELEVATION NUMBER */
int tess;          /* HOLDS FACETING VALUE */
int menur;         /* TA3A.,rotate(),;SETS ROTATION MENUS */
int shad;          /* SETS SHADING VALUE */
int exton;         /* SOLM. ; TURNS ON EXTRUDE SOLID */
int randisk;       /* SETS RANDISK */
int orgv;          /* TACURS. ; ACTIVATES ORIGIN FUNCTIONS */
int cursnum;       /* SETS CURSOR SOLID TYPE */
int cliston;       /* SETS COMMAND LIST ON */
int vali;
int faceton;       /* SET FACETING ON */
int bop;           /* FLAG FOR SUBTRACTION */
int bolcount;      /*talaBOOLEAN COUNT*/
double returnval;  /* RETURN INTEGER VALUE FORM STRING */
double zoomval;    /* INCREASES OR DECREASES WINDOW SIZE */
double xd,yd,zd;   /* DIFFERENCE FROM LAST POINT ENTERED */
double orgxd,orgyd,orgzd; /* DIFFERENCE FROM ORIGIN TO PRIMITIVE CENTROID*/
double sx,sy,sz;   /* HOLDS SCALING VALUES */
double rotnum;     /* SHOP., TA2., TA2V.; VALUE FOR ROTATION OF TURNED SOILDS*/
double x,y,z;      /* CURSOR LOCATION INDICATORS*/
double speed;      /* GRID SPACING */
double dist;       /* VIEWING DISTANCE */
double amb,ltx,lty,ltz; /* LIGHTING DIRECTION */
double vpptx,vppty,vpptz; /* TAFILES. setup(); VIEWING REFERENCE POINT */
double winx1,winx2,winyl,winyl2; /* SET WINDOW SIZE */
double disy,dish;  /* SET CLIPPING PLANE DISTANCE */
double xf,yf,zf;   /* TA1A.; STORES PREVIOUS X,Y,Z LOCATION */
double height,width,length; /* PRIMITIVE DIMENSIONS */
double orgx,orgy,orgz; /* MODELING ORIGIN LOCATION*/
double depth1,depth2,depth3; /* SHOP.,TA2V., ; HEIGHT OF SWEPT SOLID */
double orgxt, orgyt, orgzt; /* DIFFERENCE FROM ORIGIN TO CENTROID OF PRIMITIVE */
double xlistso;    /* X VALUE LIST FOR SWEPT SOLIDS */
double ylistso;    /* Y VALUE LIST FOR SWEPT SOLIDS */
double *xptr;      /* X POINTER FOR VALUE LIST FOR SWEPT SOLIDS */
double *yptr;      /* Y POINTER FOR VALUE LIST FOR SWEPT SOLIDS */
double px1, py1, pz1; /* STORES PERIMETER VALUES FOR DELETE OBJECT */
double px2, py2, pz2;
double px3, py3, pz3;
double px4, py4, pz4;
double px5, py5, pz5;
double px6, py6, pz6;
double px7, py7, pz7;
double px8,py8, pz8;

```

```

BOOL main()                                /* BEGINNING OF MAIN FUNCTION */
{                                           /* GLOBAL VARIABLE DEFINITIONS */
extern int rx,ry,rz;
extern int rcntx, rcnty,rcntz;
extern double xd,yd,zd;
extern double x, y, z, speed;
int key ;
FILE *cl;
pginit(0,1);                               /* TURNS ON SN-640 */
pgcold();                                  /* RESETS BOARD MEMORY */
cl = fopen("command.d", "w");             /* DEFINE cl TO OPEN COMMAND FILE */
fclose(cl);                               /* CLOSE FILE */
mainvar();                                /* DEFINES ALL GLOBAL VARIABLES */
cleari(0);                                /* CLEARS SCREEN */
maindisp();                               /* DISPLAYS MAIN MENU */
setscrn();                                /* SETS SCREEN COORDINATES */
keyloop();                                /* KEYBOARD LOOP */
displa(1);                                /* TURNS ON CGA MODE */
} /* end main */
/*****/
keyloop()
{
extern double x,y,z,speed,xd,yd,zd;
extern int rx,ry,rz,rcntx,rcnty,rcntz;
int key;
key = getche();                           /* GETS CHAR FROM KEYBOARD */
while(key != 113)                         /* TEST FOR "q" */
{
    keyboard(key);                        /* OPENS KEYBOARD FUNCTION */
                                        /* PRINTS CURSOR LOCATION AND ROTATION INFORMATION */

    printf(
"\n xd = %.2f   yd = %.2f   zd = %.2f      rx= %d:%d ry= %d:%d rz= %d:%d",
(xd-x),(yd-y), (zd-z),rx,rcntx,ry,rcnty,rz,rcntz);
    printf(
"\n  x = %.2f   y  = %.2f   z = %.2f      grid = %.2f"
, x, y, z, speed);
    key = getche();                       /* READS KEYBOARD CHAR */
} /* while loop*/
}
/*****/ /*****/
setscrn()
{
                                        /* DRAWS CURSOR OR CURSORS TO SCREEN */
extern int wdown;
    if(wdown==0)
        cursor();
    if(wdown==1)
        windows();
}
/*****/

```



```

/*****/
keyboard(key)
int key;
{
    /* MAIN KEYBOARD FUNCTION */
    extern double speed;

    switch(key)
    {

        case F2: extract();
                break;
        case ASTRIX: reset();
                break;
        case KS : clears(0);
                cleari(0);
                linfun(0);
                setcl2();
                displa(0);
                twvis(1);
                viewrot();
                reset();
                break;
                /* TEST CASE */
        case F1 : bop++;
                subtract();
                break;

        case OS : morgset();
                /* BRANCH TO ORIGIN SET */
                break;
        case MINUS : speed -= 1.0;
                /* DECREASE GRID */
                break;
        case PLUS : speed += 1.0;
                /* INCREASE GRID */
                break;
        /* case CU : ccleari(bcol);
                CLEARS SCREEN
                break; */
        case FOUR :
        case LARR : left();
                /* CURSOR LEFT */
                break;
        case SIX :
        case RARR : right();
                /* CURSOR RIGHT */
                break;
        case RIGHT :
        case UP : up();
                /* CURSOR UP */
                break;
        case TWO :
        case DOWN : down();
                /* CURSOR DOWN */
                break;
        case SEVEN :
        case HOME : upleft();
                /* CURSOR UPLEFT */
                break;
        case NINE :
        case PGUP : upright();
                /* CURSOR UPRIGHT */
                break;
    }
}

```

```

case THREE :
case PGDN : downright(); /* CURSOR DOWN RIGHT */
            break;
case ONE :
case END : downleft(); /* CURSOR DOWN LEFT */
            break;
case ZERO :
case INS : forward(); /* CURSOR FORWARD */
            break;
case PERIOD :
case DEL : back(); /* CURSOR BACK */
            break;
default : ;
} /* switch */

```

```
keyboard2(key);
```

```
}
```

```
/******
```

```
keyboard2(key)
```

```
char key; /* KEYBOARD EXTENSION */
```

```
{
```

```
extern int wdw,randisk;
```

```
extern double zoomval,winx1, winx2, winy1, winy2;
```

```
extern int vx1, vx2, vy1, vy2;
```

```
extern double x,y,z;
```

```
FILE *cl;
```

```
switch(key)
```

```
{
```

```
case RS : remobj(); /* REMOVES OBJECTS */
```

```
printf("remove test\n");
```

```
break;
```

```
/* case VU : randiskf(); SET RANDISK
```

```
break; */
```

```
case VS : maindisp(); /* VIEW MENU */
```

```
viewm();
```

```
break;
```

```
case XS : cvwx(); /* RESET X AXIS VIEW */
```

```
break;
```

```
case YS : cvwy(); /* RESET Y AXIS VIEW */
```

```
break;
```

```
case ZS : cvwz(); /* RESET Z AXIS VIEW */
```

```
break;
```

```
case WS : maindisp(); /* DISPLAY WINDOW MENU */
```

```
windowm();
```

```
break;
```

```
case RS : /* CLEAR FILE */
```

```
cclearf();
```

```
default : ;
```

```
} /* switch */
```

```
keyboard3(key);
```

```
}
```

```
/******
```

```

/*****/
keyboard3(key)
int key;
{
    extern int twx0,twx1,twy0, twy1,twe0,twel;
    switch(key)
    {
/*      case PERCENT : readflist();      READ FILELIST
        break;      */
        case US: displa(1);                /* DISPLAY CGA */
            break;
        case UU: displa(0);                /* DISPLAY GRAPHICS SCREEN */
            break;
        case TS: twpos (0,639,1,479,0,0);  /* CLEAR TEXT WINDOW */
            twvis (0);
            break;
        case TU : twpos (twx0,twx1,twy0, twy1,twe0,twel); /* OPEN TEXT WINDOW */
            twvis (1);
            break;
        case FS : maindisp();              /* DISPLAY FILE MENU */
            filem();
            break;
        case CR : cprim();                 /* DRAW PRIMITIVE */
            break;
/*      case MS: pgdi(1);      SET CGA
        break;      */
        default : ;
    } /* switch */
    keyboard4(key);
}

/*****/

```

```

/*****
keyboard4(key)
int key;
{
    switch(key)
    {

        case CS: if(wdow==1)                                /* SETS COLOR */
                    windows();
                    if(wdow==0)
                        cursor();
                    clrun(251);
                    ccolor();
                    clrun(251);
                    setscrn();
                    break;
        case SS: if(wdow==1)                                /* SETS SOLID */
                    windows();
                    if(wdow==0)
                        cursor();
                    clrun(251);
                    primitive();
                    clrun(251);
                    setscrn();

                    break;
        case MS: if(wdow==1)                                /* SETS MODELING */
                    windows();
                    if(wdow==0)
                        cursor();
                    clrun(251);
                    modeling();
                    clrun(251);
                    setscreen();
                    break;                                /* SETS DIMENSIONS */
        case DS:
                    cmodel();
                    break;
        /* case EL:
                    break; */
        default : ;
    } /* switch */
}

*****/

```

```

/*****

```

```

mainvar()
{
    /* SETS VALUES TO GLOBAL VARIABLES */
    extern int menur,vwx,vwy,vwz,cli,cli,twx0,twx1,twy0,twy1,twe0,twe1 ;
    extern int twv,vwx1,vwx2,vwy1,vwy2,proj,lten1,lten2;
    extern int ltum,ltmax,col,wrow;
    extern int orgv,clcount,planon,tess,rxt,ryt,rzt,shad;
    extern int vwxlp, vwx2p, vwy1p, vwy2p;
    extern int vwxle, vwx2e, vwy1e, vwy2e;
    extern int vwxlv, vwx2v, vwy1v, vwy2v;
    extern int rcntx, rcnty, rcntz;
    extern int exton,ssnum,vectotal, firstcall, randisk,cliston;
    extern int sscount,ssnum,total,ssct,vtot,returnvali,bop,bolcount;
    bolcount=0;
    bop=0;
    faceton=1;
    cliston=0;
    returnvali=0;
    total=0;
    menur=0;
    orgv=0;
    ssct=0;
    vtot=0;
    sscount=0;
    firstcall=0;
    exton=1;
    ssnum=0;
    vectotal=0;
    randisk=0;
    rcntx=0;
    rcntx= 0;
    rcnty= 0;
    rcntz= 0;
    shad = 1;
    vwxlp = 0;
    vwx2p = 213;
    vwy1p = 246;
    vwy2p = 459;
    vwxle = 0;
    vwx2e = 213;
    vwy1e = 33;
    vwy2e = 246;
    vwxlv = 213;
    vwx2v = 639;
    vwy1v = 33;
    vwy2v = 459;
    rxt=0;
    ryt=0;

```

```

rzt=0;
tess=4;
planon=0;
kurs=15;
clcount=0;
wdow=0;
vwx=0;
vwy=0;
vwx2=0;
col=0;
cli=0;
twx0=0;
twx1=639;
twy0=0;
twy1=33;
twe0= 0;
twe1= 23;
twv= 1;
vwx1=27;
vwx2=613;
vwy1=35;
vwy2=460;
proj= 50;
lten1=0;
lten2=1;
ltsum= 0;
ltmax= 31;
col=220;
clbeg(241);                                /* SETS VIEW WINDOW FOR PLAN */
    project(0);
    viewport(vwx1p, vwx2p, vwy1p, vwy2p);
clend();
clbeg(242);                                /* SETS VIEW WINDOW FOR ELEVATION */
    project(0);
    viewport(vwx1e, vwx2e, vwy1e, vwy2e);
clend();
clbeg(243);                                /* SETS VIEW PORT FOR SHADED MODEL */
    project(50);
    viewport(vwx1v, vwx2v, vwy1v, vwy2v);
clend();

    floatvar();
}
/*****/

```

```

floatvar()
{
    /* SETS VALUES TO FLOATING POINT GLOBAL VARIABLES */
    extern double orgxd,orgyd,orgzd;
    extern double x,y,z,speed,dist,amb,ltx,lty,ltz;
    extern double height,width,length,depth1,depth2,depth3;
    extern double winx1,winx2,winy1,winy2,dish,disy;
    extern double orgx, orgy, orgz, rotnum,sx,sy,sz;
    extern double orgxt, orgyt, orgzt,zoomval,returnval;
    extern double vwptx,vwpty, vwptz;
    returnval=0.0;
    zoomval=1.0;
    orgxt=10.0;
    orgyt=10.0;
    orgzt=10.0;
    orgxd=0.0;
    orgyd=0.0;
    orgzd=0.0;
    sx=1.0;
    sy=1.0;
    sz=1.0;
    rotnum=0.0;
    depth1=0.0;
    depth2=0.0;
    depth3=0.0;
    x=0.0;
    y=0.0;
    z=0.0;
    speed=20.0;
    orgx=0.0;
    orgy=0.0;
    orgz=0.0;
    dist=600.0;
    disy=2500.0;
    dish=-450.0;
    amb=0.25;
    ltx= -140.0;
    lty= -101.0;
    ltz= -41.0;
    vwptx =0.0;
    vwpty =0.0;
    vwptz =0.0;
    winx1= -284.0;
    winx2= 284.0;
    winy1= -213.0;
    winy2= 213.0;
    height= 10.0;
    width= 10.0;
    length=50.0;
    setupvars();
}
/* *****/

```

```

setupvars()
{
    /* SETS UP GRAPHIC SCREEN */
    extern int planon,wdow,vwx,vwy,vwz,cl,cli,twx0,twx1,twy0,twy1,twe0,twel ;
    extern int twv, vwx1, vwx2, vwy1, vwy2, proj, lten1, lten2;
    extern int shade,planon,ltnum, ltmax, col,tess;
    extern double amb, x,y,z, speed,dist, amb, ltx;
    extern double lty,ltz,height, width, length;
    extern double winx1,winx2,winyl,winyl,dish,disy,vwptx,vwpty,vwptz;
    ca(); /* COMMANDS IN ASCII */
    clbeg(249); /* BEGINS COMMAND LIST 249 */
    if(wdow ==0)
    {
        clears(0); /* CLEARS SCREEN */
        twpos(twx0,twx1,twy0, twy1,twe0,twel); /* SETS TEXT WINDOW SIZE */
        twvis(twv); /* ACTIVATES TEXT WINDOW */
    }
    if(planon==0)
    {
        cleari(0); /* CLEARS INTENSITY BUFFER */
        twpos(twx0,twx1,twy0, twy1,twe0,twel); /* SETS TEXT WINDOW SIZE */
        twvis(twv); /* ACTIVATES TEXT WINDOW */
        displa(0); /* ACTIVATES GRAPHICS SCREEN */
        viewport(vwx1, vwx2, vwy1, vwy2); /* SETS VIEW PORT SIZE */
        window(winx1, winx2, winyl, winyl); /* SETS WINDOW SIZE */
    }
    if(planon==0)
    {
        project(proj); /* SETS PROJECTION ANGLE */
        distan(dist); /* SETS DISTANCE */
        disth(dish); /* SETS FRONT CLIPPING PLANE */
        disty(disy); /* SET BACK CLIPPING PLANE */
        cliph(1); /* ACTIVATES FRONT CLIPPING PLANE */
        clipy(1); /* ACTIVATES BACK CLIPPING PLANE */
        light(0, amb, 31, ltx, lty, ltz); /* SET LIGHTING DIRECTION */
        liten(0, 1); /* ACTIVATES LIGHT SOURCE */
        vwprt(vwptx,vwpty,vwptz); /* SET VIEWING REFERENCE POINT */
        vwiden(); /* RESETS VIEWING MATRIX */
    }
    if(planon==0)
    {
        vwrotx(vwx); /* SETS X AXIS VIEWING ANGLE */
        vwroty(vwy); /* SETS Y AXIS VIEWING ANGLE */
        vwrotz(vwz);
    }
    lutint(6); /* CALLS LOOK UP TABLE */
    color(col); /* DEFINES COLOR */
    mdiden(); /* RESETS MODELING MATRIX */
    clend(); /* ENDS CL */
    dispat(4, tess); /* SETS FACETING DEGREE */
    if(shade==1)
        dispat(0,1);
    if(shade==0)
        dispat(2, 0);
    clrun(249); /* RUNS CL */
}
*****/

```



```

windows()
{
    /* DRAWS CURSORS TO MULTI WINDOWS */
    extern int rx,ry,rz,vwy,vwx,vwz,planon;
    extern double winx1, winx2, winy1, winy2;
    window(winx1, winx2, winy1, winy2);

    vwidth(); /* RESETS VIEW WINDOW */
    clrwin(241); /* RUNS CL 241 - SETS PLAN WINDOW */
    vwidth(vwy);
    vwidth(270); /* ACTIVATES PLAN VIEW */
    vwidth(0);
    planon=1; /* DRAWS CURSOR TO PLAN VIEW */
    cursor();
    clrwin(242); /* RUNS CL 242 - SETS ELEVATION WINDOW */
    vwidth(); /* RESETS VIEW PORT */
    vwidth(vwy); /* ROTATES Y AXIS VIEW */
    vwidth(0);
    vwidth(0);
    planon=2;
    if((vwy ==0) && (rx==0) && (ry==0)) /* TESTS FOR ROTATION */
        cursor();
    if((vwy !=0) || (rx!=0) || (ry!=0) ) /* DRAWS ELEVATION CURSOR */
        cursor();
    clrwin(243);
    vwidth(); /* RESETS VIEWPORT */
    vwidth(vwx); /* ROTATES VIEWPORT */
    vwidth(vwy);
    vwidth(vwz);
    planon=0;

    cursor();
}
/*****/
ccleari(bcol)
int bcol; /* CLEARS INTENSITY BUFFER TO GIVEN COLOR */
{
    printf("\ngive backgroundcolor: ");
    scanf("%d",&bcol);
    if ((bcol<0) || (bcol>255))
        bcol=0;
    ccleari(bcol);
    printf("\n backgroundcolor: %d ",bcol);

    ccleari(bcol);
    maindisp();
    setscrn();
}
/*****/

```

```

randiskf()
{
    /* STORES RAM DISK FILE ON DISK */
    extern int randisk;
    FILE *cl;
    FILE *f1;
    FILE *f2;
    static char input 1;
    randisk =1;
    setupf();
    cl = fopen("d:remove.i", "w");
    fclose(cl);
    printf("\n RAM DISK IN OPERATION");
    f2 = fopen("command.d", "r");
    f1 = fopen("d:command.d", "w");
    while (fgets(input,LIM,f2)!=NULL)
        fprintf(f1,"%s",input);
    fclose(f1);
    fclose(f2);
}

/*****/
cmodel()
{
    /* DIMENSIONING FUNCTION */
    extern double speed, width, height, lenght;
    extern int wdow;
    int ch;
    ch = getche();
    while(ch !=SPACE)
    {
        /* CASE FOR DIMENSION CHANGES */
        switch(ch)
        {
            case MINUS : speed -= 1.0;    /* DECREASE GRID */
                        break;
            case PLUS : speed += 1.0;    /* INCREASE GRID */
                        break;
            case FOUR :
            case LARR :
                        mcase4();
                        break;
            case SIX :
            case RARR :
                        mcase6();
                        break;
            case EIGHT:
            case UP : mcase8();
                     break;
            case TWO :
            case DOWN: mcase2();
                     break;
        }
    }
}

```

```

        case ZERO :
        case INS  : mcase0();
                    break;
        case PERIOD:
        case DEL  : mcaseb();
                    break;
        default  : ;
    }
    printf("\n width = %.2f      height = %.2f      length = %.2f      grid = %.2f",width,height,length,speed);
        ch = getch();
    }
}
/*****/
mcase4()                                /* DECREASE WIDTH */
{
    extern double speed,width;
    extern int wdown;
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
    width=width-speed;
    width=abs(width);
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
}
/*****/
mcase6()                                /* INCREASE WIDTH */
{
    extern double speed,width;
    extern int wdown;
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
    width=width+speed;
    width=abs(width);
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
}
/*****/

```

```

mcase8()
{
    extern double height;
    extern int wdown;
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
    height=height+speed;
    height=abs(height);
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
}
/*****/
mcase2()
{
    extern double speed,height;
    extern int wdown;
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
    height=height-speed;
    height=abs(height);
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
}
/*****/
mcase0()
{
    extern double speed,length;
    extern int wdown;
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
    length=length+speed;
    length=abs(length);
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
}
/*****/

```

```

mcaseb()
{
    extern double speed,length;
    extern int wdown;
    if (wdown==1)
        windows();
    if(wdown==0)
        cursor();
    length=length-speed;
    length=abs(length);
    if(wdown==1)
        windows();
    if(wdown==0)
        cursor();
}
/*****/
abs(x)
double x;
{
    double y;
    y=(x<0)?-x:x;
    return y;
}
/*****/
mcenter()
{
    extern int curs,wdown;
    extern double orgx,orgy,orgz,orgxd,orgyd,orgzd;
    extern double sx,sy,sz,x,y,z,xf,yf,zf;
    extern int randisk,rx,rz,ry,rcntx,rcnty,rcntz;
    FILE *cl;

    mriden();
    curs=15;
    if (randisk==0)
        cl = fopen("command.d", "a");
    if (randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl, "\nmriden\n");
    fclose(cl);
    sx=1.0;
    sy=1.0;
    sz=1.0;
    x=0.0;
    y=0.0;
    z=0.0;
}

```

/* DECREASE LENGHT */

/* RETURNS ABSOLUTE VALUE */

/* RESETS MODELING MATRIX AND CENTERS CURSOR */

/* RESETS MODELING MATRIX */

/* OPENS COMMAND LIST FILE */

/* CLOSES FILE */

/* RESETS SCALING VALUES */

/* SETS CURSOR TO ORIGIN */

```

xf=0.0;
yf=0.0;
zf=0.0;
rx=0;
ry=0;
rz=0;
rcntx=0;
rcnty=0;
rcntz=0;
orgx=0.0;
orgy=0.0;
orgz=0.0;
orgxd=0.0;
orgyd=0.0;
orgzd=0.0;
setupvars();
setupf();

}
/*****/
cclears()
{
extern int bcol;
cursor();
clears (bcol);
cursor();
}
/*****/
cvrpt()
{
extern double x,y,z;
extern double vwptx,vwpty, vwptz;

vwptx=x;
vwpty=y;
vwptz=z;
setupf();
getfile();
}
/* ROTATION VALUES ARE SET TO ZERO */
/* ROTATION ORDERS ARE RESET */
/* MODELING ORIGIN IS RESET */
/* CLEARS SCREEN */
/* SET CURRENT XYZ LOCTION TO VIEW REFERENCE POINT */

```

```

/*****/
cclearf()
{
extern int wdw,curs;
extern double sx,sy,sz,x,y,z,xf,yf,zf;
extern int randisk,rx, ry, rz,rcntx,rcnty ,rcntz;
extern double winx1,winy1,winx2, winy2;
extern int vx1p, vx2p, vwy1p, vwy2p;
extern int vx1e, vx2e, vwy1e, vwy2e;
extern int vx1v, vx2v, vwy1v, vwy2v;
extern int vx1, vx2, vwy1, vwy2;
extern double orgxd, orgx,orgyd;
extern double orgy, orgyd, orgy;
extern int vx,vwy, vnz;

FILE *cl;
if (randisk==0) /* OPENS FILE */
    cl = fopen("command.d", "w"); /* CLEARS FILE */
if (randisk==1)
    cl = fopen("d:command.d", "w");

fprintf(cl,"\n");
fclose(cl); /* CLOSSES FILE */

vx=0;
vwy=0;
vnz=0;
vwidn();
sx=1.0; /* MODELING VALUES ARE RESET */
sy=1.0;
sz=1.0;

rx=0;
ry=0;
rz=0;
rcntx=0;
rcnty=0;
rcntz=0;
x=0.0;
y=0.0;
z=0.0;
xf=0.0;
yf=0.0;
zf=0.0;
orgxd=0.0;
orgx=0.0;
orgyd=0.0;
orgy=0.0;
orgyd=0.0;
orgy=0.0;

```

```

mdiden();
clears(0);
cleari(0);
  maindisp();
  setscrn();

/* BORDERS ARE REDRAWN */
if(wdow==0)
{
  linfun(0);
  color(250);
  move(winxl,winyl);
  rect(winx2, winy2);
}
if(wdow==1)
{
/* BORDERS ARE REDRAWN TO MULTIPLE WINDOWS */
  viewport(vwxlp, vwx2p, vwy1p, vwy2p);
  linfun(0);
  color(250);
  move(winxl,winyl);
  rect(winx2, winy2);
  viewport(vwxle, vwx2e, vwy1e, vwy2e);
  linfun(0);
  color(250);
  move(winxl,winyl);
  rect(winx2, winy2);
  viewport(vwxlv, vwx2v, vwy1v, vwy2v);
  linfun(0);
  color(250);
  move(winxl,winyl);
  rect(winx2, winy2);
}
}
/*****/
/*****/
/* CLEARS CURRENT COMMAND LIST FILE */
reset()
{
extern int wdow,curs;
extern double sx,sy,sz,x,y,z,xf,yf,zf;
extern int randisk,rx, ry, rz,rcntx,rcnty ,rcntz;
extern double winxl,winyl,wiax2, winy2;
extern int vwxlp, vwx2p, vwy1p, vwy2p;
extern int vwxle, vwx2e, vwy1e, vwy2e;
extern int vwxlv, vwx2v, vwy1v, vwy2v;
extern int vwxl, vwx2, vwy1, vwy2;
extern double orgxd, orgx,orgyd;
extern double orgy, orgyd, orgy;
extern int vwx,vwy, vwz;

```



```

FILE *cl;
setscrn();
if (randisk==0)                               /* OPENS FILE */
    cl = fopen("command.d", "a");             /* CLEARS FILE */
if (randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl, "\nmdiden");
fclose(cl);                                    /* CLOSES FILE */
if((vwx!=0)|| (vwy!=0)|| (vwz!=0))            /* TEST IF VIEW WINDOW ACTIVE */
{
    vwiden();
    vwx=0;
    vwy=0;
    vwz=0;
    getfile();
}
sx=1.0;                                       /* MODELING VALUES ARE RESET */
sy=1.0;
sz=1.0;
rx=0;
ry=0;
rz=0;
rentx=0;
renty=0;
rentz=0;
x=0.0;
y=0.0;
z=0.0;
xf=0.0;
yf=0.0;
zf=0.0;
orgxd=0.0;
orgx=0.0;
orgyd=0.0;
orgy=0.0;
orgyd=0.0;
orgy=0.0;
mdiden();
if(wdow==0)
{
    clears(0);
    dispi(0);
}
if(wdow==1)
{
    dispi(0);
}
maindisp();                                  /* DISPLAYS MAIN MENU */
setscrn();
}
/*****/

```

```

/*****/
morgset()
{
  extern double x,y,z,xf,yf,zf,orgx,orgy,orgz;
  extern int rz,rx,ry,curs,orgv,cursnum;
  extern double orgxd,orgyd,orgzd;
  extern double orgx,orgy,orgz;
  int ch;

  orgv=1;
  setscrn();
  if(rz !=0; rx!=0; ry!=0)
  {
    mdiden();
    rz=0;
    ry=0;
    rx=0;
    orgxd=0.0;
    orgyd=0.0;
    orgzd=0.0;
    orgy=y;
    orgx=x;
    orgz=z;
    mdtran(x,y,z);
  }
  orgxt=x;
  orgyt=y;
  orgzt=z;
  setscrn();
  cursnum=curs;
  curs=15;
  ch=getche();
  VMPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  while(ch !=SPACE)
  {
    switch(ch)
    {
      case MINUS : speed -= 1.0; /* DECREASE GRID */
                  break;
      case PLUS : speed += 1.0; /* INCREASE GRID */
                  break;

      case FOUR :
      case LARR : left();
                  break;
      case SIX :
      case RARR : right();
                  break;
    }
  }
}
/* MODELING ORIGIN IS SET */

```

```

    case RIGHT :
    case UP : up();
               break;
    case TWO :
    case DOWN : down();
               break;
    case ZERO :
    case INS : forward();
               break;
    case NINE :
    case PGUP : upright();
               break;
    case SEVEN :
    case HOME : upleft();
               break;
    case ONE :
    case END : downleft();
               break;
    case THREE :
    case PGDN : downright();
               break;
    case PERIOD :
    case DEL : back();
               break;
    default: ;
  }

  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  ch=getche();
}
orgv=0;
orgx=x+orgxd;
orgy=y+orgyd;
orgz=z+orgzd;
curs=cursnum;
}
/*****/

```

```
/* THIS SET OF FUNCTIONS DRAWS A SOLID TO THE SCREEN */
```

```
#include "std.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
#define LIM 40
/* ***** */
cprim()
{
extern int vx1p, vx2p, vy1p, vy2p;
extern int vx1e, vx2e, vye, vye2e;
extern int vx1v, vx2v, vy1v, vy2v;
extern int vx1, vx2, vy1, vy2;
extern int curs, vx, vy, vz, planon, shad, tess;
extern int rz, ry, rx, morg, displ, disp2, wdw, col;
extern double winx1, winy1, winx2, winy2;
extern double x, y, z, xf, yf, zf;
extern double height, width, length;
extern double orgx, orgy, orgz;
extern double xd, yd, zd;
FILE *cl;
xd=x; /* STORES POINT LOCATION */
yd=y;
zd=z;
twvis(0); /* TURNS OFF TEXT WINDOW */
if(wdw==1) /* TESTS FOR MULTIPLE WINDOWS */
{
windows(); /* ERASES CURSOR */
linfun(0); /* PREPARES FOR PLAN DRAWING */
wirefr(2, 0);
vwide();
viewport(vx1p, vx2p, vy1p, vy2p);
project(0);
dispat(4, tess);
vrotz(vy);
vrotx(270);
cprimf(); /* DRAWS SOLID TO SCREEN */
vwide();
viewport(vx1e, vx2e, vye, vye2e); /* SET UP FOR ELEVATION */
vrotz(vy);
vrotx(0);
dispat(4, tess);
cprimf(); /* DRAWS SOLID TO SCREEN */
viewport(vx1v, vx2v, vy1v, vy2v); /* SETS UP PERSPECTIVE VIEW */
project(50);
dispat(4, tess);
```

```

dispat(0,shad);
cprimf();
dispi(0);
color(255);
color(col);
}
if(wdow==0)
{
    viewport(vwx1, vwx2, vwy1, vwy2);
    cursor();
    if(planon==1 || planon==2)
    {
        dispat(4,tess);
        cprimf();
        linfun(0);
        wirefr(2, 0);
        project(0);
    }
    if(planon==0)
    {
        project(50);
        dispat(0,shad);
        cprimf();
        dispi(0);
    }
}
cprim2();
if(planon==0)
{
    dispi(0);
}
if(wdow==0)
{
    cursor();
    viewport(vwx1, vwx2, vwy1, vwy2);
    move(winx1, winy1);
    rect(winx2, winy2);
    color(col);
}

if(wdow==1)
{
    windows();
    color(250);
    viewport(vwx1e, vwx2e, vwy1e, vwy2e);
    move(winx1, winy1);

```

/* TURNS ON SHADING */

/* SETS VIEWPORT */

/* TESTS FOR PLAN OR ELEVATION */

/* DRAWS IN WIREFRAME */

/* TESTS FOR PERSPECTIVE */

/* DRAWS IN SHADED MODE */

/* BRANCHES TO cprim2 FUNCTION */

/* cprim2 FILES SOLID TO COMMAND LIST WITH ATTRIBUTES

/* REDRAWS SCREEN BORDER */

```

rect(winx2, winy2);
viewport(vwx1p, vwx2p, vwy1p, vwy2p);
move(winx1, winy1);
rect(winx2, winy2);
viewport(vwx1v, vwx2v, vwy1v, vwy2v);
project(0);
linfun(0);
wirefr(2, 0);
move(winx1, winy1);
rect(winx2, winy2);

```

```

}
twvis(1);
}

```

```

/*****

```

```

cprim2()                                /* STORES SOLID COMMANDS TO COMMAND LIST */
{

```

```

extern int rz,ry,rx,morg,displ, disp2,wdow,col,bop;
extern double x,y,z,xf,yf,zf;
extern double height,width,length;
double xt,yt,zt,x2,y2,z2;
extern double sx,sy,sz,orgx, orgy, orgz;
extern int rxd,ryd,rzd,firstcall,curs,ramdisk;
FILE *cl;
extern int bolcount;

```

```

x2=x;                                /* CALCULATES DISTANCE FROM LAST POINT */
y2=y;
z2=z;

```

```

xt=x2-xf;
yt=y2-yf;
zt=z2-zf;
xf=x2;
yf=y2;
zf=z2;

```

```

rxd=360-rx;                            /* CALCULATES DIFFERENCE IN ROTATION TO MAKE A FULL
CIRCLE */

```

```

ryd=360-ry;
rzd=360-rz;
if(bop!=0)

```

```

{
    cl = fopen("sub.i","a");
    boolean(cl);
}

```

```

if((ramdisk==0)&&(bop==0))            /* TEST FOR RAM DISK ON */

```

```

    cl = fopen("command.d", "a");    /* OPEN APPROPRIATE FILE */

```

```

if((ramdisk==1)&&(bop==0))
    cl = fopen("d:command.d", "a");

```

```

if(bop==0)
{
if(firstcall !=0 || curs !=8)                                /* TEST FOR FIRST ENTRY */
    fprintf(cl,"locate %.1f, %.1f, %.1f\n",x,y,z);           /* WRITES X Y Z COORDINATE OF SOLID */
    fprintf(cl,"mdtran %.1f, %.1f, %.1f\n",xt,yt,zt);        /* WRITES DISTANCE FROM LAST POINT */
}
if(sx!=1.0)                                                    /* TEST FOR ACTIVE SCALE FUNCTION */
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);      /* WRITES MODELING ORIGIN TO FILE */
    fprintf(cl,"mdscal %.2f,%.2f,%.2f\n",sx,sx,sx);           /* WRITES SCALE VALUE OF SOLID */
}
if((rx != 0) && ( ry==0) && (rz==0))                          /* TEST FOR X ROTATION VALUE */
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);      /* PRINTS VALUE TO COMMAND LIST */
    fprintf(cl,"mdrotx %d\n",rx);
}
if((ry != 0) && ( rx==0) && (rz==0))                          /* TEST FOR Y ROTATION VALUE */
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz );
    fprintf(cl,"mdroty %d\n",ry);                             /* PRINTS VALUE TO COMMAND LIST */
}
if((rz != 0) && ( rx==0) && (ry==0))                          /* TEST FOR Z ROTATION VALUE */
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz );    /* PRINTS VALUE TO COMMAND LIST */
    fprintf(cl,"mdrotz %d\n",rz);
}
frcntxa(cl);                                                    /* TEST FOR ORDER OF EXECUTION */
frcntya(cl);
frcntza(cl);
fileprim(cl);                                                    /* FILES SOLID TO COMMAND LIST */

if((rx != 0) && ( ry==0) && (rz==0))                          /* THESE FUNCTIONS WRITE TO THE COMMAND LI
    ST */
{
    TURNED BACK */
    fprintf(cl,"mdrotx %d\n",rxd);
}
if((ry != 0) && ( rx==0) && (rz==0))
{
    fprintf(cl,"mdroty %d\n",ryd);
}

if((rz != 0) && ( rx==0) && (ry==0))
{
    fprintf(cl,"mdrotz %d\n",rzd);
}

```

```

frcntxb(cl);
frcntyb(cl);
frcntzb(cl);
if(sx!=1.0)
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
    TO ONE */
    fprintf(cl,"mdscal %.2f,%.2f,%.2f\n",(1.0/sx),(1.0/sx),(1.0/sx));
}

```

```

fclose(cl);
}

```

```

/*****

```

```

clean(cl)

```

```

FILE *cl;

```

```

{

```

```

    extern int col,bolcount;

```

```

    extern double x,y,z;

```

```

    if(bolcount==0)
    {

```

```

        color(32);

```

```

        fprintf(cl,"\nsolbop 1");
    }

```

```

    fprintf(cl,"\nmdiden");

```

```

    fprintf(cl,"\nmdtran %.1f, %.1f, %.1f\n",x,y,z);
    /* WRITES X Y Z COORDINATE OF SOLID */

```

```

    bolcount++;
}

```

```

/*****

```

```

frcntxa(cl)

```

```

/* CHECKS EXECUTION ORDER */

```

```

FILE *cl;

```

```

{

```

```

    extern int rcntx,rcnty,rcntz;

```

```

    extern int rz,ry,rx,morg;

```

```

    extern double x,y,z,xf,yf,zf;

```

```

    extern double orgx, orgy, orgz;

```

```

    if (rcntx==1)
    {

```

```

        if((rcnty==2)&&(rcntz==0) )
        {

```

```

            fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);

```

```

            fprintf(cl,"mdrotx %d\n",rx);

```

```

            fprintf(cl,"mdroty %d\n",ry);
        }
    }

```



```

if((rcnty==2)&&(rcntz==3) )
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
    fprintf(cl,"mdrotx %d\n",rx);
    fprintf(cl,"mdroty %d\n",ry);
    fprintf(cl,"mdrotz %d\n",rz);
}
if((rcntz==2)&&(rcnty==0) )
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
    fprintf(cl,"mdrotx %d\n",rx);
    fprintf(cl,"mdrotz %d\n",rz);
}
if((rcntz==2)&&(rcnty==3) )
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
    fprintf(cl,"mdrotx %d\n",rx);
    fprintf(cl,"mdrotz %d\n",rz);
    fprintf(cl,"mdroty %d\n",ry);
}
}
}

```

```

/*****

```

```

    frcntxb(cl)

```

```

    /* ROTATES BACK EXECUTION ORDER FOR X*/

```

```

    FILE *cl;

```

```

{
    extern int rcntx,rcnty,rcntz;
    extern int rzd,ryd,rxz,mozg;
    extern double x,y,z;
    extern double orgx, orgy, orgz;
    if (rcntx==1)
    {
        if((rcnty==2)&&(rcntz==0) )
        {
            fprintf(cl,"mdroty %d\n",ryd);
            fprintf(cl,"mdrotx %d\n",rxz);
        }
        if((rcnty==2)&&(rcntz==3) )
        {
            fprintf(cl,"mdrotz %d\n",rzd);
            fprintf(cl,"mdroty %d\n",ryd);
            fprintf(cl,"mdrotx %d\n",rxz);
        }
    }
}

```

```

    if((rcntz==2)&&(rcnty==0)        )
    {
        fprintf(cl,"mdrotz %d\n",rzd);
        fprintf(cl,"mdrotx %d\n",rxd);
    }
    if((rcntz==2)&&(rcnty==3)        )
    {
        fprintf(cl,"mdroty %d\n",ryd);
        fprintf(cl,"mdrotz %d\n",rzd);
        fprintf(cl,"mdrotx %d\n",rxd);
    }
}
}
/*****/
frcntya(cl)                                /* ROTATES BACK EXECUTION ORDER FOR Y*/
FILE *cl;
{
extern int rcntx,rcnty,rcntz,rz,ry,rx,morg;
extern double x,y,z, orgx, orgy, orgz;
if (rcnty==1)
{
    if((rcntx==2)&&(rcntz==0)        )
    {
        fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
        fprintf(cl,"mdroty %d\n",ry);
        fprintf(cl,"mdrotx %d\n",rx);
    }
    if((rcntx==2)&&(rcntz==3)        )
    {
        fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
        fprintf(cl,"mdroty %d\n",ry);
        fprintf(cl,"mdrotx %d\n",rx);
        fprintf(cl,"mdrotz %d\n",rz);
    }
    if((rcntz==2)&&(rcntx==0)        )
    {
        fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
        fprintf(cl,"mdroty %d\n",ry);
        fprintf(cl,"mdrotz %d\n",rz);
    }
    if((rcntz==2)&&(rcntx==3)        )
    {
        fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz);
        fprintf(cl,"mdroty %d\n",ry);
        fprintf(cl,"mdrotz %d\n",rz);
        fprintf(cl,"mdrotx %d\n",rx);
    }
}
}
/*****/

```

```

frcntyb(cl)
FILE *cl;
{
extern int rcntx,rcnty,rcntz, rxd,ryd,rzd;
if (rcnty==1)
{
if((rcntx==2)&&(rcntz==0))
{
fprintf(cl,"mdrotx %d\n",rxd);
fprintf(cl,"mdroty %d\n",ryd);
}
if((rcntx==2)&&(rcntz==3))
{
fprintf(cl,"mdrotx %d\n",rzd);
fprintf(cl,"mdrotx %d\n",rxd);
fprintf(cl,"mdroty %d\n",ryd);
}
if((rcntz==2)&&(rcntx==0))
{
fprintf(cl,"mdrotx %d\n",rzd);
fprintf(cl,"mdroty %d\n",ryd);
}
if((rcntz==2)&&(rcntx==3))
{
fprintf(cl,"mdrotx %d\n",rxd);
fprintf(cl,"mdrotx %d\n",rzd);
fprintf(cl,"mdroty %d\n",ryd);
}
}
}
/*****/
frcntza(cl)          /* ROTATES BACK EXECUTION ORDER FOR*/
FILE *cl;
{
extern int rcntx,rcnty,rcntz,rz,ry,rx;
extern double x,y,z,xf,yf,zf, orgx, orgy, orgz;
if (rcntz==1)
{
if((rcntx==2)&&(rcnty==0))
{
fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz );
fprintf(cl,"mdrotx %d\n",rz);
fprintf(cl,"mdrotx %d\n",rx);
}
}
}

```

```

if((rcntx==2)&&(rcnty==3))
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz );
    fprintf(cl,"mdrotz %d\n",rz);
    fprintf(cl,"mdrotx %d\n",rx);
    fprintf(cl,"mdroty %d\n",ry);
}
if((rcnty==2)&&(rcntx==0))
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz );
    fprintf(cl,"mdrotz %d\n",rz);
    fprintf(cl,"mdroty %d\n",ry);
}
if((rcnty==2)&&(rcntx==3))
{
    fprintf(cl,"mdorg %.1f,%.1f,%.1f\n",orgx,orgy,orgz );
    fprintf(cl,"mdrotz %d\n",rz);
    fprintf(cl,"mdroty %d\n",ry);
    fprintf(cl,"mdrotx %d\n",rx);
}
}
}

```

/*****/

```

frcntzb(cl)
FILE *cl;
{
    extern int rcntx,rcnty,rcntz;
    extern double orgx, orgy, orgz;
    extern int rxd,ryd,rzd;
    if (rcntz==1)
    {
        if((rcntx==2)&&(rcnty==0) )
        {
            fprintf(cl,"mdrotx %d\n",rxd);
            fprintf(cl,"mdrotz %d\n",rzd);
        }
        if((rcntx==2)&&(rcnty==3) )
        {
            fprintf(cl,"mdroty %d\n",ryd);
            fprintf(cl,"mdrotx %d\n",rxd);
            fprintf(cl,"mdrotz %d\n",rzd);
        }
        if((rcnty==2)&&(rcntx==0) )
        {
            fprintf(cl,"mdroty %d\n",ryd);
            fprintf(cl,"mdrotz %d\n",rzd);
        }
    }
}

```

```

if((rcnty==2)&&(rcntx==3)    )
{
    fprintf(cl,"mdrotx %d\n",rxd);
    fprintf(cl,"mdroty %d\n",ryd);
    fprintf(cl,"mdrotz %d\n",rzd);
}
}
}

```

```

/*****

```

```

fileprim(cl)

```

```

FILE *cl;

```

```

/* FILES SOLID TO COMMAND LIST */

```

```

{
FILE *f2;
extern double x,y,z;
extern double rotnum,height,width,length;
extern int firstcall, vectotal,curs;
static char input[10];

```

```

*   if(curs!=8)
*   fprintf(cl,"locate %.1f, %.1f, %.1f\n",x,y,z);
*/

```

```

if(curs==1)                                /* FILES BLOCK */
{
    fprintf(cl,"psolid  4, %.1f, %.1f,%.1f\n", width,height,length);
}

```

```

if(curs==2)                                /* FILES CYLINDER */
{
    fprintf(cl,"psolid  0, %.1f,%.1f\n", width/2.0,height);
}

```

```

if(curs==3)                                /* FILES TUBE */
{
    fprintf(cl,"psolid  6, %.1f, %.1f,%.1f\n", length/2.0,width/2.0, height);
}

```

```

if(curs==4)                                /* FILES SPHERE */
{
    fprintf(cl,"psolid  2, %.1f\n", height/2.0);
}

```

```

if(curs==5)                                /* FILES TORUS */
{
    fprintf(cl,"psolid  3, %.1f, %.1f\n", length/2.0, width/2.0);
}

```

```

if(curs==6)                                /* FILES CONE */
{
    fprintf(cl,"psolid  1, %.1f,%.1f,%.1f\n", width/2.0,length/2.0, height);
}

```

```

if(curs==7)                                /* FILES WEDGE */
{
    fprintf(cl,"psolid 5, %.1f, %.1f,%.1f\n\n",width,height,length);
}
if(curs==8)
{
    if(firstcall ==0)                        /* TEST FOR FIRST ENTRY */
    {
        fprintf(cl,"\nclbeg l\n");

        f2 = fopen("ssolid.d", "r");          /* WRITES SWEEP SOLID VERTICES TO COMMAND LIS
    T */
        while (fgets(input,LIN,f2)!=NULL)
            fprintf(cl,"%s",input);
        fclose(f2);
        firstcall++;
        fprintf(cl,
            "\nclend l\n");
    fprintf(cl,"locate %.1f, %.1f, %.1f\n".x.y.z);    /* WRITES X Y Z COORDINATE OF SOLID */

    }
    fprintf(cl,
        "clrun l\n");

}

}
/ *****/
cprintf()
{
    extern int col,bop,curs;
    extern double height,width,length;

    if(curs==1)
    {
        block(4,width, height, length);
    }
    if(curs==2)
    {
        column(0, width/2.0, height);
    }
    if(curs==3)
    {
        tube(6,length/2.0,width/2.0, height);
    }
}

```

```

if(curs==4)
{
    sphere(2,height/2.0);
}
if(curs==5)
{
    torus(3, length/2.0, width/2.0);
}
if(curs==6)
{
    cone(1,width/2.0,length/2.0, height);
}
if(curs==7)
{
    wedge(5,width, height,length);
}
if(curs==8)
{
    printf("\n EXECUTE CLCORS \n");
    clrscr(1);
}

}

/*****/
col2()
{
    extern double fvar,fvar1,fvar2,fvar3;
    column(0,fvar2, fvar3);
    printf("\ncolumn 1, %.1f, %.1f", fvar2,fvar3);
}

/*****/
sphere2()
{
    extern double fvar,fvar1,fvar2;
    sphere( 2, fvar2);
    printf("\nsphere 2 %.1f", fvar2);
}

/*****/

```

```
/**FUNCTIONS IN THIS SECTION DEAL WITH EXECUTION OF THE COMMANDS FROM THE DISK */
```

```
#include <fcntl.h>
```

```
#include <sys/types.h>
```

```
/* DEFINES EXTERNAL FILES */
```

```
#include <sys/stat.h>
```

```
#include <io.h>
```

```
#include <stdio.h>
```

```
#include "std.h"
```

```
#include <sys/types.h>
```

```
#include <io.h>
```

```
#include <string.h>
```

```
#include "commands.h"
```

```
char sign;
```

```
int ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6,count, clcount; /* EXTERNAL VARIABLES */
```

```
double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
```

```
double xcon, ycon, zcon;
```

```
/******
```

```
xyzsend()
```

```
/* STORES X Y Z LOCATION OF CURSOR */
```

```
{
```

```
extern double xcon, ycon, zcon, x, y, z;
```

```
xcon=x;
```

```
ycon=y;
```

```
zcon=z;
```

```
}
```

```
/******
```

```
xyzreturn()
```

```
/* RETURNS X Y Z CURSOR LOCATION */
```

```
{
```

```
extern double xcon, ycon, zcon, x, y, z;
```

```
x= xcon;
```

```
y= ycon;
```

```
z= zcon;
```

```
}
```

```
/******
```

```
/******
```

```
func1(input)
```

```
/* ACTIVATES COMMANDS IF MATCH OCCURS */
```

```
char input10;
```

```
{
```

```
distanf(input);
```

```
litenf(input);
```

```
lightf(input);
```

```
vwidenf(input);
```

```
vwrptf(input);
```

```
vwrptf(input);
```

```
lutintf(input);
```

```
colorf(input);
```

```
mdidenf(input);
```

```
mdtranf(input);
```

```
solidf(input);
```



```

    rotxf(input);
    scalf(input);
    rotyf(input);
    rotzf(input);
    orgf(input);
    vwrotyf(input);
    solbegf(input);
    solendf(input);
    solbopf(input);
    solrunf(input);
    clbegf(input);
    clendf(input);
    clrunf(input);
}
/*****/
func2(input)
char input[];
{
    caf(input);
    clearsfc(input);
    clearif(input);
    twposf(input);
    twvisf(input);
    displaf(input);
    vreportf(input);
    windowf(input);
    projectf(input);
    distanf(input);
    litenf(input);
    lightf(input);
    vwidenf(input);
    vwrptf(input);
    vwrptf(input);
    vwrotxf(input);
    vwrotyf(input);
    vwrotzf(input);
    lutintf(input);
    colorf(input);
    mdidenf(input);
    mdtranf(input);
    solidf(input);
    dispatf(input);
    rotxf(input);
    scalf(input);
    rotyf(input);
    rotzf(input);
    orgf(input);
    solbegf(input);

```

```

solendf(input);
solbopf(input);
solrunf(input);
clbegf(input);
clendf(input);
clrunf(input);
}

```

```

/*****/
solrunf(input)                /* ACTIVATES SOLRUN COMMMAD */
char input[];
{
extern int count,ivar1;
if (((strcmp(input,SOLRUN)== 0)) && (count< 1))
{
solrun(ivar1);
printf("\nSOLRUN %d",ivar1);
count++;
}
}
/*****/

```

```

solbegf(input)                /* ACTIVATES SOLBEG COMMMAD */
char input[];
{
extern int count,ivar1;
if (((strcmp(input,SOLBEG)== 0)) && (count< 1))
{
solbeg(ivar1);
printf("\nSOLBEG %d",ivar1);
count++;
}
}

```

```

/*****/
solendf(input)                /* ACTIVATES SOLEND COMMMAD */
char input[];
{
extern int count;
if (((strcmp(input,SOLEND)== 0)) && (count< 1))
{
solend();
printf("\nSOLEND ");
count++;
}
}

```

```

/*****/
solbopf(input)                                /* ACTIVATES SOLBOP COMMAND */
char input40;
{
extern int count,ivar1;
if (((strcmp(input,SOLBOP)== 0)) && (count< 1))
{
solbop(ivar1);
printf("\nSOLBOP %d",ivar1);
count++;
}
}
/*****/
solidf(input)                                /* EXECUTES PRIMITIVE COMMANDS */
char input40;
{
extern double fvar,fvar1,fvar2, fvar3,fvar4;
extern int ivar;
extern int count;
COUNT num;
if (((strcmp(input,PSOLID)== 0)))
{
if(ivar1 == 0)
{
num=0;
printf("COLUMN1 %.1f, %.1f\n",fvar2, fvar3);
col2();
}
if(ivar1 == 1)
{
cone(1, fvar2, fvar3,fvar4);
printf("CONE %.1f, %.1f, %.1f\n",fvar2, fvar3,fvar4);
count++;
}
if(ivar1 == 2)
{
sphere2();
printf("SPHERE %.1f\n",fvar2);
count++;
}
if(ivar1 == 3)
{
torus(3, fvar2, fvar3);
printf("TORUS %.1f %.1f\n",fvar2,fvar3);
count++;
}
solidf2(input);
}
}

```

```

/*****/
solidf2(input)                                /* EXECUTES PRIMITIVE COMMANDS */
char input40;
{
extern double fvar, fvar1, fvar2, fvar3, fvar4;
extern int ivar;
extern int count;
COUNT num;
if(ivar1 ==4)
{
    block( 4, fvar2, fvar3, fvar4);
    printf("BLOCK %.1f, %.1f, %.1f\n",fvar2, fvar3,fvar4);
    count++;
}
if(ivar1 ==5)
{
    wedge(5, fvar2,fvar3,fvar4);
    printf("WEDGE %.1f %.1f %.1f\n",fvar2,fvar3,fvar4);
    count++;
}
if(ivar1 == 6)
{
    tube(6, fvar2,fvar3,fvar4);
    printf("TUBE %.1f %.1f %.1f\n",fvar2,fvar3,fvar4);
    count++;
}
}
/*****/
dispatf(input)                                /* EXECUTES DISPAT COMMANDS */
char input40;
{
extern int ivar,ivar1,ivar2;
extern int count;
if (((strcmp(input,DISPAT)== 0)) && (count < 1))
{
    if(ivar1 == 0)
    {
        /* EXECUTES SHADING COMMANDS */
        dispat(0, ivar2);
        printf("dispat %d, %d\n",ivar1,ivar2);
        count++;
    }
    if(ivar1 == 1)
    {
        dispat(1, ivar2);                                /* EXECUTES TRANSLUCENCY COMMANDS */
        printf("dispat %d, %d\n",ivar1, ivar2);
        count++;
    }
}
}

```

```

if(ivar1 == 2)
{
    dispat(2, ivar2);          /* EXECUTES WIREFRAME COMMANDS */
    printf("dispat %d\n",ivar2);
    count++;
}
if(ivar1 == 3)                /* EXECUTES BACKFACE COMMANDS */
{
    dispat(3, ivar2);
    printf("dispat %d %d\n",ivar1,ivar2);
    count++;
}
if(ivar1 ==4)                 /* EXECUTES FACET COMMANDS */
{
    dispat(4, ivar2);
    printf("dispat %d, %d\n",ivar1, ivar2);
    count++;
}
}

```

```

/*****/
mdtranf(input)                /* EXECUTES TRANSLATION COMMANDS */
char input10;
{
    extern double fvar,fvar1,fvar2,fvar3;
    extern int count;
    if (((strcmp(input,MDTRAN)== 0)) && (count < 1))
    {
        mdtran(fvar1, fvar2, fvar3);
        printf("mdtrantest\n");
        count++;
    }
}

```

```

/*****/
mdidenf(input)                /* EXECUTES MODELING MATRIX RESET COMMANDS */
char input10;
{
    extern int count;
    if (((strcmp(input,MDIDEN)== 0)) && (count < 1))
    {
        printf("mdtest\n");
        mdiden();
        count++;
    }
}

```

```

/*****/
colorf(input)                /* EXECUTES COLORT COMMANDS */
char inputc;
{
extern int ivar,ivar1;
extern int count;
if (((strcmp(input,COLOR)== 0)) && (count < 1))
{
printf("coltest\n");
color(ivar1);
count++;
}
}

/*****/
lutintf(input)               /* EXECUTES LUT SET COMMANDS */
char inputc;
{
extern int ivar,ivar1;
extern int count;
if (((strcmp(input,LUTINT)== 0)) && (count < 1))
{
printf("luttest\n");
lutint(6);
count++;
}
}

/*****/
vwrotzf(input)               /* EXECUTES VIEW WINDOW ROTATE Z COMMANDS */
char inputc;
{
extern int ivar,ivar1;
extern int count;
if (((strcmp(input,VWROTZ)== 0)) && (count < 1))
{
printf("vwztest\n");
vwrotz(ivar1);
count++;
}
}

/*****/
vwrotyf(input)               /* EXECUTES VIEW WINDOW ROTATE Y COMMANDS */
char inputc;
{
extern int ivar,ivar1;
extern int count;

```

```

if (((strcmp(input,VWROPY)== 0)) && (count < 1))
{
    printf("vwytest\n");
    vwroty(ivar1);
    count++;
}
}
/*****/
vwrotxf(input)                /* EXECUTES VIEW WINDOW ROTATE X COMMANDS */
char input0;
{
    extern int ivar,ivar1;
    extern int count;
    if (((strcmp(input,VWROTX)== 0)) && (count < 1))
    {
        printf("vwxtest\n");
        vwrotx(ivar1);
        count++;
    }
}
/*****/
vwrpptf(input)                /* EXECUTES VIEW REFERENCE POINT COMMANDS */

char input0;
{
    extern double fvar,fvar1,fvar2,fvar3;
    extern int count;
    if (((strcmp(input,VWRPT)== 0)) && (count < 1))
    {
        printf("vwrptest\n");
        wrpvt(fvar1,fvar2,fvar3);
        count++;
    }
}
/*****/                /* EXECUTES VIEW WINDOW RESET COMMANDS */
vwidenf(input)
char input0;
{
    extern int count;
    if (((strcmp(input,VWIDEN)== 0)) && (count < 1))
    {
        printf("vwtest\n");
        vwiden();
        count++;
    }
}

```

```

/*****
*****/
lightf(input)                                /* EXECUTES LIGHTING ATTRIBUTES COMMANDS */
char input[];
{
extern int ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6;
extern double fvar,fvar1;
extern int count;
if (((strcmp(input,LIGHT)== 0)) && (count < 1))
{
printf("lighttest %d, %.2f, %d, %.1f, %.1f, %.1f\n",ivar1,fvar2,ivar3,fvar4,fvar5,fvar6);
light(ivar1,fvar2,ivar3,fvar4,fvar5,fvar6);
count++;
}
}
/*****
*****/
litenf(input)                                /* ENABLES LIGHTING NUMBER COMMANDS */
char input[];
{
extern int ivar,ivar1,ivar2;
extern int count;
if (((strcmp(input,LITEN)== 0)) && (count < 1))
{
printf("litentest\n");
liten(ivar1,ivar2);
count++;
}
}
/*****
*****/
distanf(input)                               /* SETS VIEWING DISTANCE COMMANDS */
char input[];
{
extern double fvar1;
extern int count;
if (((strcmp(input,DISTAN)== 0)) && (count < 1))
{
printf("disttest\n");
distan(fvar1);
count++;
}
}
/*****
*****/

```



```

projectf(input)                                /* SETS PROJECTION ANGLE COMMANDS */
char input10;
{
extern int ivar1;
extern int count;
if (((strcmp(input,PROJECT)== 0)) && (count < 1))
{
printf("projtest\n");
project(ivar1);
count++;
}
}
/*****/
windowf(input)                                /* SETS WINDOW SIZE COMMANDS */
char input10;
{
extern double winx1, winx2, winy1, winy2;
extern int count;
if (((strcmp(input, WINDOW)== 0)) && (count < 1))
{
printf("windotest\n");
window(winx1, winx2, winy1, winy2);
count++;
}
}
/*****/
vwportf(input)                                /* SETS VIEW PORT SIZE COMMANDS */
char input10;
{
extern int wdown, vwx1, vwx2, vwy1, vwy2;
extern int vwx1v, vwx2v, vwy1v, vwy2v;
extern int count;
if (((strcmp(input,VWPORT)== 0)) && (count < 1))
{
if(wdown==1)
{
vwport(vwx1v, vwx2v, vwy1v, vwy2v);
}
if(wdown==0)
{
vwport(vwx1, vwx2, vwy1, vwy2);
}
printf("vportest\n");
count++;
}
}
/*****/

```

```

displaf(input)                /* SETS GRAPIC DISPLAY COMMANDS */
char inputio;
{
extern int ivarl;
extern int count;
if (((strcmp(input,DISPLA)== 0)) && (count < 1))
{
printf("testdisp\n");
displa(ivarl);
count++;
}
}

/*****/
twvisf(input)                /* SETS TEXTWINDOW VISIBLE COMMANDS */
char inputio;
{
extern int ivar,ivarl;
extern int count;
if (((strcmp(input,TWVIS)== 0)) && (count < 1))
{
printf("twvistest\n");
twvis(1);
count++;
}
}

/*****/
twposf(input)                /* SETS TEXTWINDOW POSITION COMMANDS */
char inputio;
{
extern int ivar,ivarl,ivar2,ivar3,ivar4,ivar5,ivar6,count;
if( ((strcmp(input,TWPOS)== 0)) && (count< 1))
{
printf( "twpostest,%d,%d, %d, %d, %d,%d\n",ivarl,ivar2,ivar3,ivar4,ivar5,ivar6);
twpos(ivarl,ivar2,ivar3,ivar4,ivar5,ivar6);
count++;
}
}

/*****/
clearif(input)                /* CLEARS INTENSITY BUFFER COMMANDS */
char inputio;
{
extern int count;
if( ((strcmp(input,CLEARI)== 0)) && (count< 1))
{
printf("CLEARItest\n");
cleari(0);
count++;
}
}

```

```

/*****
clearsf(input)                                /* CLEARS SCREEN COMMANDS */
char input;
{
    extern int count;
    if (((strcmp(input,CLEAR)== 0)) && (count< 1))
    {
        printf("clertest\n");
        clears(0);
        count++;
    }
}

/*****
caf(input)                                    /* ASCII COMMANDS */
char input;
{
    extern int count;
    if (((strcmp(input,CA)== 0)) && (count==0))
    {
        printf("catest\n");
        ca();
        count++;
    }
}

/*****
pginitf(input)                                /* INITIATES GRAPHICS COMMANDS */
char input;
{
    extern int count;
    if (((strcmp(input,PGINIT) == 0)) && (count==0))
    {
        printf("pgtest\n");
        pginit(0, 1);
        count++;
    }
}

/*****
clipyf(input)                                /* SETS YOM CLIPPING PLANE */
char input;
{
    extern int count;
    extern int ivar,ivar1;
    if (((strcmp(input,CLIPY) == 0)) && (count==0))
    {
        printf("CLIPYtest\n");
        clipy(ivar1);
        count++;
    }
}

```

```

/*****/
cliphf(input)
char input;
{
    extern int count;
    extern int ivar,ivar1;
    if (((strcmp(input,CLIPH) == 0)) && (count==0))
    {
        printf("CLIPHtest\n");
        cliph(ivar1);
        count++;
    }
}

/*****/
disthf(input)
char input;
{
    extern double fvar,fvar1;
    extern int count;
    if (((strcmp(input,DISTH) == 0)) && (count==0))
    {
        printf("DISTHtest\n");
        disth(fvar1);
        count++;
    }
}

/*****/
distyf(input)
char input;
{
    extern double fvar,fvar1;
    extern int count;
    if (((strcmp(input,DISTY) == 0)) && (count==0))
    {
        printf("DISTYtest\n");
        disty(fvar1);
        count++;
    }
}

/* SETS HITHER CLIPPING PLANE */
/* SETS HITHER CLIPPING PLANE DISTANCE*/
/* SETS YON CLIPPING PLANE DISTANCE*/

```

```

/*****/
clbegf(input)                                /* BEGINS COMMAND LIST */
char input10;
{
extern int ivar,ivar1;
extern int count;
if (((strcmp(input,CLBEG) == 0)) && (count==0))
{
printf("CLISTBEGtest\n");
clbeg(ivar1);
clcount++;
count++;
}
}

/*****/
/*****/
clendf(input)                                /* ENDS COMMAND LIST */
char input10;
{
extern int count;
if (((strcmp(input,CLEND) == 0)) && (count==0))
{
printf("CLIST in ENDtest\n");
clend();
count++;
}
}

/*****/
+ clrunf(input)                                /* RUNS COMMAND LIST */
char input10;
{
extern int count;
extern int ivar,ivar1;
if (((strcmp(input,CLRUN) == 0)) && (count==0))
{
printf("CLRUNtest %d\n",ivar1);
clrun(ivar1);
count++;
}
}

```

```

/******
scalf(input)                /* ACTIVATES SCALE COMMAND */
char input;
{
    extern int count;
    extern double fvar,fvar1,fvar2,fvar3;
    if (((strcmp(input,MDSCAL) == 0)) && (count==0))
    {
        printf("MDSCALE test%.2f,%.2f,%.2f,\n",fvar1, fvar2, fvar3);
        mdscal(fvar1, fvar2, fvar3);
        count++;
    }
}

/******
rotzf(input)                /* ACTIVATES ROTATION X COMMAND */
char input;
{
    extern int cont,ivar,ivar1;
    if (((strcmp(input,MDROTX) == 0)) && (count==0))
    {
        printf("rot x test\n");
        mdrotx(ivar1);
        count++;
    }
}

/******
rotyf(input)                /* ACTIVATES ROTATION Y COMMAND */
char input;
{
    extern int count,ivar,ivar1;
    if (((strcmp(input,MDROTY) == 0)) && (count==0))
    {
        printf(" ROT Y test\n");
        mdroty(ivar1);
        count++;
    }
}

/******
rotzf(input)                /* ACTIVATES ROTATION Z COMMAND */
char input;
{
    extern int count,ivar,ivar1;
    if (((strcmp(input,MDROTZ) == 0)) && (count==0))
    {
        printf("ROT Ztest\n");
        mdrotz(ivar1);
        count++;
    }
}

```

```

/*****
orgf(input)                                /* ACTIVATES MODELING ORIGIN COMMAND */
char input;
{
    extern int count;
    extern double fvar,fvar1,fvar2,fvar3;
    if (((strcmp(input,MDORG) == 0)) && (count==0))
    {
        printf("MDORG test%.2f,%.2f,%.2f,\n",fvar1, fvar2, fvar3);
        mdorg(fvar1, fvar2, fvar3);
        count++;
    }
}
*****/

ssolidf(input)                             /* ACTIVATES SWEPT SOLID COMMAND */
char input;
{
    extern int count;
    extern double fvar2,fvar3,rotnum;
    extern int    ivar,ivar1;
    extern double *xlist;
    extern double *ylist;
    if (((strcmp(input,SSOLID) == 0)) && (count==0))
    {
        printf("ssolid test\n");
        ssolid (ivar1, ivar2,xlist,ylist,rotnum);
        count++;
    }
}

```

```

: THIS SET OF FILES READS IN THE COMMAND LIST FOR VIEWING, PLAN AND ELEVATION*/
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
#include "std.h"
#include "commands.h"

./*****/
getfile()                                /* RECEIVES COMMAND LIST FOR VIEWING */
{
extern int vx1v, vx2v, vy1v, vy2v;
extern int shade, planon, wdw;
clears(0);
cleari(0);
vwport(0,639,0,479);
xyzsend();                             /*STORES X Y Z CURSOR LOCATION LOCATION */
shade=1;                               /* SETS SHADING ON */
planon=0;                              /* SETS VIEWING ON */
setupvars();                           /* SETS UP VIEWING VARIABLES */
if(wdw==1)                             /* IF WINDOWS ARE ON */
vwport(vx1v,vx2v,vy1v,vy2v);          /* SET SMALL PERSPECTIVE VIEW */
getfile0();                            /* READS IN COMMAND LIST FILE */
getfile1();                            /* RESETS SCREEN WITH CURSOR */
}

./*****/
getfile0()
{

extern int randisk,wdw;
extern int count,planon;
extern double x, y, z;
extern int vectotal,total;
extern int sscount,ssnum,total,ssct,vtot;
extern double xlist25;
extern double ylist25;
extern double *xptr;
extern double *yptr;
static char input0;
static char str10;
FILE *fl;
char string0;

if(randisk==0)                          /* SETS FILE TO READ RANDISK OR DISK*/
fl = fopen("command.d","r");
if(randisk==1)
fl = fopen("d:command.d","r");

```



```

yptr=ylist;          /* DEFINES POINTER FOR VECTOR LIST IN SWEEP SOLIDS */
xptr=xlist;
vectotal=0;          /* RESETS VECTOR TOTAL */
vtot=0;
ssct=0;
total=0;             /* SET COMMAND TOTAL */
sscount=0;
while (fgets(input,LIM,fl)!=NULL) /* READS FILE TO END */
{
    strcpy(str1,input); /* COPS STRING TO STR1 */
    fit(input,6);       /* TAKES FIRST 6 CHARACTERS */
    ssolidr(input,str1); /* IF SWEEP SOLID SENDS TO SS FUNCTION */
    if(sscount==0)      /* IF SS FINISHED THEN BEGIN */
    {
        strcpy(string,&str1); /* COPS STRING DELETING FIRST SIX CHARACTERS */
        tok(string);          /* SEPARATES NUMBER CHARACTERS */
        count=0;
        func2(input);         /* STRING COMMAND MATCHING FUNCTION */
    }
}
} /* END */
./*****/
ssolidr(input,str1)    /* READS SSOLID COMMAND AND EXECUTES */

char input40;
char str140;

{
extern int sscount,ssnum,total,ssct,vtot;
extern int wdw;
extern int count,planon;
extern double x, y, z;
extern double depth1,depth2,depth3,rotnum,fvar, fvar1,fvar2,fvar3;
extern int ivar, ivar1, ivar2;
extern int vectotal,total;
int ct = 0;
extern double xlist25;
extern double ylist25;
extern double *xptr;
extern double *yptr;
static char string40;

```

```

if (((strcmp(input,SSOLID) == 0))||(vtot!=0)||(sscount==1)) /* TEST FOR SSOLID COMMAND */
{
    strcpy(string,&str16); /* TAKES STRING FROM 6 CHARACTERS BACK*/
    tok(string); /* DIVIDES STRING INTO GROUPS */
    if(vectotal==0) /* ASSIGNS VECTOR TOTAL NUMBER */
    {
        ssnum=ivar1;
        vectotal=ivar2;
        vtot=ivar2;
    }
    if((sscount==1)&&(ssnum==1)) /* TESTS FOR SSOLID ROTATE */
    {
        rotnum=fvar1; /* ASSIGNS ROTATION ANGLE */
        ssolid(ssnum,vectotal,xlist,ylist,rotnum); /* EXECUTES SSOLID COMMAND */
        vectotal=0; /* RESETS ALL VALUES ASSOCIATED WITH SSOLID */
        rotnum=0.0;
        vtot=0;
        ssct=0;
        total=0;
        sscount=-1;
    }
    if((sscount==1)&&(ssnum==0))
    {
        depth1 =fvar1; /* ASSIGNS DEPTH VALUE */
        depth2 =fvar2;
        depth3 =fvar3;
        ssolid(ssnum,vectotal,xlist,ylist,depth1,depth2,depth3);
        /* EXECUTES EXTRUDED SSOLID COMMAND */
        vectotal=0; /* RESETS VALUES */
        rotnum=0.0;
        vtot=0;
        ssct=0;
        total=0;
        sscount=-1;
    }
    if((total<vtot)&&(sscount>0))/*EXECUTES UNTIL VECTOR TOTAL EXECUTED IS GREATER THAN TOTAL */
    {
        xlistset=fvar1; /* ASSIGNS TO X VECTOR LIST */
        ylistset=fvar2; /* ASSIGNS VALUE TO Y VECTOR LIST */
        total=ssct; /*COUNTS NUMBER OF TIMES VECTORS HAVE BEEN EXECUTED */
        ssct++;
        if(total==vtot-1) /* TESTS FOR FINAL VECTOR */
        {
            sscount=-2; /* ASSIGNS LIST TO POINTER VALUE */
            xptr=xlist;
            yptr=ylist;
            vtot=0;
        }
    }
}

```

```

        sscount++;
    }
}
}
./xxxxxxxxxxxxxxxx/
getfile1()
{
extern double orgx,orgy,orgz, winx1, winx2, winy1, winy2;
extern int planon, wdown;
extern int vxz , vzy, vyz;
extern double vwptx,vwpty,vwptz, x, y, z,sx,sy,sz;
extern int vxz1, vxz2, vzy1, vzy2;
extern int vxzlv, vxz2v, vzylv, vzy2v;
extern double disy, dist,dish;
x=0.0; /* SET XYZ TO ZERO */
y=0.0;
z=0.0;
mdiden(); /* RESETS MODELING MATRIX */
xyzreturn(); /* RETURNS PREVIOUS X Y Z VALUE */
mdtran(x, y, z); /* TRANSLATES CURSOR LOCATION */
wirefr(2, 0); /* SETS TO WIREFRAME */
window(winx1, winx2, winy1, winy2); /* SETS WINDOW SIZE */
if(wdown==1) /* TEST FOR MULTIPLE WINDOWS */
{
dispi(0); /* DISPLAYS SHADED IMAGE */
cplan(); /* DRAWS PLAN */
celevat(); /* DRAWS ELEVATION */
vwrotx(vxz); /* ROTATES VIEW X AXIS */
vwroty(vzy); /* ROTATES VIEW Y AXIS */
vwrotz(vyz); /* ROTATES VIEW Z AXIS */
vwport(vxzlv, vxz2v, vzylv, vzy2v); /* SETS PERSPECTIVE VIEW */
project(50); /* SETS PROJECTION ANGLE */
disth(dish); /* SETS HITHER CLIPPING */
disty(disz); /* SETS YOM CLIPPING */
distan(dist); /* SETS VIEWING DISTANCE */
vwprt(vwptx,vwpty,vwptz); /* SETS VIEWING REFERENCE POINT */
planon=0; /* VALUE TO PERSPECTIVE VIEW */
windows(); /* DRAWS CURSOR TO WINDOWS */
wirefr(2, 0); /* SETS TO WIRE FRAME */
linfun(0); /* SETS REPLACE LINE TYPE */
color(250); /* SETS COLOR */
move(winx1,winyl); /* DRAWS BORDERS */
rect(winx2, winy2);
}
}

```

```

if(wdow==0)                                /* TEST FOR SINGLE SCREEN */
{
    viewport(vwx1, vwx2, vwy1, vwy2);      /* SETS FULL VIEWING WINDOW */
    project(50);                           /* SET PROJECTION ANGLE */
    vwrpt(vwptx, vwpty, vwptz);            /* SETS REFERENCE POINT */
    planon=0;                              /* SET TO PERSPECTIVE */
    mdorg(orgx, orgy, orgz);               /* SETS MODELING ORIGIN */
    rcntxa();                              /* TEST ROTATION ORDER */
    rcntya();
    rcntza();
    mdscal(sx, sy, sz);                   /* SCALES */
    dispi(0);                             /* DISPLAYS SCREEN */
    cursor();                             /* DRAWS CURSOR */
    color(250);                           /* DRAWS BORDER */
    move(winx1, winy1);
    rect(winx2, winy2);
}
clrwn(251);                               /* MAIN MENU LIST */
}
/*****
rcntxa()                                  /* THE FOLLOWING FUNCTIONS CHECK ROTATION ORDER */
{                                          /* AND EXECUTE ROTATION FUNCTIONS */
extern int rcntx, rcnty, rcntz;
extern int rx, ry, rz, morg;
extern double x, y, z, xf, yf, zf;
if (rcntx==1)
{
    if((rcnty==0)&&(rcntz==0))
        mdrotx(rx);
    if((rcnty==2)&&(rcntz==0))
    {
        mdrotx(rx);
        mdroty(ry);
    }
    if((rcnty==2)&&(rcntz==3))
    {
        mdrotx(rx);
        mdroty(ry);
        mdrotz(rz);
    }
    if((rcntz==2)&&(rcnty==0))
    {
        mdrotx(rx);
        mdrotz(rz);
    }
}

```

```

    if((rcntz==2)&&(rcnty==3))
    {
        mdrotx(rx);
        mdrotz(rz);
        mdroty(ry);
    }
}

```

```

./xxxxxxxxxxxxxxxxxxxx/

```

```

rcntya()                                /* FOR Y ROTATION VALUES */

```

```

{
extern int rcntx,rcnty,rcntz;
extern int rz,ry,rx,morg;
extern double x,y,z,xf,yf,zf;
if (rcnty==1)
{
    if((rcntx==0)&&(rcntz==0))
        mdroty(ry);
    if((rcntx==2)&&(rcntz==0))
    {
        mdroty(ry);
        mdrotx(rx);
    }
    if((rcntx==2)&&(rcntz==3))
    {
        mdroty(ry);
        mdrotx(rx);
        mdrotz(rz);
    }
    if((rcntz==2)&&(rcntx==0))
    {
        mdroty(ry);
        mdrotz(rz);
    }
    if((rcntz==2)&&(rcntx==3))
    {
        mdroty(ry);
        mdrotz(rz);
        mdrotx(rx);
    }
}
}

```

```

/*****/
rcntza()
/* FOR Y ROTATION VALUES */
{
extern int rcntx,rcnty,rcntz;
extern int rz,ry,rx,morg;
extern double x,y,z,xf,yf,zf;
if (rcntz==1)
{
if((rcntx==0)&&(rcnty==0) )
mdrotz(rz);
if((rcntx==2)&&(rcnty==0) )
{
mdrotz(rz);
mdrotx(rx);
}
if((rcntx==2)&&(rcnty==3) )
{
mdrotz(rz);
mdrotx(rx);
mdroty(ry);
}
if((rcnty==2)&&(rcntx==0) )
{
mdrotz(rz);
mdroty(ry);
}
if((rcnty==2)&&(rcntx==3) )
{
mdrotz(rz);
mdroty(ry);
mdrotx(rx);
}
}
}
/*****/
cplan() /* SAME AS GETFILE EXCEPT A PLAN VIEW IS SET */
{
extern int wrow;
extern int count,planon;
extern double orgx,orgy,orgz,sx,sy,sz,x,y,z;
extern double depth1,depth2,depth3,rotnum,fvar,fvar1,fvar2,fvar3;
extern int ivar,ivar1,ivar2;
extern int vectotal;
extern int sscount,ssnum,total,ssct,vtot;
static char input0;

```

```

static char str140;
int ct = 0;
extern int vx1p, vx2p, vy1p, vy2p;
extern int randisk, vwy;
extern double winx1, winx2, winy1, winy2;

char string40;
extern double xlist225;
extern double ylist225;
extern double *xptr;
extern double *yptr;
FILE *f1;
int ret;
yptr=ylist;
xptr=xlist;
vectotal=0;
vtot=0;
ssct=0;
total=0;
sscount=0;
    if(randisk==0)
        f1 = fopen("command.d","r");
    if(randisk==1)
        f1 = fopen("d:command.d","r");

ret = 0;
xyzsend();
setupvars();
planon=1;
dispat(2,0);
if(wdow==1)
{
    viewport(vx1p, vx2p, vy1p, vy2p);
    linfun(0);
    move(winx1, winy1);
    rect(winx2, winy2);
}
if(wdow==0)
    clears(0);
LINFUN (0);
window(winx1, winx2, winy1, winy2);
widen();
project(0);
vwroty(vwy);
vwrotx(270);

```

```

    dispat(2,1);
while (fgets(input,LIM,f1)!=NULL)
    {
        strcpy(str1,input);
        fit(input,6);
        ssolidr(input,str1);
        if(sscount==0)
            {
                strcpy(string,&str1);
                tok(string);
                count=0;
                funcp(input);
                ct++;
            }
    }
fclose(f1);
x=0.0;
y=0.0;
z=0.0;
mdiden();
xyzreturn();
mdtran(x,y,z);
planon=1;
mdorg(orgx,orgy,orgz);
rentxa();
rentya();
rentza();
mdscal(sx,sx,sx);
dispat(2,0);
if(wdow==0)
    {
        clrwn(251);
        setscrn();
    }
}

```

```

/*****

```

```

    funcp(input)
    char input10;
    {
        vwrptf(input);
        lutintf(input);
        colorf(input);
        mdidenf(input);
        mdtranf(input);
        solidf(input);
        rotxf(input);
        scalf(input);
    }

```

```

/* FUCTIONS USED BY PLAN FUNCTION MAY BE ELIMINATED */

```



```

    rotyf(input);
    rotzf(input);
    orgf(input);
    solbegf(input);
    solendf(input);
    solbopf(input);
    solrunf(input);
    clbegf(input);
    clendf(input);
    clrunf(input);

```

```

}

```

```

/*****

```

```

    celevat() /* SAME AS PLAN AND GETFILE EXCEPT AN ELEVATION VIEW IS SET */

```

```

{
    extern int wdw;
    extern int count,planon;
    extern double sx,sy,sz,x,y,z,orgx,orgy,orgz ;
    extern double depth1,depth2,depth3,rotnum,fvar, fvar1,fvar2,fvar3;
    extern int ivar, ivar1, ivar2;
    extern int vectotal;
    extern int sscount,ssnum,total,ssct,vtot;
    static char input40;
    static char str140;
    int ct = 0;
    extern int vx1e, vx2e, vy1e, vy2e;
    extern int randisk,vwy;
    extern double winx1, winx2, winy1, winy2;

```

```

    char string40;
    extern double xlist225;
    extern double ylist225;
    extern double *xptra;
    extern double *yptra;
    FILE *fl;
    int ret;
    yptra=ylist;
    xptra=xlist;
    vectotal=0;
    vtot=0;
    ssct=0;
    total=0;
    sscount=0;
    if(randisk==0)
        fl = fopen("command.d","r");
    if(randisk==1)
        fl = fopen("d:command.d","r");

```

```

xyzsend();
setupvars();
planon=2;
if(wdow==1)
{
    viewport(vwx1e, vwx2e, vwyle, vwyl2e);
}
if(wdow==0)
    clears(0);
window(winx1, winx2, winyl, winyl2);
linfun(0);
color(250);
move(winx1, winyl);
rect(winx2, winyl2);
dispat(2,0);
vwiden();
project(0);
vrotx(0);
vroty(vwy);
start:
while (fgets(input,LIM,fl)!=NULL)
{
    strcpy(str1,input);
    fit(input,6);
    ssolidr(input,str1);
    if(sscount==0)
    {
        strcpy(string,&str1);
        tok(string);
        count=0;
        funcp(input);
        ct++;
    }
}
fclose(fl);
if(randisk==0)
    fl = fopen("command.d","r");
if(randisk==1)
    fl = fopen("d:command.d","r");
x=0.0;
y=0.0;
z=0.0;
mdiden();
xyzreturn();
mdtran(x,y,z);
planon=2;

```

```

mdorg(orgx,orgy,orgz );
rentxa();
rentya();
rentza();
mdscal(sx,sx,sx);
dispat(2,0);
if(wdow==0)
{
    clrun(251);
    setscrn();
}
}

```

```

/*****

```

```

*****/

```

```

funcel(input) /* COMMANDS USED BY ELEVATION FUNCTION */
char input10;
{
    vwrptf(input);
    lutintf(input);
    colorf(input);
    mdidenf(input);
    mdtranf(input);
    solidf(input);
    rotxf(input);
    scalf(input);
    rotyf(input);
    rotzf(input);
    orgf(input);
    solbegf(input);
    solendf(input);
    solbopf(input);
    solrunf(input);
    clbegf(input);
    clendf(input);
    clrunf(input);
}

```

```

/*****/

```

* THESE FUNCTIONS INCLUDE THE MAIN MENU AND SUB MENUS COLOR; SHADING; FACETING; */
 * MODELING; TRANSLUCENCY */

```
#include "keys.h"
#include "std.h"
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
#include <stdio.h>
```

./*****

```
mainmenu()
```

```
{
extern int wdw;          /* VARIABLE FOR WINDOW SELECT */
int key;                /* READS IN KEY VALUE */
int cnt;                /* FLAG FOR SUB FUNCTIONS */
if(wdw==0)              /* ERASES CURSOR */
    cursor();
if(wdw==1)
    windows();
cnt=0;
```

```
maindisp();             /* DISPLAYS MAIN MENU */
```

```
move(75.0, 465.0);
```

```
color(32);
```

```
rectr(75.0, 15.0);      /* DRAWS MENU CURSOR */
```

```
key=getche();
```

```
while(key != QS)        /* SETS LOOP */
```

```
{
    switch(key)
    {
```

```
        case CS: cnt=4;          /* SETS COLOR */
```

```
            goto out;
```

```
            break;
```

```
        case SS: cnt=1;          /* SET SOLIDS */
```

```
            goto out;
```

```
            break;
```

```
        case MS: cnt=2;          /* SETS MODELING */
```

```
            goto out;
```

```
            break;
```

```
        case DS: cnt=0;          /* SET DIMENSIONS */
```

```
            goto out;
```

```
            break;
```

```
        case RS: goto out;       /* ERASES */
```

```
            break;
```

```
        case PS :                /* MOVES CURSOR TO LEFT */
```

```
        case LARR :
```

```

case FOUR : linfun(2);
            color(32);
            rectr(75.0, 15.0);
            mover(75.0, 0.0);
            rectr(75.0, 15.0);
            cnt++;
            if(cnt > 5)
            {
                rectr(75.0, 15.0);
                move(75.0, 465.0);
                rectr(75.0, 15.0);
                cnt=0;
            }
            break;
case RARR :                               /* MOVES CURSOR TO THE RIGHT */
case F10 :
case SIX: linfun(2);
            color(32);
            rectr(75.0, 15.0);
            mover(-75.0, 0.0);
            rectr(75.0, 15.0);
            cnt--;
            if(cnt < 0)
            {
                rectr(75.0, 15.0);
                move(450.0, 465.0);
                rectr(75.0, 15.0);
                cnt=5;
            }
            break;
case SPACE : goto out;
case ESC : cnt = 99;
           goto out;
default: ;
}
key=getche();
}
out:
mainmenu2(cnt);
}

```

```
./*****
```

```

maindisp()                                /* SET MENU DISPLAY AT THE TOP OF SCREEN */
{
    clbeg(251);                            /* STARTS COMMANDLIST */
    linfun(2);                             /* SET TO XOR */
    VMPORT(0.639,0.479);                  /* SETS VIEW PORT */
    window(0.0 , 639.0, 0.0, 479.0);      /* SETS WINDOW */
    TSIZE(10.0);                          /* SETS TEXT SIZE */
    TJUST(1,1);                           /* SETS JUSTIFICATION */
    MOVE(5.0, 468.0);                     /* MOVES TO TOP LEFT CORNER */
    COLOR(32);                             /* SET COLOR RED */
    text("MAIN ");                        /* PRINTS "MAIN" */
    MOVE(75.0, 0.0);
    COLOR(32);
    text("D ");
    COLOR(255);
    text(" IMM ");
    COLOR(32);
    MOVE(50.0,0.0);
    text("S ");
    COLOR(255);
    text(" OLD ");
    MOVE(50.0, 0.0);
    COLOR(32);
    text("M ");
    COLOR(255);
    text(" ODL ");
    MOVE(50.0,0.0);
    COLOR(32);
    text("O ");
    COLOR(255);
    text(" RGM ");
    MOVE(50.0, 0.0);
    COLOR(32);
    text("V ");
    COLOR(255);
    text(" IRM ");
    MOVE(50.0,0.0);
    COLOR(32);
    text("C ");
    COLOR(255);
    text(" OLR ");
    MOVE(50.0,0.0);
    COLOR(32);
    text("R ");
    COLOR(255);
    text(" NOV ");
    MOVE(50.0,0.0);
    COLOR(32);

```

```

text("W ");
COLOR(255);
text(" NDW ");
MOVER(50.0,0.0);
COLOR(32);
text("F ");
COLOR(255);
text(" ILR ");
MOVER(50.0,0.0);
COLOR(32);
text("R ");
COLOR(255);
text(" RAS ");
MOVER(50.0,0.0);

clend();
clrun(251);
}
/* ENDS COMMAND LIST */
/* RUNS COMMAND LIST */
}
/*****/
mainmenu2(cnt)
int cnt;
{
extern int wdw;
extern double winx1, winx2, winy1, winy2;
extern int vwx1, vwx2, vwyl, vwyl2;
int key;
rectr(75.0, 15.0);
clrun(251);
if(cnt==0)
{
setscrn();
cmodel();
setscrn();
goto end;
}
if(cnt==1)
{
primitive();
goto end;
}
if(cnt==2)
{
mmodeling();
goto end;
}
if(cnt==3)
{

```

```

/*  mview(); */
    goto end;
}
if(cnt==4)
{
    ccolor();                /* COLOR FUNCTIONS */
    goto end;
}
end;;
}
/*****/
mmodeling()                /* MAIN MODELING MENU FUNCTIONS */
{
    extern int wdw;
    extern double winx1, winx2, winy1, winy2;
    extern int vx1, vx2, vy1, vy2;
    int key;
    int cnt2;
    cnt2=0;
    modeldisp();
    move(75.0,465.0);
    color(32);
    rectr(75.0, 15.0);
    key=getche();
    while(key != QU)
    {
        switch(key)
        {
            case CS : cnt2=6;                /* GOTO CENTERING */
                        goto out;
                        break;
            case SS: cnt2=5;
                        goto out;            /* GOTO SCALE */
                        break;
            case RS : cnt2=0;
                        goto out;            /* GOTO ROTATE */
                        break;
            case FS : cnt2=1;
                        goto out;            /* GOTO FACT */
                        break;
            case GS : cnt2=2;
                        goto out;            /* GOTO GOURAUD SHADE */
                        break;
            case AS : cnt2=3;
                        goto out;            /* GO TO ANGULAR SHADE */
                        break;
            case TS : cnt2=4;
                        goto out;            /* GO TO TRANSLUCENCY */
                        break;

```



```

case F10 :
case RARR : /* MOVE CURSOR TO THE RIGHT */
case SIX : linfun(2);
            color(32);
            rectr(75.0, 15.0);
            MOVER(75.0, 0.0);
            rectr(75.0, 15.0);
            cnt2++;
            if(cnt2>6)
            {
                rectr(75.0, 15.0);
                MOVER(75.0, 465.0);
                rectr(75.0, 15.0);
                cnt2=0;
            }
            break;
case F9 :
case LARR : /* MOVE CURSOR TO THE LEFT */
case FOUR : linfun(2);
            color(32);
            rectr(75.0, 15.0);
            MOVER(-75.0, 0.0);
            rectr(75.0, 15.0);
            cnt2--;
            if(cnt2<0)
            {
                rectr(75.0, 15.0);
                move(525.0, 465.0);
                rectr(75.0, 15.0);
                cnt2=6;
            }
            break;
case SPACE : goto out; /* SELECT */
case ESC : cnt2=99; /* RETURN TO MAIN */
           goto out;

default : ;

    }
    key=getche();
}

out:
modeling2(cnt2);
}

```

```

/******
modeldisp()          /* MODELING DISPLAY */
{
  clbeg(253);
  linfun(2);
  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(5.0, 468.0);
  COLOR(32);
  text(" MODEL ");
  MOVER(75.0, 0.0);
  COLOR(255);
  text(" ROTATE ");
  MOVER(75.0, 0.0);
  text(" FACET ");
  MOVER(75.0, 0.0);
  text(" GSHADE ");
  MOVER(75.0, 0.0);
  text(" ASHADE ");
  MOVER(75.0, 0.0);
  text(" TRANSL ");
  MOVER(75.0, 0.0);
  text(" SCALE ");
  MOVER(75.0, 0.0);
  text(" CENTER ");
  clend();
  clrun(253);
}
/******
mmodeling2(cnt2)      /* BRANCHES TO FUNCTIONS */
int cnt2;
{
  extern int rx,ry,rz,rcntx,rcnty,rcntz,wdow;
  extern double winx1, winx2, winy1, winy2;
  extern int vx1, vx2, vy1, vy2;

  rectr(75.0, 15.0);
  clrun(253);
  if(cnt2==0)
  {
    mrotate();
  }
  if(cnt2==1)
  {
    mtesal();
  }
}

```

```

if(cnt2==2)
{
    mgouraud();
}
if(cnt2==3)
{
    mconshade();
}
if(cnt2==4)
{
    mtranslu();
}
if(cnt2==5)
{
    mscale();
}
if(cnt2==6)
{
    mcenter();
}
windowset();
}
/*      RESET WINDOW AND VIEW WINDOW */

/*****
/*****
mtesal()
{
extern int tess,wrow;
int key;
int cnt3,fcnt;
windowset();
    fcnt=0;
    if(fcnt==0)
    {
        tessal(4,0);
        fcnt++;
    }

if(wrow==0)
cursor();
if(wrow==1)
windows();
cnt3=0;
tessaldisp();
move(60.0, 463.0);
color(32);
rectr(30.0, 15.0);
key=getche();

```



```

windowset();
setscrn();
tesscurs(cnt3);
setscrn();
    VWPORT(0,639,0,479);
    window(0.0 , 639.0, 0.0, 479.0);
    key=getche();
}
out:
ntesal2(cnt3);
windowset();
setscrn();
}

tessaldisp()
{
clbeg(253);
linfun(2);
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
TSIZE(10.0);
TJUST(1,1);
MOVE(3.0, 465.0);
COLOR(32);
text("FACET");
MOVE(60.0, 0.0);
text("90");
MOVE(60.0, 0.0);
text("60");
MOVE(60.0, 0.0);
COLOR(255);
text("45");
MOVE(60.0, 0.0);
text("30");
MOVE(60.0, 0.0);
text("20");
MOVE(60.0, 0.0);
text("15");
MOVE(60.0, 0.0);
text("10");
MOVE(60.0,0.0);
text("6");
MOVE(60.0,0.0);
text("EXIT");
clend();
clrun(253);
}

```

```

./*****/
tesscurs(cnt3)
int cnt3;
{
extern int tess,wrow;
extern double winx1, winx2, winy1, winy2;
extern int vxw1, vxw2, vwy1, vwy2;
char ch3;
if(cnt3==0)
{
tessal(4,0);
}
if(cnt3==1)
{
tessal(4,1);
}
if(cnt3==2)
{
tessal(4,2);
}
if(cnt3==3)
{
tessal(4,3);
}
if(cnt3==4)
{
tessal(4,4);
}
if(cnt3==5)
{
tessal(4,5);
}
if(cnt3==6)
{
tessal(4,6);
}
if(cnt3==7)
{
tessal(4,7);
}
}

```

```

/*****/
ntesal2(cnt3)
int cnt3;          /* SEND TO FILE BRANCH FUNCTIONS */
{
extern int tess, wrow;
extern double winx1, winx2, winy1, winy2;
extern int vxw1, vxw2, vwy1, vwy2;
char ch3;
clrwn(253);
if(cnt3==0)
{
    m4();
}
if(cnt3==1)
{
    m6();
}
if(cnt3==2)
{
    m8();
}
if(cnt3==3)
{
    m12();
}
if(cnt3==4)
{
    m18();
}
if(cnt3==5)
{
    m24();
}
if(cnt3==6)
{
    m36();
}
if(cnt3==7)
{
    m60();
}
if(wrow==0)
{
    window(winx1, winx2, winy1, winy2);
    VWPORT(vxw1, vxw2, vwy1, vwy2);
}
}

```

```

if(wdow==1)
{
    window(-640.0, 640.0, -479.0, 479.0);
    viewport(0, 319, 0, 239);
}
printf("\ncount => %d",cnt3);
}
/*****/
M4()
{
    extern int shad,tess,randisk;
    FILE *cl;
    if(randisk==0)                /* FILE AND SET 0 FACET LEVEL */
        cl = fopen("command.d", "a");

    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"dispat 0, 0\n");
    fprintf(cl,"dispat 4, 0\n");
    fclose(cl);
    shad=0;
    tess=0;
    dispat(0, 0);
    tessal(4, 0);
}
/*****/
M6()                /* FILE AND SET 1 FACET LEVEL */
{
    extern int tess,randisk,shad;
    FILE *cl;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"dispat 0, 0\n");
    fprintf(cl,"dispat 4, 1\n");
    fclose(cl);
    dispat(0, 0);
    shad=0;
    tess=1;
    tessal(4, 1);
}

```



```

/*****/
M8()
{
extern int shad,tess,randisk;
FILE *cl;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 4, 2\n");
fprintf(cl,"dispat 0, 0\n");
fclose(cl);
tess=2;
shad=0;
dispat(0, 0);
tessal(4, 2);
}

/*****/
M12()
{
extern int tess;
FILE *cl;
extern int randisk;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 4, 3\n");
fclose(cl);
tess=3;
tessal(4, 3);
}

/*****/
M18()
{
extern int tess;
FILE *cl;
extern int randisk;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 4, 4\n");
fclose(cl);
tess=4;
tessal(4, 4);
}

```

/* FILE AND SET 2 FACET LEVEL */

/* FILE AND SET 3 FACET LEVEL */

/* FILE AND SET 4 FACET LEVEL */

```

/*****/
M24()
{
extern int tess;
FILE *cl;
extern int randisk;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 4, 5\n");
fclose(cl);
tess=5;
tessal(4, 5);
}
/*****/
M36()
{
extern int tess;
FILE *cl;
extern int randisk;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
tessal(4, 6);
fprintf(cl,"dispat 4, 6\n");
fclose(cl);
tess=6;
}
/*****/
M60()
{
extern int tess;
FILE *cl;
extern int randisk;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
tessal(4, 7);
fprintf(cl,"dispat 4, 7\n");
fclose(cl);
tess=7;
}

```

```

/*****/
mgouraud()
{
extern int shad;
extern int randisk;
FILE *cl;
shad=1;
dispat( 0, 1);
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 0, 1\n");
fclose(cl);
}

/*****/
mconshade()
{
extern int shad;
extern int randisk;
FILE *cl;
shad=0;
dispat(0, 0);
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 0, 0\n");
fclose(cl);
}

/*****/
mtranslu()          /* SET TRANSLUCENCY LEVEL */
{
extern int tess,wrow;
extern double winx1, winx2, winy1, winy2;
extern int vwx1, vwx2, vwyl, vwy2;
int key;
int cnt3;
cnt3=0;
transldisp();
move(60.0, 463.0);
color(32);
rectr(20.0, 15.0);
key=getche();
while(key != QS)
    {

```

```

switch(key)
{
    case RARR :
    case F10 :
        case SIX : linfun(2);
                    color(32);
                    rectr(20.0, 15.0);
                    mover(30.0, 0.0);
                    rectr(20.0, 15.0);
                    cnt3++;
                    if(cnt3>16)
                    {
                        rectr(20.0, 15.0);
                        move(30.0, 463.0);
                        rectr(20.0, 15.0);
                        cnt3=0;
                    }
                    break;
    case LARR :
    case F9 :
    case FOUR : linfun(2);
                 color(32);
                 rectr(20.0, 15.0);
                 mover(-30.0, 0.0);
                 rectr(20.0, 15.0);
                 cnt3--;
                 if(cnt3<0)
                 {
                     rectr(20.0, 15.0);
                     move(540.0, 463.0);
                     rectr(20.0, 15.0);
                     cnt3=8;
                 }
                 break;
    case SPACE: color(32);
                 linfun(2);
                 rectr(20.0, 15.0);
                 goto out;
    case ESC:  linfun(2);
                 cnt3=99;
                 goto out;
    default;;
}
key=getche();
}
out:
mtransl(cnt3);
}

```

```

transldisp()                                /* TRANSLUCENCY MENU DISPLAY */
{
  clbeg(253);
  linfun(2);
  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(3.0, 465.0);
  COLOR(32);
  text("TRANSL");
  MOVE(60.0, 0.0);
  text(" 0 ");
  MOVE(30.0, 0.0);
  text("1");
  MOVE(30.0, 0.0);
  COLOR(255);
  text("2");
  MOVE(30.0, 0.0);
  text(" 3");
  MOVE(30.0, 0.0);
  text("4");
  MOVE(30.0, 0.0);
  text("5");
  MOVE(30.0, 0.0);
  text("6");
  MOVE(30.0,0.0);
  text("7");
  MOVE(30.0,0.0);
  text("8");
  MOVE(30.0, 0.0);
  COLOR(255);
  text("9 ");
  MOVE(30.0, 0.0);
  text(" 10");
  MOVE(30.0, 0.0);
  text("11");
  MOVE(30.0, 0.0);
  text("12");
  MOVE(30.0, 0.0);
  text("13");
  MOVE(30.0,0.0);
  text("14");
  MOVE(30.0,0.0);
  text("15");
  MOVE(30.0,0.0);
  text("EXT");
  clend();
  clrun(253);
}

```

```

/*****

```

```

    mtransl(cnt3)                /* TRANSLUCENCY BRANCH FUNCTIONS */
    int cnt3;
    {
        extern int tess,wdow;
        extern double winx1, winx2, winy1, winy2;
        extern int vxw1, vxw2, vwy1, vwy2;
        char ch3;
        clrwn(253);
        if(cnt3==0)
        {
            trl0();
        }
        if(cnt3==1)
        {
            trl1();
        }
        if(cnt3==2)
        {
            trl2();
        }
        if(cnt3==3)
        {
            trl3();
        }
        if(cnt3==4)
        {
            trl4();
        }
        if(cnt3==5)
        {
            trl5();
        }
        if(cnt3==6)
        {
            trl6();
        }
        if(cnt3==7)
        {
            trl7();
        }
        if(cnt3==8)
        {
            trl8();
        }
    }

```

```

if(cnt3==9)
{
    trl9();
}
if(cnt3==10)
{
    trl10();
}
if(cnt3==11)
{
    trl11();
}
if(cnt3==12)
{
    trl12();
}
if(cnt3==13)
{
    trl13();
}
if(cnt3==14)
{
    trl14();
}
if(cnt3==15)
{
    trl15();
}
wndowset();
}
/*****/
trl0()
{
    extern int tess,randisk;          /*TRANSLUCENCY LEVEL 0 SET */
    FILE *cl;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"dispat 1, 0\n");
    fclose(cl);
    dispat(1, 0);
}
/*****/

```

```

trl1()
{
extern int tess,randisk;
FILE *cl;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 1, 1\n");
fclose(cl);
dispat(1, 1);
}
/*****/
trl2()
{
extern int tess,randisk;
FILE *cl;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 1, 2\n");
fclose(cl);
dispat(1, 2);
}
/*****/
trl3()
{
extern int tess;
FILE *cl;
extern int randisk;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"dispat 1, 3\n");
fclose(cl);
dispat(1, 3);
}
/*****/

```



```

trl4()
{
    /*TRANSLUCENCY LEVEL 4 SET */
    extern int tess;
    FILE *cl;
    extern int randisk;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"dispat 1, 4\n");
    fclose(cl);
    tess=4;
    dispat(1, 4);
}
/*****/
trl5()
{
    /*TRANSLUCENCY LEVEL 5 SET */
    extern int tess;
    FILE *cl;
    extern int randisk;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"dispat 1, 5\n");
    fclose(cl);
    dispat(1, 5);
}
/*****/
trl6()
    /*TRANSLUCENCY LEVEL 6 SET */
{
    extern int tess;
    FILE *cl;
    extern int randisk;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    dispat(1, 6);
    fprintf(cl,"dispat 1, 6\n");
    fclose(cl);
}

```

```

/*****/
    trl7()
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 7);
        fprintf(cl, "dispat 1, 7\n");
        fclose(cl);
    }
/*****/
    trl8()
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 8);
        fprintf(cl, "dispat 1, 8\n");
        fclose(cl);
    }
/*****/
    trl9()
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 9);
        fprintf(cl, "dispat 1, 9\n");
        fclose(cl);
    }

```

```

/*****/
    trl10()
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 10);
        fprintf(cl, "dispat 1,10\n");
        fclose(cl);
    }
/*****/
    trl11()
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 11);
        fprintf(cl, "dispat 1,11\n");
        fclose(cl);
    }
/*****/
    trl12()
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 12);
        fprintf(cl, "dispat 1,12\n");
        fclose(cl);
    }

```

```

/*****/
    trl13()
    {
        extern int tess;                /*TRANSLUCENCY LEVEL 13 SET */
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 13);
        fprintf(cl,"dispat 1,13\n");
        fclose(cl);
    }
/*****/
    trl14()                            /*TRANSLUCENCY LEVEL 14 SET */
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 14);
        fprintf(cl,"dispat 1,14\n");
        fclose(cl);
    }
/*****/
    trl15()                            /*TRANSLUCENCY LEVEL 15 SET */
    {
        extern int tess;
        FILE *cl;
        extern int randisk;
        if(randisk==0)
            cl = fopen("command.d", "a");
        if(randisk==1)
            cl = fopen("d:command.d", "a");
        dispat(1, 15);
        fprintf(cl,"dispat 1,15\n");
        fclose(cl);
    }

```

```

/*****/
ccolor()                                /* SET COLOR FUNCTION */
{
extern int wrow;
extern double winx1, winx2, winy1, winy2;
extern int vx1, vx2, vy1, vy2;
int colnum;
int key;
int cnt;
colnum=0;
cnt=0;
colordisp();
move(70.0, 463.0);
color(32);
rectr(70.0, 15.0);
key=getche();
while(key != QS)
{
switch(key)
{
case SPACE: goto out;
case WS: cnt=0;
goto out;
case BS: cnt=1;
goto out;
case TS: cnt=2;
goto out;
case GS: cnt=3;
goto out;
case BS: cnt=4;
goto out;
case SS: cnt=5;
goto out;
case PS: cnt=6;
goto out;
case RARR :
case F10 :
case SIX: linfun(2);
color(32);
rectr(70.0, 15.0);
mover(70.0, 0.0);
rectr(70.0, 15.0);
cnt++;

```

```

        if(cnt>6)
        {
            rectr(70.0, 15.0);
            move(70.0, 463.0);
            rectr(70.0, 15.0);
            cnt=0;
        }
        break;
case LARR:
case F9 :
case FOUR : linfun(2);
            color(32);
            rectr(70.0, 15.0);
            mover(-70.0, 0.0);
            rectr(70.0, 15.0);
            cnt--;
            if(cnt<0)
            {
                rectr(70.0, 15.0);
                move(490.0, 463.0);
                rectr(70.0, 15.0);
                cnt=6;
            }
            break;
case ESC :
            linfun(2);
            cnt=99;
            goto out;

        default: ;
    }
    key=getche();
}
out:
ccolor2(cnt);
}

```

```

/*****/
colordisp()                /* SET COLOR MENU */
{
  clbeg(252);
  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(2.0, 465.0);
  COLOR(32);
  text("COLOR");
  MOVER(70.0, 0.0);
  color(224);
  text(" WHITE");
  MOVER(70.0, 0.0);
  color(32);
  text(" RED");
  MOVER(70.0, 0.0);
  COLOR(128);
  text("YELLOW");
  MOVER(70.0, 0.0);
  COLOR(64);
  text(" GREEN");
  MOVER(70.0,0.0);
  COLOR(96);
  text(" BLUE");
  MOVER(70.0, 0.0);
  color(218);
  text(" SKY");
  MOVER(70.0, 0.0);
  COLOR(185);
  text(" PURPLE");
  clend();
  clrun(252);

}

/*****/                /* COLOR BRANCH FUNCTIONS */

```

```

ccolor2(cnt)
int cnt;
{
extern int wdown;
extern double winx1, winx2, winy1, winy2;
extern int vx1, vx2, vy1, vy2;
int colnum;
rectr(70.0, 15.0);
clrwn(252);
if(cnt==0)
{
    cwhite();
}
if(cnt==1)
{
    cred();
}
if(cnt==2)
{
    cyellow();
}
if(cnt==3)
{
    cgreen();
}
if(cnt==4)
{
    cblue();
}
if(cnt==5)
{
    csky();
}
if(cnt==6)
{
    cpurple();
}
if(wdown==0)
{
    window(winx1, winx2, winy1, winy2);
    WNPOR(vx1, vx2, vy1, vy2);
}
if(wdown==1)
{
    vwport(0, 319, 0, 239);
}
}

```

```

/*****

```



```

cred()
{
extern int col,randisk;
FILE *cl;
color(28);
col=28;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"color 28 \n");
fclose(cl);
}
/*****/
cgreen()
{
extern int col,randisk;
FILE *cl;
color(59);
col=59;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"color 59 \n");
fclose(cl);
}
/*****/
cblue()
{
extern int col,randisk;
FILE *cl;
color(92);
col=92;
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl,"color 92 \n");
fclose(cl);
}

/*****/

```

```

cyellow()
{
    extern int col,randisk;
    FILE *cl;
    color(128);
    col=128;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"color 128\n");
    fclose(cl);
}
/*****/
cpurple()
{
    extern int col,randisk;
    FILE *cl;
    color(156);
    col=156;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"color 156\n");
    fclose(cl);
}
/*****/
csky()
{
    extern int col,randisk;
    FILE *cl;
    color(188);
    col=188;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"color 188\n");
    fclose(cl);
}
/*****/

```

```

cwhite()                                /* SET AND FILE WHITE */
{
    extern int col,randisk;
    FILE *cl;
    color(224);
    col=224;
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl,"color 224\n");
    fclose(cl);
}
/*****/

```

THESE FUNCTIONS INCLUDE ROTATION MAIN MENU; SCALING FUNCTIONS; ROTATION ORDER FUNCTIONS */

```
#include "std.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
#include "keys.h"
```

/*****/

```
mrotate() /* MAIN ROTATION MENU SELECT */
```

```

{
extern int wdown,menur;
extern double winx1, winx2, winy1, winy2;
extern int vxw1, vxw2, vwy1, vwy2;
char ch2;
int cnt2;
menur=1;
cnt2=0;
clbeg(253);
linfun(2);
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
mrotdisp();
move(75.0,462.0);
color(32);
rectr(75.0, 15.0);
ch2=getche();
while(ch2 != ESC)
{
    switch(ch2)
    {
        case '6' : linfun(2);
                    color(32);
                    rectr(75.0, 15.0);
                    MOVER(75.0, 0.0);
                    rectr(75.0, 15.0);
                    cnt2++;
                    if(cnt2>3)
                    {
                        rectr(75.0, 15.0);
                        MOVER(75.0, 462.0);
                        rectr(75.0, 15.0);
                        cnt2=0;
                    }
                    break;

```

```

case '4' : linfun(2);
           color(32);
           rectr(75.0, 15.0);
           MOVER(-75.0, 0.0);
           rectr(75.0, 15.0);
           cnt2--;
           if(cnt2<0)
           {
               rectr(75.0, 15.0);
               move(300.0, 462.0);
               rectr(75.0, 15.0);
               cnt2=3;
           }
           break;
case SPACE: goto out;
case 'x':cnt2=0;                /* BRANCHES TO X ROTATION MENU */
           goto out;
case 'y':cnt2=1;                /* BRANCHES TO Y ROTATION MENU */
           goto out;
case 'z':cnt2=2;                /* BRANCHES TO Z ROTATION MENU */
           goto out;
case 'c':cnt2=3;
           goto out;
default: ;
    }

```

```

ch2=getche();

```

```

}

```

```

out:
mrotate2(cnt2);
menur=0;
}

```

```

/*****

```

```

mrotdisp()                /* MAIN MENU ROTION DISPLAY */
{
    TSIZE(10.0);
    TJUST(1,1);
    MOVE(5.0, 465.0);
    COLOR(32);
    text(" ROTATE");
    MOVER(75.0, 0.0);
    COLOR(255);
    text("X ROT ");
    MOVER(75.0, 0.0);
    text("Y ROT ");
    MOVER(75.0, 0.0);
    text("Z ROT ");
    MOVER(75.0, 0.0);
    text(" CURS ");
    clend();
    clrwn(253);
}

```

```

: /*****
mrotate2(cnt2)                                /* ROTATION BRANCH TO FUNCTION ROUTINE */
int cnt2;
{
extern int rx,ry,rz,rcntx,rcnty,rcntz,wrow;
extern double winx1, winx2, winy1, winy2;
extern int vwx1, vwx2, vwy1, vwy2;
char ch2;
rectr(75.0, 15.0);
clrun(253);
if(cnt2==0)
{
    filterx();
    mrotx();
}
if(cnt2==1)
{
    filtery();
    mroty();
}
if(cnt2==2)
{
    filterz();
    mrotz();
}
if(cnt2==3)
{
    cursrot();
}
if(wrow==0)
{
    window(winx1, winx2, winy1, winy2);
    WNPORt(vwx1, vwx2, vwy1, vwy2);
}
if(wrow==1)
{
    window(winx1, winx2, winy1, winy2);
    vwport(0, 319, 0, 239);
}
printf("\ncount => %d",cnt2);
}

```

```

/*****
cursrot()                      /* A TEST FUNCTION FOR REAL TIME ROTATION */
{
  int ch2;
  extern double x,y,z,orgx,orgy,orgz;
  extern int rx,rxt, ry, rz, speed,wdow;
  extern double winx1, winx2, winy1, winy2;
  extern int vx1, vx2, vy1, vy2;
  extern int ryt, rzt;
  if(wdow==0)
  {
    window(winx1, winx2, winy1, winy2);
    viewport(vx1, vx2, vy1, vy2);
  }
  if(wdow==1)
  {
    window(winx1, winx2, winy1, winy2);
    viewport(0, 319, 0, 239);
  }
  if(wdow==0)
    cursor();
  if(wdow==1)
    windows();
  mdorg(orgx,orgy,orgz);
  printf("\nORIGIN %.2f,%.2f,%.2f",orgx,orgy,orgz);
  ch2=getche();
  while(ch2 != ESC)
  {
    switch(ch2)
    {
      case '6' :cursor();
                  mdrotx(5);
                  cursor();
                  rxt=rxt+5;
                  printf("CURSOR TEST %d",rxt);
                  break;
      case '4' :cursor();
                  mdrotx(-5);
                  cursor();
                  rxt=rxt-5;
                  printf("CURSOR TEST %d",rxt);
                  break;
      case '8' :cursor();
                  mdroty(5);
                  cursor();
                  ryt=ryt+5;
                  printf("CURSOR TEST %d",rxt);
                  break;
    }
  }
}

```

```

        case '2': cursor();
                    mdroty(-5);
                    cursor();
                    ryt=ryt-5;
                    printf("CURSOR TEST  %d",rxt);
                    break;
        case '0': cursor();
                    mdrotz(5);
                    cursor();
                    rzt=rzt+5;
                    printf("CURSOR TEST  %d",rxt);
                    break;
        case '.': cursor();
                    mdrotz(-5);
                    cursor();
                    rzt=rzt-5;
                    printf("CURSOR TEST  %d",rxt);
                    break;
        default: ;
    }
    printf("\nORIGIN %.2f,%.2f,%.2f",orgx,orgy,orgz);
    ch2=getche();
}
rx=rxt;
ry=ryt;
rz=rzt;
}
/*****
filterx()
{
extern double x,y,z,orgx,orgy,orgz;
extern int rcntx,rcnty, rcntz, rx, ry, rz;
if ((rcntx!=0)&&((rcnty!=0)||!(rcntz!=0)))
{
    mdorg(orgx,orgy,orgz);
    if(rcntx==1)
    {
        if((rcnty==2)&&(rcntz==0))
        {
            mdroty(360-ry);
            mdrotx(360-rx);
            ry=0;
            rx=0;
            rz=0;
            rcnty=0;
            rcntx=1;
            rcntz=0;
        }
    }
}

```



```

if((rcnty==2)&&(rcntz==3))
{
    mdrotz(360-rz);
    mdroty(360-ry);
    mdrotx(360-rx);
    ry=0;
    rx=0;
    rz=0;
    rcnty=0;
    rcntz=0;
    rcntx=1;
}
if((rcntz==2)&&(rcnty==0))
{
    mdrotz(360-rz);
    mdrotx(360-rx);
    ry=0;
    rx=0;
    rz=0;
    rcnty=0;
    rcntz=0;
    rcntx=1;
}
if((rcntz==2)&&(rcnty==3))
{
    mdrotz(360-ry);
    mdrotz(360-rz);
    mdrotx(360-rx);
    ry=0;
    rx=0;
    rz=0;
    rcnty=0;
    rcntz=0;
    rcntx=1;
}
}
if(rcntx==2)
{
    if((rcnty==1)&&(rcntz==0))
    {
        mdrotx(360-rx);
        rx=0;
        rz=0;
        rcnty=1;
        rcntx=2;
    }
}

```

```

        if((rcnty==1)&&(rcntz==3))
        {
            mdrotz(360-rz);
            rz=0;
            mdrotx(360-rx);
            rx=0;
            rcnty=1;
            rcntz=0;
            rcntx=2;
        }
    }
    filterx2();
}
filterx3();
}

/*****/
filterx2()
{
extern double x,y,z;
extern int rcntx,rcnty, rcntz, rx, ry, rz;
if(rcntx==3)
{
    if((rcnty==1)&&(rcntz==2))
    {
        mdrotx(360-rx);
        rx=0;
        rcnty=1;
        rcntz=2;
        rcntx=3;
    }
    if((rcntz==1)&&(rcnty==2))
    {
        mdrotx(360-rx);
        rx=0;
        rcnty=2;
        rcntz=1;
        rcntx=3;
    }
}
}

```

```

/*****/
filterx3()
{
extern double x,y,z;
extern int rcntx,rcnty, rcntz, rx, ry, rz;
if((rcntx==0)&&(ry==0)&&(rz==0))
    rcntx=1;
if((rcnty==1)&&(rcntz==0))
    rcntx=2;
if((rcnty==2) &&(rcntz==1))
    rcntx=3;
if((rcntz==1) &&(rcnty==0))
    rcntx=2;
if((rcntz==2)&&(rcnty==1))
    rcntx=3;
printf("TEST TEST TEST rcntx = %d",rcntx);
}
/*****/
filtery()
{
extern int rcntx,rcnty, rcntz, rx, ry, rz;
extern double x, y, z,orgx,orgy,orgz;
if ((rcnty!=0)&&((rcntx!=0);!(rcntz!=0)))
{
    mdorg(orgx,orgy,orgz);
    if(rcnty==1)
    {
        if((rcntx==2)&&(rcntz==0))
        {
            mdrotx(360-rx);
            mdroty(360-ry);
            ry=0;
            rx=0;
            rcnty=1;
            rcntx=0;
        }
        if((rcntx==2)&&(rcntz==3))
        {
            mdrotz(360-rz);
            mdrotx(360-rx);
            mdroty(360-ry);
            ry=0;
            rx=0;
            rz=0;
            rcnty=1;
            rcntz=0;
            rcntx=0;
        }
    }
}

```

```

if((rcntz==2)&&(rcntx==0))
{
    mdrotz(360-rz);
    mdroty(360-ry);
    rz=0;
    ry=1;
    rcnty=1;
    rcntz=0;
}
if((rcntz==2)&&(rcntx==3))
{
    mdrotx(360-rx);
    mdrotz(360-rz);
    mdroty(360-ry);
    ry=1;
    rx=0;
    rz=0;
    rcnty=1;
    rcntz=0;
    rcntx=0;
}
}
if(rcnty==2)
{
    if((rcntx==1)&&(rcntz==0))
    {
        rcntx=1;
        rcnty=2;
    }
    if((rcntx==1)&&(rcntz==3))
    {
        mdrotz(360-rz);
        rz=0;
        mdroty(360-ry);
        ry=0;
        rcntx=1;
        rcntz=0;
        rcnty=2;
    }
}
    filtery2();
}
    filtery3();

```

```

}
/*****/

```

```

filtery2()
{
extern int rcntx,rcnty, rcntz, rx, ry, rz;
extern double x, y, z;
if(rcnty==3)
{
    if((rcntx==1)&&(rcntz==2))
    {
        ndroty(360-ry);
        ry=0;
        rcntx=1;
        rcntz=2;
        rcnty=3;
    }
    if((rcntz==1)&&(rcntx==2))
    {
        ndroty(360-ry);
        ry=0;
        rcntz=1;
        rcntx=2;
        rcnty=3;
    }
}
}
/*****/
filtery3()
{
extern int rcntx,rcnty, rcntz, rx, ry, rz;
extern double x, y, z;
if((rcnty==0)&&(rz==0)&&(rx==0))
    rcnty=1;
if((rcntx==1)&&(rcntz==0))
    rcnty=2;
if((rcntx==2)&&(rcntz==1))
    rcnty=3;
if((rcntz==1)&&(rcntx==0))
    rcnty=2;
if((rcntz==2)&&(rcntx==1))
    rcnty=3;
}

```

```

/*****/
filterz()
{
extern int rcntx,rcnty, rcntz, rx, ry, rz;
extern double x, y, z,orgx,orgy,orgz;
if ((rcntz!=0)&&((rcntx!=0)|| (rcnty!=0)))
{
mdorg(orgx,orgy,orgz);
if(rcntz==1)
{
if((rcntx==2)&&(rcnty==0))
{
mdrotx(360-rx);
mdrotz(360-rz);
rz=0;
rx=0;
rcntz=1;
rcntx=0;
}
if((rcntx==2)&&(rcnty==3))
{
mdroty(360-ry);
mdrotx(360-rx);
mdrotz(360-rz);
ry=0;
rx=0;
rz=0;
rcnty=0;
rcntz=1;
rcntx=0;
}
if((rcnty==2)&&(rcntx==0))
{
mdroty(360-ry);
mdrotz(360-rz);
rz=0;
ry=0;
rcntz=1;
rcnty=0;
}
}
}

```

```

if((rcnty==2)&&(rcntx==3))
{
    mdrotx(360-rx);
    mdroty(360-ry);
    mdrotz(360-rz);
    ry=0;
    rx=0;
    rz=0;
    rcnty=0;
    rcntz=1;
    rcntx=0;
}
}
if(rcntz==2)
{
    if((rcntx==1)&&(rcnty==0))
    {
        mdrotz(360-rz);
        rz=0;
        rcntx=1;
        rcntz=2;
    }
    if((rcntx==1)&&(rcnty==3))
    {
        mdroty(360-ry);
        ry=0;
        mdrotz(360-rz);
        rz=0;
        rcntx=1;
        rcnty=0;
        rcntz=2;
    }
}
    filters2();
}
    filters3();
}
/*****/

```

```

filterz2()
{
extern int rcntx,rcnty, rcntz, rx, ry, rz;
extern double x, y, z;
if(rcntz==3)
{
if((rcntx==1)&&(rcnty==2))
{
mdrotz(360-rz);
rz=0;
rcntx=1;
rcnty=2;
rcntz=3;
}
if((rcnty==1)&&(rcntx==2))
{
mdrotz(360-rz);
rz=0;
rcnty=1;
rcntx=2;
rcntz=3;
}
}
}

```

```

/*****/

```

```

filterz3()
{
extern int rcntx,rcnty, rcntz, rx, ry, rz;
extern double x, y, z;
if((rcntz==0)&&(ry==0)&&(rx==0))
rcntz=1;
if((rcnty==1)&&(rcntx==0))
rcntz=2;
if((rcnty==2)&&(rcntx==1))
rcntz=3;
if((rcntx==1)&&(rcnty==0))
rcntz=2;
if((rcntx==2)&&(rcnty==1))
rcntz=3;
}

```

```

/*****/

```



```

mscale()
{
extern int wrow;
extern double winx1, winx2, winy1, winy2;
extern double vwx1, vwx2, vwyl, vwyl2;
char ch3;
int cnt3;
cnt3=0;
clbeg(253);
linfun(2);
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
scaledisp();
move(0.0, 415.0);
color(32);
rectr(60.0, 15.0);
ch3=getche();
while(ch3 != 'n')
{
    switch(ch3)
    {
        case '2' : linfun(2);
                    color(32);
                    rectr(60.0, 15.0);
                    mover(0.0, -25.0);
                    rectr(60.0, 15.0);
                    cnt3++;
                    if(cnt3>7)
                    {
                        rectr(60.0, 15.0);
                        move(0.0, 415.0);
                        rectr(60.0, 15.0);
                        cnt3=0;
                    }
                    break;
        case '8' : linfun(2);
                    color(32);
                    rectr(60.0, 15.0);
                    mover(0.0, 25.0);
                    rectr(60.0, 15.0);
                    cnt3--;
                    if(cnt3<0)
                    {
                        rectr(60.0, 15.0);
                        move(0.0, 240.0);
                        rectr(60.0, 15.0);
                        cnt3=7;
                    }
                    break;
    }
}
}

```

```

        default: goto out ;
    }
    printf("count== %d\n",cnt3);
    ch3=getche();
}
out:
mscale2(cnt3);
}
/*****/
scaledisp()
{
    TSIZE(10.0);
    TJUST(1,1);
    MOVE(2.0, 468.0);
    COLOR(255);
    text("SCALE ");
    MOVER(0.0, -50.0);
    text("X Y Z ");
    MOVER(0.0, -25.0);
    text("X AXIS");
    MOVER(0.0, -25.0);
    COLOR(255);
    text("Y AXIS");
    MOVER(0.0, -25.0);
    text("Z AXIS");
    MOVER(0.0, -25.0);
    text("X Y ");
    MOVER(0.0, -25.0);
    text("X Z");
    MOVER(0.0, -25.0);
    text("Y Z");
    MOVER(0.0, -25.0);
    text("ENTER ");
    clend();
    clrun(253);
}
/*****/
mscale2(cnt3)
int cnt3;
{
    extern int wdw;
    extern double winx1, winx2, winy1, winy2;
    extern int vwx1, vwx2, vwyl, vwyl2;
    char ch3;
    rectr(60.0, 15.0);
    clrun(253);
}

```

```

if(cnt3==0)
{
    mscalexyz();
}
if(cnt3==1)
{
    mscalexyz();
}
if(cnt3==2)
{
    mscalex();
}
if(cnt3==3)
{
    mscaley();
}
if(cnt3==4)
{
    mscalez();
}
if(cnt3==5)
{
    mscalexy();
}
if(cnt3==6)
{
    mscalexz();
}
if(cnt3==7)
{
    mscaleyz();
}
if(wdow==0)
{
    window(winx1, winx2, winy1, winy2);
    viewport(vwx1, vwx2, vwy1, vwy2);
}
if(wdow==1)
{
    viewport(0, 319, 0, 239);
}
printf("\ncount => %d",cnt3);
}
/ *****/

```

```

mscalxyz()
{
extern double x,y,z,orgx,orgy,orgz;
extern int randisk;
extern double sx,sy,sz;
float sx1,sy1,sz1;
FILE *cl;
char s;
if(sx!=1.0)
{
mdorg(orgx,orgy,orgz);
mdscal((1.0/sx),(1.0/sx),(1.0/sx));
}
printf("\nENTER SCALE RATIO");
while(scanf("%f",&sx1) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%*s");
}
while(getchar()!='\n');
printf("\nTEST TEST sx = %.2f", sx1);
sy=sx1;
sz=sx1;
sx=sx1;
mdorg(orgx,orgy,orgz);
mdscal(sx,sy,sz);
printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);

if(randisk==0)
cl = fopen("command.d", "a");
if(randisk==1)
cl = fopen("d:command.d", "a");
fprintf(cl, "\nmdorg %.2f, %.2f, %.2f\n",orgx,orgy,orgz);
fprintf(cl, "\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
fclose(cl);
/
}
/*****/

```

```

mscaler()
{
extern int randisk;
FILE *cl;
float sx,sy,sz;
char s;
sx=1.0;
sy=1.0;
sz=1.0;
printf("\nEnter SCALE RATIO");
while(scanf("%f",&sx) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
printf("\nTEST TEST sx = %.2f", sx);
mdscal(sx, sy, sz);
printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);
if(randisk==0)
cl = fopen("command.d", "a");
if(randisk==1)
cl = fopen("d:command.d", "a");
fprintf(cl,"\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
fclose(cl);
}

/*****
mscaley()
{
extern int randisk;
FILE *cl;
float sx,sy,sz;
sx=1.0;
sy=1.0;
sz=1.0;
printf("\nEnter SCALE RATIO");
while(scanf("%f",&sy) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
printf("\nTEST TEST sx = %.2f", sy);
mdscal(sx, sy, sz);
printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);
if(randisk==0)
cl = fopen("command.d", "a");
if(randisk==1)
cl = fopen("d:command.d", "a");
fprintf(cl,"\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
fclose(cl);
}

```

```

/*****/
mscalez()
{
extern int randisk;
FILE *cl;
float  sx,sy,sz;
sx=1.0;
sy=1.0;
sz=1.0;
printf("\nENTER SCALE RATIO");
while(scanf("%f",&sz) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%*s");
}
while(getchar()!='\n');
printf("\nTEST TEST sz = %.2f", sz);
mdscal(sx, sy, sz);
printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);
if(randisk==0)
cl = fopen("command.d", "a");
if(randisk==1)
cl = fopen("d:command.d", "a");
fprintf(cl, "\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
fclose(cl);
}
/*****/
mscalexy()
{
extern int randisk;
FILE *cl;
float  sx,sy,sz;
sx=1.0;
sy=1.0;
sz=1.0;
printf("\nENTER SCALE RATIO");
while(scanf("%f",&sx) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%*s");
}
while(getchar()!='\n');
printf("\nTEST TEST sx = %.2f", sx);
printf("\nENTER SCALE RATIO Y");

```

```

while(scanf("%f",&sy) !=1)
{
    printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
    scanf("%*s");
}
while(getchar()!='\n');
printf("\nTEST TEST sy = %.2f", sy);
printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);
mdscal(sx, sy, sz);
if(randisk==0)
    cl = fopen("command.d", "a");
if(randisk==1)
    cl = fopen("d:command.d", "a");
fprintf(cl, "\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
fclose(cl);
}
/*****/
mscalxsz()
{
    extern int randisk;
    FILE *cl;
    float sx,sy,sz;
    sx=1.0;
    sy=1.0;
    sz=1.0;
    printf("\nENTER SCALE RATIO X");
    while(scanf("%f",&sx) !=1)
    {
        printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
        scanf("%*s");
    }
    while(getchar()!='\n');
    printf("\nTEST TEST sx = %.2f", sx);
    printf("\nENTER SCALE RATIO Z");
    while(scanf("%f",&sz) !=1)
    {
        printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
        scanf("%*s");
    }
    while(getchar()!='\n');
    printf("\nTEST TEST sz = %.2f", sz);
    printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);
    mdscal(sx, sy, sz);
    if(randisk==0)
        cl = fopen("command.d", "a");
    if(randisk==1)
        cl = fopen("d:command.d", "a");
    fprintf(cl, "\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
    fclose(cl);
}

```

```

/*****/
mscaleyz()
{
extern int randisk;
FILE *cl;
float  sx,sy,sz;
sx=1.0;
sy=1.0;
sz=1.0;
printf("\nENTER SCALE RATIO  Z");
while(scanf("%f",&sz) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
printf("\nTEST TEST sz = %.2f", sz);
printf("\nENTER SCALE RATIO  Y");
while(scanf("%f",&sy) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
printf("\nTEST TEST sy = %.2f", sy);
printf("\nsx = %.2f, sy = %.2f, sz =%.2f",sx, sy, sz);
mdscal(sx, sy, sz);
if(randisk==0)
cl = fopen("command.d", "a");
if(randisk==1)
cl = fopen("d:command.d", "a");
fprintf(cl, "\nmdscal %.2f, %.2f, %.2f\n",sx,sy, sz);
fclose(cl);
}
/*****/

```


/*THIS SET OF FUNCTIONS DEALS MAINLY WITH EXECUTING ROTATION FUNCTIONS */

```

#include "std.h"
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
#include <stdio.h>
#include "keys.h"
/*****
mrotx()                                /* BRANCHES TO ROTATE X EXECUTION FUNCTIONS */
{
extern double orgx,orgy,orgz;
extern int rx,rxcntx,wdow;
char ch3;
int cntrx;
int rxd;
mdorg(orgx,orgy,orgz);
if(rx!=0)
{
    rxd=360-rx;
    mdrotx(rxd);
    rx=0;
}
setscreen();
cntrx=0;
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
rxdisp();                                /* DISPLAYS MENU */
move(0.0, 465.0);
color(32);
rectr(32.0, 15.0);
ch3=getche();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);

while(ch3 != 'q')
{
    switch(ch3)
    {
        case 'y' : goto out2;            /* BRANCH TO Y AXIS */
                    break;

        case 'z' : goto out2;            /* BRANCH TO Z AXIS */
                    break;

        case '4' : left();                /* TRANSLATE LEFT */
                    break;
    }
}

```

```

case '6' : right();          /* TRANSLATE RIGHT */
            break;
case CR : linfun(2);          /* DRAWS IMAGE */
            color(32);
            rectr(32.0, 15.0);
            cprim();
            VWPORT(0,639,0,479);
            window(0.0 , 639.0, 0.0, 479.0);
            linfun(2);
            color(32);
            move(0.0, 465.0);
            mover((cntrx*35.0),0.0);
            rectr(32.0, 15.0);
            break;
case '8' : up();              /* TRANSLATE */
            break;
case '2' : down();            /* TRANSLATE */
            break;
case '0' : forward();         /* TRANSLATE */
            break;

case '7' : upleft();          /* TRANSLATE */
            break;

case '3' : downright();       /* TRANSLATE */
            break;

case '.' : back();            /* TRANSLATE */
            break;
case '9' : up();              /* TRANSLATE */
case F10 : linfun(2);
            color(32);
            rectr(32.0, 15.0);
            mover(35.0, 0.0);
            rectr(32.0, 15.0);
            cntrx++;
            if(cntrx>17)
            {
                rectr(32.0, 15.0);
                move(0.0, 465.0);
                rectr(32.0, 15.0);
                cntrx=0;
            }
            rcurs(cntrx);
            break;

```

```

case '1': down();
case F9 : linfun(2);
        color(32);
        rectr(32.0, 15.0);
        mover(-35.0, 0.0);
        rectr(32.0, 15.0);
        cntrx--;
        if(cntrx<0)
        {
            rectr(32.0, 15.0);
            move(595.0, 465.0);
            rectr(32.0, 15.0);
            cntrx=17;
        }
        rcurs2(cntrx);
        break;
case ESC:  cntrx=18;
        color(32);
        linfun(2);
        rectr(32.0, 15.0);
        goto out ;
        default: ;
    }

VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
ch3=getche();
}
out:

rxcntrx=cntrx;
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
mrotx2(cntrx);
out2:
if(ch3=='z')
{
    rxdisp();
    linfun(2);
    move(0.0, 465.0);
    mover((cntrx*35.0),0.0);
    color(32);
    rectr(32.0, 15.0);
    setscrn();
    filterz();
    mrotz();
}
/* TRANSLATE */
/* RETURN TO MAIN */
/* BRANCH TO Z AXIS */

```

```

if(ch3=='y')                                /* BRANCH TO Y AXIS */
{
    rxdisp();
    linfun(2);
    move(0.0, 465.0);
    mover((cntrx*35.0),0.0);
    color(32);
    rectr(32.0, 15.0);
    setscrn();
    filtery();
    mroty();
}

```

```

}
/*****/                                     /* DRAWS CURSOR TO THE SCREEN */
setscreen()

```

```

{
    extern double winx1, winx2, winy1, winy2;
    extern int wdw,vwx1, vwx2, vwyl, vwy2;
    if(wdw==0)
    {
        cursor();
    }
    if(wdw==1)
    {
        windows();
    }
}

```

```

/*****/                                     /* EXECUTES ROTATION ANGLE TO BOARD */
rcurs(cntrx)
int cntrx;

```

```

{
    extern int rx,wdw;
    extern double x,y,z,orgx,orgy,orgz;
    char ch3;

```

```

setscrn();
mdorg(orgx,orgy,orgz);                     /* SETS MODELING ORIGIN */

```

```

if(rx ==360)
    rx=0;                                   /* FILTERS CHOICE THEN EXECUTES COMMAND */

```

```

if(cntrx==0)
{
    rx=0;
}

```

```
if(cntx==1)
{
    rx=0;
}
if(cntx==2)
{
    rx=30;
    mdrotx(30);
}
if(cntx==3)
{
    rx=45;
    mdrotx(15);
}
if(cntx==4)
{
    rx=60;
    mdrotx(15);
}
if(cntx==5)
{
    rx=90;
    mdrotx(30);
}
if(cntx==6)
{
    rx=120;
    mdrotx(30);
}
if(cntx==7)
{
    rx=135;
    mdrotx(15);
}
if(cntx==8)
{
    rx=150;
    mdrotx(15);
}
if(cntx==9)
{
    rx=180;
    mdrotx(30);
}
if(cntx==10)
{
    rx=210;
    mdrotx(30);
}
```

```

if(cntrx==11)
{
    rx=225;
    mdrotx(15);
}
if(cntrx==12)
{
    rx=240;
    mdrotx(15);
}
if(cntrx==13)
{
    rx=270;
    mdrotx(30);
}
if(cntrx==14)
{
    rx=300;
    mdrotx(30);
}
if(cntrx==15)
{
    rx=315;
    mdrotx(15);
}
if(cntrx==16)
{
    rx=330;
    mdrotx(15);
}
if(cntrx==17)
{
    rx=0;;
    if(rx >= 360)
    rx=0;
    mdrotx(30);
}
if(rx >= 360)
    rx=0;
setscreen();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
}
/*****/

```

```

rcurs2(cntrx)                                /* ROTATES NEGATIVE */
int cntrx;
{
extern int wrow ,rx;
extern double x,y,z,orgx,orgy,orgz;
extern int vx1, vx2, vy1, vy2;
char ch3;
setscreen();
mdorg(orgx,orgy,orgz);
if(cntrx==17)
{
    rx=0;
}
if(cntrx==0)
{
    rx=0;
}
if(cntrx==1)
{
    rx=0;
    mdrotx(-30);
}
if(cntrx==2)
{
    rx=30;
    mdrotx(-15);
}
if(cntrx==3)
{
    rx=45;
    mdrotx(-15);
}
if(cntrx==4)
{
    rx=60;
    mdrotx(-30);
}
if(cntrx==5)
{
    rx=90;
    mdrotx(-30);
}
if(cntrx==6)
{
    rx=120;
    mdrotx(-15);
}

```

```
if(cntrx==7)
{
    rx=135;
    mdrotx(-15);
}
if(cntrx==8)
{
    rx=150;
    mdrotx(-30);
}
if(cntrx==9)
{
    rx=180;
    mdrotx(-30);
}
if(cntrx==10)
{
    rx=210;
    mdrotx(-15);
}
if(cntrx==11)
{
    rx=225;
    mdrotx(-15);
}
if(cntrx==12)
{
    rx=240;
    mdrotx(-30);
}
if(cntrx==13)
{
    rx=270;
    mdrotx(-30);
}
if(cntrx==14)
{
    rx=300;
    mdrotx(-15);
}
if(cntrx==15)
{
    rx=315;
    mdrotx(-15);
}
```



```

if(cntrx==16)
{
    rx=330;
    mdrotx(-30);
}
setscreen();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
}
/*****
rxdisp()                                /* X ROTATION MENU DISPLAY */

{
clbeg(253);
linfun(2);

VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
TSIZE(10.0);
TJUST(1,1);
MOVER(1.0, 468.0);
COLOR(210);
text(" X ");
MOVER(35.0, 0.0);
color(96);
text(" 0°");
MOVER(35.0, 0.0);
color(224);
text(" 30°");
MOVER(35.0, 0.0);
color(218);
text(" 45°");
MOVER(35.0, 0.0);
color(224);
text(" 60°");
MOVER(35.0, 0.0);
COLOR(96);
text(" 90°");
color(224);
MOVER(35.0, 0.0);
text("120°");
MOVER(35.0, 0.0);
color(218);
text("135°");
color(224);
MOVER(35.0, 0.0);
text("150°");
MOVER(35.0, 0.0);

```

```

color(96);
text("180");
MOVER(35.0, 0.0);
color(224);
text("210");
MOVER(35.0, 0.0);
color(218);
text("225");
color(224);
MOVER(35.0, 0.0);
text("240");
MOVER(35.0, 0.0);
color(96);
text("270");
MOVER(35.0, 0.0);
color(224);
text("300");
MOVER(35.0, 0.0);
color(218);
text("315");
MOVER(35.0, 0.0);
color(224);
text("330");
color(96);
MOVER(35.0, 0.0);
text("EXIT");
clend();
clrun(253);
}

```

```

/*****

```

```

  mrotx2(cntrx)                                /*ASSIGNS VALUE TO RX "X ROTATION VARIABLE"*/
  int cntrx;
  {
    extern int rx,rcntx,wdow;
    extern double winx1, winx2, winy1, winy2;
    extern double vwx1, vwx2, vwy1, vwy2;
    char ch3;
    int rxd;
    clrun(253);
    if(cntrx==0)
    {
      entervalx();
    }
    if(cntrx==1)
    {
      rx=0;
      rcntx=0;
    }
  }

```

```
if(cntrx==2)
{
    rx=30;
}
if(cntrx==3)
{
    rx=45 ;
}
if(cntrx==4)
{
    rx=60 ;
}
if(cntrx==5)
{
    rx=90 ;
}
if(cntrx==6)
{
    rx=120 ;
}
if(cntrx==7)
{
    rx=135 ;
}
if(cntrx==8)
{
    rx=150 ;
}
if(cntrx==9)
{
    rx=180 ;
}
if(cntrx==10)
{
    rx=210 ;
}
if(cntrx==11)
{
    rx=225 ;
}
if(cntrx==12)
{
    rx=240 ;
}
if(cntrx==13)
{
    rx=270 ;
}
```

```

if(cntrx==14)
{
    rx=300 ;
}
if(cntrx==15)
{
    rx=315 ;
}
if(cntrx==16)
{
    rx=330 ;
}
setscreen();
printf("\ncount => %d",cntrx);
}
/*****
entervalx()                                /* ALLOWS USER TO TYPE IN VALUE */
{
    extern double x,y,z,orgx,orgy,orgz;
    extern int rxt,rcntx;
    extern int wdow,rx,rx2;
    int rxd;
    extern double winx1, winx2, winy1, winy2;
    extern int vx1, vx2, vy1, vy2;
    FILE *cl;
    setscreen();                            /* ERASES CURSOR*/
    if(rx!=0)                               /* RESETS ROTATION VALUE */
    {
        rxd=360-rx;
        mdrotx(rxd);
    }
    printf("\nENTER X ROTATION");           /* DISPLAYS QUERY */
    while(scanf("%d",&rx) !=1)             /* TESTS FOR NUMERIC VALUE */
    {
        printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
        scanf("%s");
    }
    while(getchar()!='\n');                 /* TEST FOR CARRIAGE RETURN */
    if(rx>360)
        rx=0;
    if(rx==360)
        rx=0;
    if(rxt >360)
    {
        rxt=rxt-360;
    }
    if(rx==0)
        rcntx=0;

```

```

mdorg(orgx,orgy,orgz);          /* SETS MODELING ORIGIN */
mdrotx(rx);                     /* EXECUTES ROTATION COMMAND */
rxt=rxt+rx;
    rx2=rx;
setscreen();
}
/*****/
mroty()                          /* SAME PROCEDURE AS MROTX()*/
{
extern int ry,rycnty,wdow;
extern double x,y,z,orgx,orgy,orgz, winx1, winx2, winy1, winy2;
extern int vx1, vx2, vy1, vy2;
char ch3;
int cntry;
int ryd;
mdorg(orgx,orgy,orgz);
if(ry!=0)
{
    ryd=360-ry;
    mdroty(ryd);
    ry=0;
}
setscreen();
cntry=0;
rydisp();
move(0.0, 465.0);
color(32);
rectr(32.0, 15.0);
ch3=getche();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);

while(ch3 != 'q')
{
    switch(ch3)
    {
        case 'x' : goto out2;
                    break;

        case 'z' : goto out2;
                    break;
        case '4' : left();
                    break;
        case '6' : right();
                    break;
    }
}

```

```

case CR : linfun(2);
          color(32);
          rectr(32.0, 15.0);
          cprim();
          VWPORT(0,639,0,479);
          window(0.0 , 639.0, 0.0, 479.0);
          linfun(2);
          color(32);
          move(0.0, 465.0);
          mover((cntry*35.0),0.0);
          rectr(32.0, 15.0);
          break;

case '8' : up();
          break;

case '2' : down();
          break;

case '0' : forward();
          break;

case '7' : upleft();
          break;

case '3' : downright();
          break;

case '.' : back();
          break;

case ESC: cntry=18;
          color(32);
          linfun(2);
          rectr(32.0, 15.0);
          goto out ;

case NINE: up();
case F10 : linfun(2);
          color(32);
          rectr(32.0, 15.0);
          mover(35.0, 0.0);
          rectr(32.0, 15.0);
          cntry++;
          if(cntry>17)
          {
            rectr(32.0, 15.0);
            move(0.0, 465.0);
            rectr(32.0, 15.0);
            cntry=0;
          }

```

```

        rcursy(cntry);
        break;
case ONE: down();
case F9 : linfun(2);
        color(32);
        rectr(32.0, 15.0);
        mover(-35.0, 0.0);
        rectr(32.0, 15.0);
        cntry--;
        if(cntry<0)
        {
            rectr(32.0, 15.0);
            move(595.0, 465.0);
            rectr(32.0, 15.0);
            cntry=17;
        }
        rcursy2(cntry);
        break;
default: ;
    }

VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);

printf("count== %d\n",cntry);
    ch3=getche();
}

out:
rycnty=cntry;
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
mroty2(cntry);
out2:
if(ch3=='z')
{
    rydisp();
    linfun(2);
    move(0.0, 465.0);
    mover((cntry*35.0),0.0);
    color(32);
    rectr(32.0, 15.0);
    setscrn();
    filterz();
    mrotz();
}

```

```

if(ch3=='x')
{
    rydisp();
    linfun(2);
    move(0.0, 465.0);
    mover((centry*35.0),0.0);
    color(32);
    rectr(32.0, 15.0);
    setscrn();
    filterx();
    mrotx();
}

}

/*****/
rcursy(centry)
int centry;
{
    extern int ry,wdow;
    extern double x,y,z,orgx,orgy,orgz,winx1, winx2, winy1, winy2;
    extern int vx1, vx2, vy1, vy2;
    char ch3;
    setscreen();
    mdorg(orgx,orgy,orgz);
    if(ry >=360)
        ry=0;
    if(centry==0)
    {
        ry=0;
    }
    if(centry==1)
    {
        ry=0;
    }
    if(centry==2)
    {
        ry=30;
        mdroty(30);
    }
    if(centry==3)
    {
        ry=45;
        mdroty(15);
    }
    if(centry==4)
    {
        ry=60;
        mdroty(15);
    }
}

```



```
if(cntry==5)
{
    ry=90;
    ndroty(30);
}
if(cntry==6)
{
    ry=120;
    ndroty(30);
}
if(cntry==7)
{
    ry=135;
    ndroty(15);
}
if(cntry==8)
{
    ry=150;
    ndroty(15);
}
if(cntry==9)
{
    ry=180;
    ndroty(30);
}
if(cntry==10)
{
    ry=210;
    ndroty(30);
}
if(cntry==11)
{
    ry=225;
    ndroty(15);
}
if(cntry==12)
{
    ry=240;
    ndroty(15);
}
if(cntry==13)
{
    ry=270;
    ndroty(30);
}
```

```

if(cntry==14)
{
    ry=300;
    mdroty(30);
}
if(cntry==15)
{
    ry=315;
    mdroty(15);
}
if(cntry==16)
{
    ry=330;
    mdroty(15);
}
if(cntry==17)
{
    ry=0;
    mdroty(30);
}
setscreen();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
}
/*****
rcursy2(cntry)
int cntry;
{
    extern int ry,wrow;
    extern double x,y,z,orgx,orgy,orgz,winx1, winx2, winy1, winy2;
    extern int vx1v, vx2v, vy1v, vy2v,vwx1, vwx2, vwy1, vwy2;

    char ch3;
    setscreen();
    mdorg(orgx,orgy,orgz);
    if(cntry==0)
    {
        ry=0;
    }
    if(cntry==1)
    {
        ry=0;
    }

```

```
if(centry==1)
{
    ry=30;
    mdroty(-30);
}
if(centry==2)
{
    ry=45;
    mdroty(-15);
}
if(centry==3)
{
    ry=60;
    mdroty(-15);
}
if(centry==4)
{
    ry=90;
    mdroty(-30);
}
if(centry==5)
{
    ry=120;
    mdroty(-30);
}
if(centry==6)
{
    ry=135;
    mdroty(-15);
}
if(centry==7)
{
    ry=150;
    mdroty(-15);
}
if(centry==8)
{
    ry=180;
    mdroty(-30);
}
if(centry==9)
{
    ry=210;
    mdroty(-30);
}
```

```

if(cntry==10)
{
    ry=225;
    mdroty(-15);
}
if(cntry==11)
{
    ry=240;
    mdroty(-15);
}
if(cntry==12)
{
    ry=270;
    mdroty(-30);
}
if(cntry==13)
{
    ry=300;
    mdroty(-30);
}
if(cntry==14)
{
    ry=315;
    mdroty(-15);
}
if(cntry==15)
{
    ry=330;
    mdroty(-15);
}
if(cntry==16)
{
    ry=0;
    mdroty(-30);
}
setscreen();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
}
/*****/

```

```

rydisp()
{
  clbeg(253);
  linfun(2);
  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(1.0, 468.0);
  COLOR(32);
  text(" Y ");
  MOVER(35.0, 0.0);
  color(96);
  text(" 0");
  MOVER(35.0, 0.0);
  color(224);
  text(" 30");
  MOVER(35.0, 0.0);
  color(218);
  text(" 45");
  MOVER(35.0, 0.0);
  color(224);
  text(" 60 ");
  MOVER(35.0, 0.0);
  COLOR(96);
  text(" 90");
  color(224);
  MOVER(35.0, 0.0);
  text("120");
  MOVER(35.0, 0.0);
  color(218);
  text("135");
  color(224);
  MOVER(35.0, 0.0);
  text("150");
  MOVER(35.0, 0.0);
  color(96);
  text("180");
  MOVER(35.0, 0.0);
  color(224);
  text("210");
  MOVER(35.0, 0.0);
  color(218);
  text("225");
  color(224);
  MOVER(35.0, 0.0);
  text("240");
  MOVER(35.0, 0.0);

```

```

color(96);
text("270");
MOVER(35.0, 0.0);
color(224);
text("300");
MOVER(35.0, 0.0);
color(218);
text("315");
MOVER(35.0, 0.0);
color(224);
text("330");
color(96);
MOVER(35.0, 0.0);
text("EXIT");
clend();
clrun(253);
}
/*****/
mrot2(cntry)
int cntry;
{
extern int ry,rcnty,wdow;
extern double winx1, winx2, winy1, winy2;
extern double vwx1, vwx2, vwy1, vwy2;
char ch3;
int ryd;
clrun(253);
if(cntry==0)
{
entervaly();
}
if(cntry==1)
{
ry=0;
rcnty=0;
}
if(cntry==2)
{
ry=30;
}
if(cntry==3)
{
ry=45 ;
}
if(cntry==4)
{
ry=60 ;
}

```

```
if(cntry==5)
{
    ry=90 ;
}
if(cntry==6)
{
    ry=120 ;
}
if(cntry==7)
{
    ry=135 ;
}
if(cntry==8)
{
    ry=150 ;
}
if(cntry==9)
{
    ry=180 ;
}
if(cntry==10)
{
    ry=210 ;
}
if(cntry==11)
{
    ry=225 ;
}
if(cntry==12)
{
    ry=240 ;
}
if(cntry==13)
{
    ry=270 ;
}
if(cntry==14)
{
    ry=300 ;
}
if(cntry==15)
{
    ry=315 ;
}
if(cntry==16)
{
    ry=330 ;
}
```

```

if(wdow==0)
{
    cursor();
}
if(wdow==1)
{
    windows();
}
printf("\ncount => %d",cnty);
}
/*****
entervaly()
{
extern double x,y,z,orgx,orgy,orgz;
extern int ryt,rcnty;
extern int wdow,ry,ry2;
int ryd;
extern double winx1, winx2, winy1, winy2;
extern int vx1, vx2, vy1, vy2;
FILE *cl;
if(wdow==0)
{
    cursor();
}
if(wdow==1)
{
    windows();
}
if(ry!=0)
{
    ryd=360-ry;
    mdroty(ryd);
}
printf("\nENTER Y ROTATION");
while(scanf("%d",&ry) !=1)
{
    printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
    scanf("%s");
}
while(getchar()!='\n');
printf("\nTEST TEST ry = %d", ry);
if(ry>360)
    ry=0;
if(ry==360)
    ry=0;
if(ryt >360)
{
    ryt=ryt-360;
}

```



```

if(ry==0)
    rcnty=0;
mdorg(orgx,orgy,orgz);
mdroty(ry);
ryt=ryt+ry;
printf("\nryt = %d", ryt);
ry2=ry;
if(wdow==0)
{
    cursor();
}
if(wdow==1)
{
    windows();
}
}
/***** */
/* SAME PROCEDURE AS MROTX() */
mrotz()
{
extern int rz,rzcntz,wdow;
extern double x,y,z,orgx,orgy,orgz, winx1, winx2, winy1, winy2;
extern int vx1, vx2, vy1, vy2;
char ch3;
int cnt3;
int rzd;
mdorg(orgx,orgy,orgz);
cnt3=0;
if(rz!=0)
{
    rzd=360-rz;
    mdrotz(rzd);
    rz=0;
}
setscreen();
rzdisp();
move(0.0, 465.0);
color(32);
rectr(32.0, 15.0);
ch3=getche();
while(ch3 != 'q')
{
    switch(ch3)
    {
        case 'x' : goto out2;
        break;
    }
}

```

```

case 'y' : goto out2;
           break;
case '4' : left();
           break;
case '6' : right();
           break;

case CR : linfun(2);
          color(32);
          rectr(32.0, 15.0);
          cprim();
          VWPORT(0,639,0,479);
          window(0.0 , 639.0, 0.0, 479.0);
          linfun(2);
          color(32);
          move(0.0, 465.0);
          mover((cnt3*35.0),0.0);
          rectr(32.0, 15.0);
          break;
case '8' : up();
           break;
case '2' : down();
           break;
case '0' : forward();
           break;

case '9' : upright();
           break;

case '7' : upleft();
           break;

case '1' : downleft();
           break;

case '3' : downright();
           break;

case '.' : back();
           break;
case ESC: cnt3=18;
          color(32);
          linfun(2);
          rectr(32.0, 15.0);
          goto out ;

```

```

case #10 : linfun(2);
          color(32);
          rectr(32.0, 15.0);
          mover(35.0, 0.0);
          rectr(32.0, 15.0);
          cnt3++;
          if(cnt3>17)
            {
              rectr(32.0, 15.0);
              move(0.0, 465.0);
              rectr(32.0, 15.0);
              cnt3=0;
            }
          rcursz(cnt3);
          break;
case #9 : linfun(2);
          color(32);
          rectr(32.0, 15.0);
          mover(-35.0, 0.0);
          rectr(32.0, 15.0);
          cnt3--;
          if(cnt3<0)
            {
              rectr(32.0, 15.0);
              move(595.0, 465.0);
              rectr(32.0, 15.0);
              cnt3=17;
            }
          rcursz2(cnt3);
          break;
          default: ;
        }
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
ch3=getche();
}

out:
rzcntz=cnt3;
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
mrotz2(cnt3);
out2:

```

```

if(ch3=='y')
{
    rzdisp();
    linfun(2);
    move(0.0, 465.0);
    mover((cnt3*35.0),0.0);
    color(32);
    rectr(32.0, 15.0);
    setscrn();
    filtery();
    mroty();
}
if(ch3=='x')
{
    rzdisp();
    linfun(2);
    move(0.0, 465.0);
    mover((cnt3*35.0),0.0);
    color(32);
    rectr(32.0, 15.0);
    setscrn();
    filterx();
    mrotx();
}

}
/*****/
rcursz(cnt3)
int cnt3;
{
    extern int rz,wdow;
    extern double x,y,z,orgx,orgy,orgz,winx1, winx2, winy1, winy2;
    extern int vx1, vx2, vy1, vy2;
    char ch3;
    setscreen();
    mdorg(orgx,orgy,orgz);
    if(cnt3==0)
    {
        rz=0;
    }
    if(cnt3==1)
    {
        rz=0;
    }
    if(cnt3==2)
    {
        rz=30;
        mdrotz(30);
    }
}

```

```
if(cnt3==3)
{
    rz=45;
    mdrotz(15);
}
if(cnt3==4)
{
    rz=60;
    mdrotz(15);
}
if(cnt3==5)
{
    rz=90;
    mdrotz(30);
}
if(cnt3==6)
{
    rz=120;
    mdrotz(30);
}
if(cnt3==7)
{
    rz=135;
    mdrotz(15);
}
if(cnt3==8)
{
    rz=150;
    mdrotz(15);
}
if(cnt3==9)
{
    rz=180;
    mdrotz(30);
}
if(cnt3==10)
{
    rz=210;
    mdrotz(30);
}
if(cnt3==11)
{
    rz=225;
    mdrotz(15);
}
```

```

if(cnt3==12)
{
    rz=240;
    mdrotz(15);
}
if(cnt3==13)
{
    rz=270;
    mdrotz(30);
}
if(cnt3==14)
{
    rz=300;
    mdrotz(30);
}
if(cnt3==15)
{
    rz=315;
    mdrotz(15);
}
if(cnt3==16)
{
    rz=330;
    mdrotz(15);
}
if(cnt3==17)
{
    rz=0;
    mdrotz(30);
}
setscreen();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
}
/*****/

```

```

rcursz2(cnt3)
int cnt3;
{
extern int wrow,rz;
extern double x,y,z,orgx,orgy,orgz,winx1, winx2, winy1, winy2;
extern int vxw1, vxw2, vwy1, vwy2;
char ch3;
setscreen();
mdorg(orgx,orgy,orgz);
if(cnt3==0)
{
    rz=0;
}
if(cnt3==1)
{
    rz=0;
    mdrotz(-30);
}
if(cnt3==2)
{
    rz=30;
    mdrotz(-15);
}
if(cnt3==3)
{
    rz=45;
    mdrotz(-15);
}
if(cnt3==4)
{
    rz=60;
    mdrotz(-30);
}
if(cnt3==5)
{
    rz=90;
    mdrotz(-30);
}
if(cnt3==6)
{
    rz=120;
    mdrotz(-15);
}
if(cnt3==7)
{
    rz=135;
    mdrotz(-15);
}
}

```

```
if(cnt3==8)
{
    rz=150;
    mdrotz(-30);
}
if(cnt3==9)
{
    rz=180;
    mdrotz(-30);
}
if(cnt3==10)
{
    rz=210;
    mdrotz(-15);
}
if(cnt3==11)
{
    rz=225;
    mdrotz(-15);
}
if(cnt3==12)
{
    rz=240;
    mdrotz(-30);
}
if(cnt3==13)
{
    rz=270;
    mdrotz(-30);
}
if(cnt3==14)
{
    rz=300;
    mdrotz(-15);
}
if(cnt3==15)
{
    rz=315;
    mdrotz(-15);
}
if(cnt3==16)
{
    rz=330;
    mdrotz(-30);
}
if(cnt3==17)
{
    rz=0;
}
```



```

setscreen();
VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);
}
/******/
rzdisp()
{
clbeg(253);
linfun(2);

VWPORT(0,639,0,479);
window(0.0 , 639.0, 0.0, 479.0);

TSIZE(10.0);
TJUST(1,1);
MOVE(1.0, 468.0);
COLOR(32);
text(" Z ");
MOVE(35.0, 0.0);
color(96);
text(" 0");
MOVE(35.0, 0.0);
color(224);
text(" 30");
MOVE(35.0, 0.0);
color(218);
text(" 45");
MOVE(35.0, 0.0);
color(224);
text(" 60 ");
MOVE(35.0, 0.0);
COLOR(96);
text(" 90");
color(224);
MOVE(35.0, 0.0);
text("120");
MOVE(35.0, 0.0);
color(218);
text("135");
color(224);
MOVE(35.0, 0.0);
text("150");
MOVE(35.0, 0.0);
color(96);
text("180");
MOVE(35.0, 0.0);
color(224);
text("210");

```

```

    MOVER(35.0, 0.0);
    color(218);
    text("225");
    color(224);
    MOVER(35.0, 0.0);
    text("240");
    MOVER(35.0, 0.0);
    color(96);
    text("270");
    MOVER(35.0, 0.0);
    color(224);
    text("300");
    MOVER(35.0, 0.0);
    color(218);
    text("315");
    MOVER(35.0, 0.0);
    color(224);
    text("330");
    color(96);
    MOVER(35.0, 0.0);
    text("EXIT");
    clend();
    clrun(253);
}
/*****/
mrotz2(cnt3)
int cnt3;
{
    extern int rz,rcntz,wdow;
    extern double winx1, winx2, winy1, winy2;
    extern double vwx1, vwx2, vwy1, vwy2;
    char ch3;
    int rzd;
    clrun(253);
    if(cnt3==0)
    {
        entervalz();
    }
    if(cnt3==1)
    {
        rz=0;
        rcntz=0;
    }
    if(cnt3==2)
    {
        rz=30;
    }
}

```

```
if(cnt3==3)
{
    rz=45 ;
}
if(cnt3==4)
{
    rz=60 ;
}
if(cnt3==5)
{
    rz=90 ;
}
if(cnt3==6)
{
    rz=120 ;
}
if(cnt3==7)
{
    rz=135 ;
}
if(cnt3==8)
{
    rz=150 ;
}
if(cnt3==9)
{
    rz=180 ;
}
if(cnt3==10)
{
    rz=210 ;
}
if(cnt3==11)
{
    rz=225 ;
}
if(cnt3==12)
{
    rz=240 ;
}
if(cnt3==13)
{
    rz=270 ;
}
if(cnt3==14)
{
    rz=300 ;
}
```

```

if(cnt3==15)
{
    rz=315 ;
}
if(cnt3==16)
{
    rz=330 ;
}
setscreen();
printf("\ncount => %d",cnt3);
}
/*****/
entervalz()
{
extern int rzt,rcntz, vx1, vx2, vwy1, vwy2, wdw, rz, rz2;
int rzd;
extern double orgx,orgy,orgz,winx1, winx2, winy1, winy2,x,y,z;
FILE *cl;
setscreen();
if(rz!=0)
{
    rzd=360-rz;
    mdrotz(rzd);
}
printf("\nENTER Z ROTATION");
while(scanf("%d",&rz) !=1)
{
    printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
    scanf("%s");
}
while(getchar()!='\n');
if(rz>360)
    rz=0;
if(rz==360)
    rz=0;
if(rzt >360)
{
    rzt=rzt-360;
}
if(rz==0)
    rcntz=0;
mdorg(orgx,orgy,orgz);
mdrotz(rz);
rzt=rzt+rz;
    rz2=rz;
setscreen();
}
/*****/

```

```
/*THIS SET OF FUNCTIONS DEAL WITH SOLID MENU ACCESS AND DISPLAY */
```

```
#include "std.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
# define COMMAND "ssolid.bak"
# define COMMENT " "
# define LIM 40
# define LARROW 75
# define RARROW 77
```

```
/******
```

```
*****
```

```
primitive()
{
extern int wrow;
extern int vx1, vx2, vy1, vy2;
extern double wx1, wx2, wy1, wy2;
char ch3;
int cnt3;
cnt3=0;
prindisp();
move(75.0, 463.0);
color(32);
rectr(50.0, 15.0);
ch3=getche();
while(ch3 != 'n')
{
switch(ch3)
{
case 'b': cnt3=0;
goto out;
case 'c': cnt3=1;
goto out;
case 't': cnt3=2;
goto out;
case 's': cnt3=3;
goto out;
case 'w': cnt3=4;
goto out;
case 'p': cnt3=5;
goto out;
case 'r': cnt3=6;
goto out;
```

```

    case 'm': cnt3=9;
                goto out;
    case 'e': cnt3=7;
                goto out;

    case 'l': cnt3=8;
                goto out;
    case 'x':
                goto out;
    case ' ': goto out;
    case '6': linfun(2);
                color(32);
                rectr(50.0, 15.0);
                mover(50.0,0.0);
                rectr(50.0, 15.0);
                cnt3++;
                if(cnt3>10)
                {
                    rectr(50.0, 15.0);
                    move(75.0, 463.0);
                    rectr(50.0, 15.0);
                    cnt3=0;
                }
                break;
    case '4' : linfun(2);
                color(32);
                rectr(50.0, 15.0);
                mover(-50.0, 0.0);
                rectr(50.0, 15.0);
                cnt3--;
                if(cnt3<0)
                {
                    rectr(50.0, 15.0);
                    move(575.0, 463.0);
                    rectr(50.0, 15.0);
                    cnt3=10;
                }
                break;
    default: ;
}
printf("count== %d\n",cnt3);
ch3=getche();
}
out:
primitive2(cnt3);
}

```

```

/*****

```

```

prindisp()
{
  clbeg(253);
  linfun(2);
  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(5.0, 465.0);
  COLOR(32);
  text("SOLIDS");
  MOVER(75.0, 0.0);
  COLOR(255);
  text("BLOC");
  MOVER(50.0, 0.0);
  text("COLM");
  MOVER(50.0, 0.0);
  COLOR(255);
  text("TUBE ");
  MOVER(50.0, 0.0);
  text("SPHR ");
  MOVER(50.0, 0.0);
  text("WEDG");
  MOVER(50.0, 0.0);
  text("PYRY");
  MOVER(50.0, 0.0);
  text("RING");
  MOVER(50.0, 0.0);
  text("RXYR");
  MOVER(50.0, 0.0);
  text("LAYH");
  MOVER(50.0, 0.0);
  text("MUTI");
  COLOR(32);
  MOVER(50.0, 0.0);
  text("EXIT");
  COLOR(255);
  clend();
  clrun(253);
}
/*****/

```

```

primitive2(cnt3)
int cnt3;
{
extern int wdw,curs;
extern int vx1, vx2, vy1, vy2;
extern double winx1, winx2, winy1, winy2;
char ch3;
extern int exton;
exton =0;
rectr(50.0, 15.0);
clrun(253);
if(cnt3==0)
{
    pblock();
}
if(cnt3==1)
{
    pcyylinder();
}
if(cnt3==2)
{
    ptube();
}
if(cnt3==3)
{
    psphere();
}
if(cnt3==4)
{
    pwedge();
}
if(cnt3==5)
{
    pcone();
}
if(cnt3==6)
{
    ptorus();
}
if(cnt3==7)
{
    curs=8;
    mconshade();
    exton=1;
    ssmenu();
}

```



```

if(cnt3==8)
{
    curs=8;
    mconshade();
    exton=0;
    ssmenu();
}

if(cnt3==9)
{
    curs=9;
    many();
}

if(wdow==0)
{
    window(winx1, winx2, winy1, winy2);
    viewport(vwx1, vwx2, vwy1, vwy2);
}
if(wdow==1)
{
    window(-640.0, 640.0, -479.0, 479.0);
    viewport(0, 319, 0, 239);
}
printf("\nEXIT SOLIDS MENU count => %d",cnt3);
}
/*****/

/*****/
pcylinder()
{
    extern int curs;
    curs=2;
    printf("\n CURSOR IS A CYLINDER %d ",curs);
}
/*****/
many()
{
    extern int curs;
    curs=8;
    printf("\n CURSOR IS COMPOSITE %d ",curs);
}

```

```

/*****/
ptube()
{
extern int curs;
curs=3;
printf("\n CURSOR IS A TUBE %d ",curs);
}
/*****/
psphere()
{
extern int curs;
curs=4;
printf("\n CURSOR IS A SPHERE %d ",curs);
}
/*****/
pblock()
{
extern int curs;
curs=1;
printf("\n CURSOR IS A BLOCK %d ",curs);
}
/*****/
ptorus()
{
extern int curs;
curs=5;
printf("\n CURSOR IS A TORUS %d ",curs);
}
/*****/
pcone()
{
extern int curs;
curs=6;
printf("\n CURSOR IS A CONE %d ",curs);
}
/*****/
pwedge()
{
extern int curs;
curs=7;
printf("\n CURSOR IS A WEDGE %d ",curs);
}
/*****/

```

```

/*****/
ssmenu()
{
extern int exton,wrow;
extern int vx1, vx2, vy1, vy2;
extern double winx1, winx2, winy1, winy2;
char ch3;
int cnt3;
cnt3=0;
ssdisp();
move(100.0, 463.0);
color(32);
rectr(100.0, 15.0);
ch3=getche();
while(ch3 != 'n')
{
switch(ch3)
{

case 'r': cnt3=0;
goto out;
case 'c': cnt3=1;
goto out;
case ' ': goto out;
case '6' :
linfun(2);
color(32);
rectr(100.0, 15.0);
mover(100.0,0.0);
rectr(100.0, 15.0);
cnt3++;
if(cnt3>2)
{
rectr(100.0, 15.0);
move(100.0, 463.0);
rectr(100.0, 15.0);
cnt3=0;
}
break;
case '4':
linfun(2);
color(32);
rectr(100.0, 15.0);
mover(-100.0, 0.0);
rectr(100.0, 15.0);
cnt3--;

```

```

        if(cnt3<0)
        {
            rectr(100.0, 15.0);
            move(300.0, 463.0);
            rectr(100.0, 15.0);
            cnt3=2;
        }
        break;
    default: ;
    }
    printf("count== %d\n",cnt3);
    ch3=getche();
}

```

```

out:
ssmenu2(cnt3);
}

```

```

/*****

```

```

ssdisp()
{
    clbeg(253);
    linfun(2);
    VWPORT(0,639,0,479);
    window(0.0 , 639.0, 0.0, 479.0);
    TSIZE(10.0);
    TJUST(1,1);
    MOVE(5.0, 465.0);
    COLOR(32);
    text("LATH");
    MOVER(100.0, 0.0);
    COLOR(255);
    text("RECEIVE");
    MOVER(50.0, 0.0);
    text(" ");
    MOVER(50.0, 0.0);
    COLOR(255);
    text("CREATE ");
    COLOR(32);
    MOVER(100.0, 0.0);
    text("EXIT");
    COLOR(255);
    clend();
    clrun(253);
}

```

```

*****/

```

```

ssmenu2(cnt3)
int cnt3;
{
extern int exton,wow,curs;
extern int firstcall,vwx1, vwx2, vwy1, vwy2;
extern double winx1, winx2, winy1, winy2;
char ch3;
rectr(100.0, 15.0);
clrun(253);
if(wow==0)
{
    window(winx1, winx2, winy1, winy2);
    VMPORT(vwx1, vwx2, vwy1, vwy2);
}
if(wow==1)
{
    window(-640.0, 640.0, -479.0, 479.0);
    viewport(0, 319, 0, 239);
}

if(cnt3==0)
{
    firstcall=0;
    recissol();
}
if(cnt3==1)
{
    moveline();
    curs=8;
    dispi(0);
}

printf("\nEXIT SSOILIDS MENU count => %d",cnt3);
}
/*****/

```

```

/*****
enn()
{
extern int exton,wdow;
extern int vx1, vx2, vy1, vy2;
extern double winx1, winx2, winy1, winy2;
char ch3;
int cnt3;
cnt3=0;
viewdisp();
move(50.0, 463.0);
color(32);
rectr(100.0, 15.0);
ch3=getche();
while(ch3 != 'n')
{
switch(ch3)
{

case 'x': cnt3=1;
goto out;
case 'y': cnt3=0;
goto out;
case 'z': cnt3=2;
goto out;

case 'd': cnt3=3;
goto out;
case 'v': cnt3=4;
goto out;
case 'e': cnt3=6;
goto out;
case 'w': cnt3=5;
goto out;

case ' ': goto out;
case '6' :
linfun(2);
color(32);
rectr(100.0, 15.0);
mover(100.0,0.0);
rectr(100.0, 15.0);
cnt3++;

```

```

        if(cnt3>5)
        {
            rectr(100.0, 15.0);
            move(50.0, 463.0);
            rectr(100.0, 15.0);
            cnt3=0;
        }
        break;
    case '4':
        linfun(2);
        color(32);
        rectr(100.0, 15.0);
        mover(-100.0, 0.0);
        rectr(100.0, 15.0);
        cnt3--;
        if(cnt3<0)
        {
            rectr(100.0, 15.0);
            move(550.0, 463.0);
            rectr(100.0, 15.0);
            cnt3=5;
        }
        break;
    default: ;
}
printf("count== %d\n",cnt3);
ch3=getche();
}
out:
veiw2(cnt3);
}
/*****/

```

```

viewdisp()
{
  clbeg(253);
  linfun(2);
  VMPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(5.0, 465.0);
  COLOR(32);
  text("VIEW ");
  MOVER(50.0, 0.0);
  COLOR(255);
  text("ROTATE ");
  COLOR(32);
  text("      Y");

  MOVER(100.0, 0.0);
  COLOR(255);
  text("ROTATE ");
  COLOR(32);
  text("      X");
  MOVER(100.0, 0.0);
  COLOR(255);
  text("ROTATE ");
  COLOR(32);
  text("      Z");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" INSTANCE");
  COLOR(32);
  text("D      ");
  COLOR(255);
  MOVER(100.0, 0.0);
  text("IEW PT");
  COLOR(32);
  text("V      ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ");
  text("INDOW");
  COLOR(32);
  text("W      ");
  COLOR(255);
  clend();
  clrun(253);
}

```

```

/*****/

```



```

veiw2(cnt3)
int cnt3;
{
extern int exton,wrow,curs;
extern int firstcall,vwx1, vwx2, vwy1, vwy2;
extern double winx1, winx2, winy1, winy2;
char ch3;
rectr(100.0, 15.0);
clrun(253);
if(wrow==0)
{
window(winx1, winx2, winy1, winy2);
VWPORT(vwx1, vwx2, vwy1, vwy2);
}
if(wrow==1)
{
window(-640.0, 640.0, -479.0, 479.0);
vwport(0, 319, 0, 239);
}
if(cnt3==1)
{
cvwx();
}
if(cnt3==0)
{
cvwy();
}
if(cnt3==2)
{
cvwx2();
}
if(cnt3==3)
{
cdistan();
}
if(cnt3==4)
{
cwrpt();
}
if(cnt3==5)
{
cwindow();
}
if(cnt3==6)
{
maindisp();
}
}
/*****/

```

```

/*****

```

```

windowm()

```

```

{

```

```

    extern int exton,wdow;

```

```

    extern int vwx1, vwx2, vwyl, vwy2;

```

```

    extern double winx1, winx2, winyl, winy2;

```

```

    char ch3;

```

```

    int cnt3;

```

```

    cnt3=0;

```

```

    windowdisp();

```

```

    move(100.0, 463.0);

```

```

    color(32);

```

```

    rectr(100.0, 15.0);

```

```

    ch3=getche();

```

```

    while(ch3 != 'n')

```

```

    {

```

```

        switch(ch3)

```

```

        {

```

```

            case 'v': cnt3=0;

```

```

                goto out;

```

```

            case 'p': cnt3=1;

```

```

                goto out;

```

```

            case 'e': cnt3=2;

```

```

                goto out;

```

```

            case 'm': cnt3=3;

```

```

                goto out;

```

```

            case '0': cnt3=6;

```

```

                goto out;

```

```

            case ' ':

```

```

                goto out;

```

```

            case '6' :

```

```

                linfun(2);

```

```

                color(32);

```

```

                rectr(100.0, 15.0);

```

```

                mover(100.0,0.0);

```

```

                rectr(100.0, 15.0);

```

```

                cnt3++;

```

```

                if(cnt3>3)

```

```

                {

```

```

                    rectr(100.0, 15.0);

```

```

                    move(100.0, 463.0);

```

```

                    rectr(100.0, 15.0);

```

```

                    cnt3=0;

```

```

                }

```

```

                break;

```

```

    case '4':

        linfun(2);
        color(32);
        rectr(100.0, 15.0);
        mover(-100.0, 0.0);
        rectr(100.0, 15.0);
        cnt3--;
        if(cnt3<0)
        {
            rectr(100.0, 15.0);
            move(400.0, 463.0);
            rectr(100.0, 15.0);
            cnt3=3;
        }
        break;
    default: ;
}
printf("count== %d\n",cnt3);
ch3=getche();
}

out:
windowm2(cnt3);
}

/*****/

```

```

windowdisp()
{
  clbeg(253);
  linfun(2);
  viewport(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(5.0, 465.0);
  COLOR(32);
  text("WINDOW ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" TRW ");
  COLOR(32);
  text("V");

  MOVER(100.0, 0.0);
  COLOR(255);
  text(" IAW ");
  COLOR(32);
  text("P");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" LEVATION ");
  COLOR(32);
  text("E ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ULTI ");
  COLOR(32);
  text("M ");
  COLOR(255);
  MOVER(100.0, 0.0);
  text(" ");
  COLOR(32);
  text(" ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ");
  text("");
  COLOR(32);
  text(" ");
  COLOR(255);
  clend();
  clrun(253);
}

```

```

/*****/

```

```

windoww2(cnt3)
int cnt3;
{
extern int wdown, planon, exton, wdown, curs;
extern int firstcall, vx1, vx2, vy1, vy2;
extern double zoomval, winx1, winx2, winy1, winy2;
char ch3;
rectr(100.0, 15.0);
clrwn(253);
if(wdown==0)
{
window(winx1, winx2, winy1, winy2);
VWPORT(vx1, vx2, vy1, vy2);
}
if(wdown==1)
{
window(-640.0, 640.0, -479.0, 479.0);
vwport(0, 319, 0, 239);
}

if(cnt3==1)
{
winx1= -284.0*zoomval;
winx2= 284.0*zoomval;
winy1=-213.0*zoomval;
winy2=213.0*zoomval;
wdown=0;
vwport(vx1, vx2, vy1, vy2);
window(winx1, winx2, winy1, winy2);
planon=1;
cplan();
}
if(cnt3==0)
{
winx1= -284.0*zoomval;
winx2= 284.0*zoomval;
winy1=-213.0*zoomval;
winy2=213.0*zoomval;
vwport(vx1, vx2, vy1, vy2);
window(winx1, winx2, winy1, winy2);
wdown=0;
planon=0;
getfile();
}
}

```

```

if(cnt3==2)
{
    winx1= -284.0*zoomval;
    winx2= 284.0*zoomval;
    winy1=-213.0*zoomval;
    winy2=213.0*zoomval;
    viewport(vwx1, vwx2, vwy1, vwy2);
    window(winx1, winx2, winy1, winy2);
    wdraw=0;
    planon=2;
    celevat();

```

```

}
if(cnt3==3)
{
    winx1= -213.0*zoomval;
    winx2= 213.0*zoomval;
    winy1=-213.0*zoomval;
    winy2=213.0*zoomval;
    wdraw =1;
    getfile();

```

```

}

```

```

if(cnt3==4)
{
}
if(cnt3==5)
{
}

```

```

if(cnt3==6)
{
    maindisp();
}

```

```

    printf("\nEXIT VIEW MENU count => %d",cnt3);
}

```

```

/*****/

```

```

/*****/
len()
{
extern int exton,wrow;
extern int vx1, vx2, vy1, vy2;
extern double winx1, winx2, winy1, winy2;
char ch3;
int cnt3;
cnt3=0;
filedisp();
move(100.0, 463.0);
color(32);
rectr(100.0, 15.0);
ch3=getche();
while(ch3 != 'n')
{
switch(ch3)
{

case 'r': cnt3=0;
goto out;
case 's': cnt3=1;
goto out;
case 'e': cnt3=2;
goto out;
case 'm': cnt3=3;
goto out;
case '0': cnt3=6;

goto out;

case ' ':
goto out;
case '6' :
linfun(2);
color(32);
rectr(100.0, 15.0);
mover(100.0,0.0);
rectr(100.0, 15.0);
cnt3++;
if(cnt3>1)
{
rectr(100.0, 15.0);
move(100.0, 463.0);
rectr(100.0, 15.0);
cnt3=0;
}
}
break;
}
}

```

```

        case '4':
            linfun(2);
            color(32);
            rectr(100.0, 15.0);
            mover(-100.0, 0.0);
            rectr(100.0, 15.0);
            cnt3--;
            if(cnt3<0)
            {
                rectr(100.0, 15.0);
                move(200.0, 463.0);
                rectr(100.0, 15.0);
                cnt3=1;
            }
            break;
        default: ;
    }
    printf("count== %d\n",cnt3);
    ch3=getche();
}

out:
    filem2(cnt3);
}

./*****/

```



```

filedisp()
{
  clbeg(253);
  linfun(2);
  VWPORT(0,639,0,479);
  window(0.0 , 639.0, 0.0, 479.0);
  TSIZE(10.0);
  TJUST(1,1);
  MOVE(5.0, 465.0);
  COLOR(32);
  text("FILES ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ECRIVE");
  COLOR(32);
  text("R");

  MOVER(100.0, 0.0);
  COLOR(255);
  text(" END ");
  COLOR(32);
  text("S");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ");
  COLOR(32);
  text(" ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ");
  COLOR(32);
  text(" ");
  COLOR(255);
  MOVER(100.0, 0.0);
  text(" ");
  COLOR(32);
  text(" ");
  MOVER(100.0, 0.0);
  COLOR(255);
  text(" ");
  text("");
  COLOR(32);
  text(" ");
  COLOR(255);
  clend();
  clrun(253);
}

```

```

/*****

```

```

filem2(cnt3)
int cnt3;
{
extern int wdown,planon,exton,wdown,curs;
extern int firstcall,vwx1, vwx2, vwy1, vwy2;
extern double zoomval, winx1, winx2, winy1, winy2;
char ch3;
rectr(100.0, 15.0);
clrwn(253);
if(wdown==0)
{
window(winx1, winx2, winy1, winy2);
VWPORT(vwx1, vwx2, vwy1, vwy2);
}
if(wdown==1)
{
window(-640.0, 640.0, -479.0, 479.0);
vwport(0, 319, 0, 239);
}

if(cnt3==0)
{
recifile();
}
if(cnt3==1)
{
sendfile();
maindisp();
}

if(cnt3==6)
{
maindisp();
}

printf("\nEXIT File (solm.c) MENU count => %d",cnt3);
}
/*****/

```

```

/*****
cistan()
{
extern double returnval;
double num;
static char string10;
extern double dist;
printf("\nGIVE DISTANCE :");
gets(string);
printf(" sl= %s",string);

while(scanf("%d",&num) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%*s");
}
while(getchar()!='\n');
*/
strtoflt(string);

dist=returnval;
printf("\nthis is the Distance => %.1f",dist);
if(dist<450.0)
dist=450.0;
cleari(0);
clears(0);
xyzsend();
mdiden();
setupf();
distan(dist);
getfile();
}
*****/

```

```

cwindow()
{
extern double zoomval,returnval, winx1, winx2, winy1, winy2;
extern int planon;
char string10;
double num;
printf("\n ENTER ZOOM VALUE ");
gets(string);
/*
while(scanf("%f",&num) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%*s");
}
while(getchar()!='\n');
*/
printf("string=%s ",string);
strtofll(string);
zoomval=returnval;
printf("ZOOMVAL %.2f X1=%.2f,X2=%.2f,Y1=%.2f,Y2=%.2f,",
zoomval,winx1,winx2,winy1,winy2);
winx1*= zoomval;
winx2*= zoomval;
winy1*= zoomval;
winy2*=zoomval;
setupf();

if(planon==0)
getfile();
if(planon==1)
cplan();
if(planon==2)
celevat();

}
/*****/

```

```

/*****/
cvwx()
{
extern int vwx ;
int angle;

printf("\nWHAT ANGLE TO ROTATE ON 'X' ?");
while(scanf("%d",&angle) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
if(angle>=360)
angle=0;
vwx=angle;
setupf();
getfile();
}
/*****/
cvwy()
{
extern int vwy;
int angle;
printf("\nWHAT ANGLE TO ROTATE ON 'Y' ?");
while(scanf("%d",&angle) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
if(angle>=360)
angle=0;
vwy=angle;
setupf();
getfile();
}
/*****/

```

```

cvwz()
{
extern int vwz;
int angle;
printf("\nWHAT ANGLE TO ROTATE ON 'Z' ?");
while(scanf("%d",&angle) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%*s");
}
while(getchar()!='\n');
if(angle>=360)
angle=0;

vwz=angle;
setupf();
getfile();

}

```

```

/* THESE FUNCTIONS ARE ASSOCIATED WITH THE LOADED COMMAND LIST COMMAND */
/* THEY ALLOW THE USER TO LOAD A FILE AND VIEW IT FROM THE CLRUN COMMAND*/
#include "std.h"
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
#include <stdio.h>
#include "keys.h"

/*****/
viewrot()          /* SET SELECTABLE VIEWING TRANSFORMATIONS TO THE ADDED COMMAND LIST */
{

extern double vwptx,vwpty,vwptz;
extern int twv,tess,vwx,vwy,vwz,proj;
extern double disy,dist,dish.speed,ltx,lty,ltz;
int ch2;
extern int shade ;
linfun(0);
shade =0;
ch2=getche();
while(ch2 != ESC)
{
    switch(ch2)
    {
        case LS:  lighting();
                  break;

        case YS:  ltx=0.0;          /* SETS LIGHTING "BUGS" */
                  lty=1.0;          /* SETS LIGHTING "BUGS" */
                  ltz=1.0;
                  twv=0;
                  setupvars();
                  if(shade==1)
                      dispat(0,1);
                  clrun(2);
                  break;
    }
}

```

```

case ZS: ltx=0.0;          /* SETS LIGHTING "BUGS" */
        lty=0.0;
        ltz=1.0;
        twv=0;
        setupvars();
        if(shade==1)
            dispat(0,1);
        clrun(2);
        break;

case FS: entess();          /* BRANCHES TO FACTING COMMANDS */
        break;

case ASTRIX:                /* RESETS VIEWING MATRIX */
        vx=0;
        vy=0;
        vz=0;
        setupvars();
        if(shade==1)
            dispat(0,1);
        clrun(2);
        break;

case F5: disy+=speed;        /* SETS YON CLIPPING PLANE */
        twv=0;
        setupvars();
        if(shade==1)
            dispat(0,1);
        clrun(2);
        break;

case F6: disy-=speed;        /* SETS YON CLIPPING PLANE */
        twv=0;
        setupvars();
        if(shade==1)
            dispat(0,1);
        clrun(2);
        break;

case F9: dish+=speed;        /* SETS HITHER CLIPPING PLANE */
        twv=0;
        setupvars();
        if(shade==1)
            dispat(0,1);
        clrun(2);
        break;

```



```

case F10: dish-=speed;          /* SETS HITHER CLIPPING PLANE */
      twv=0;
      setupvars();
      if(shade==1)
        dispat(0,1);
      clrun(2);
      break;

case F7: dist+=speed;           /* SETS VIEWING DISTANCE */
      twv=0;
      setupvars();
      if(shade==1)
        dispat(0,1);
      clrun(2);
      break;

case F8: dist-=speed;           /* SETS VIEWING DISTANCE */
      setupvars();
      if(shade==1)
        dispat(0,1);
      clrun(2);
      break;

case PLUS: speed+=20.0;         /* SETS GRID SPACING DISTANCE */
      break;

case MINUS: speed-=20.0;        /* SETS GRID SPACING DISTANCE */
      break;

case LARR: vwptx-=speed;        /* SETS VIEWING REFERENCE POINT X */
      twv=0;
      setupvars();
      clrun(2);
      break;

case RARR: vwptx+=speed;        /* SETS VIEWING REFERENCE POINT X */
      twv=0;
      setupvars();
      clrun(2);
      break;

case UP: vwpty+=speed;          /* SETS VIEWING REFERENCE POINT Y */
      twv=0;
      setupvars();
      clrun(2);
      break;

```

```

case DOWN: vwpty-=speed;          /* SETS VIEWING REFERENCE POINT Y */
        twv=0;
        setupvars();
        clrwn(2);
        break;
case INS: vwptz+=speed;           /* SETS VIEWING REFERENCE POINT Z */
        twv=0;
        setupvars();
        clrwn(2);
        break;
case DEL: vwptz-=speed;           /* SETS VIEWING REFERENCE POINT Z */
        twv=0;
        setupvars();
        clrwn(2);
        break;

case SIX : vwy+=30;               /* SETS VIEWING REFERENCE ANGLE Y */
        twv=0;
        setupvars();
        clrwn(2);
        break;
case FOUR : vwy-=30;              /* SETS VIEWING REFERENCE ANGLE Y */
        twv=0;
        setupvars();
        clrwn(2);
        break;
case RIGHT : vwz-=30;             /* SETS VIEWING REFERENCE ANGLE X */
        twv=0;
        setupvars();
        clrwn(2);
        break;
case TWO :                        /* SETS VIEWING REFERENCE ANGLE X */
        twv=0;
        vwz+=30;
        setupvars();
        clrwn(2);
        break;
case ZERO :                       /* SETS VIEWING REFERENCE ANGLE Z */
        twv=0;
        vwz-=30;
        setupvars();
        clrwn(2);
        break;
        /* SETS VIEWING REFERENCE ANGLE Z */

```

```

case PERIOD :
    twv=0;
    vwz+=30;
    setupvars();
    clrwn(2);
    break;

case SS :                                /* SETS SHADING ON */
    twv=0;
    setupvars();
    dispat(0,1);
    clrwn(2);

    shade=1;
    dispi(0);
    break;

case WS : dispat(2,1);                  /* SETS WIREFRAME ON */
    shade=0;
    break;

case KS : clrwn(251);                  /* ENTERS INTO MAIN KEYBORED LOOP */
    keyloop();
    break;

case ES:                               /* DISPLAYS ELEVATION VIEWS */
    twv=0;
    vwv+=90;
    vwz=0;
    vwx=0;
    setupvars();
    project(0);
    clrwn(2);
    break;

    default: ;
}
if(shade==1)
    dispi(0);

    twvis(1);
printf("\n DIST = %.1f,    YON = %.1f,    HITHER = %.1f",dist,disy,dish);
printf("\n SPKED = %.1f,  X = %.1f,    Y = %.1f,    Z = %.1f",speed,vwptx,vwpty,vwptz);
    ch2=getche();
}
}
/*****/

```

```

entess()                                /* SET FACETING ANGLE */

{
extern int tess;
extern int faceton;
extern int shade;
int ch;
shade=1;
ch=getche();
    switch(ch)
    {
        case PLUS: faceton=1;
            break;

        case MINUS: faceton=0;
            break;
        case ZERO: tess=0;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrwn(2);
            break;
        case ONE: tess=1;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrwn(2);
            break;
        case TWO: tess=2;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrwn(2);
            break;
        case THREE: tess=3;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrwn(2);
            break;
        case FOUR: tess=4;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrwn(2);
            break;
    }
}

```

```

case FIVE: tess=5;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrun(2);
            break;
case SIX: tess=6;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrun(2);
            break;
case SEVEN: tess=7;
            setupvars();
            if(shade==1)
                dispat(0,1);
            clrun(2);
            break;
default: ;
    )
}

```

/*****/

```
lighting()
{

extern int twv;
extern double speed,ltx,lty,ltx;
extern int shade;
int ch;
printf("\n ENTER ENTER LIGHTING VALUES : ");
shade=1;
ch=getche();
while(ch != ESC)
{

switch(ch)
{
case XS: ltx*=-1;          /* SETS VIEWING REFERENCE POINT X*/
twv=0;
setupvars();
dispat(0,1);
clrwn(2);
break;
case YS: lty*=-1;          /* SETS VIEWING REFERENCE POINT Y*/
twv=0;
setupvars();
dispat(0,1);
clrwn(2);
break;
case ZS: ltx*=-1;          /* SETS VIEWING REFERENCE POINT X*/
twv=0;
setupvars();
dispat(0,1);
clrwn(2);
break;

case LARR: ltx+=speed;      /* SETS VIEWING REFERENCE POINT X*/
twv=0;
setupvars();
dispat(0,1);
clrwn(2);
break;
}
```

```

case BARR: ltx-=speed;          /* SETS VIEWING REFERENCE POINT X*/
           twv=0;
           setupvars();
           dispat(0.1);
           clrun(2);
           break;
case UP: lty-=speed;            /* SETS VIEWING REFERENCE POINT Y*/
           twv=0;
           setupvars();
           clrun(2);
           break;
case DOWN: lty+=speed;          /* SETS VIEWING REFERENCE POINT Y*/
           twv=0;
           setupvars();
           clrun(2);
           break;
case INS: ltz-=speed;           /* SETS VIEWING REFERENCE POINT Z*/
           twv=0;
           setupvars();
           clrun(2);
           break;
case DEL: ltz+=speed;           /* SETS VIEWING REFERENCE POINT Z*/
           twv=0;
           setupvars();
           clrun(2);
           break;
case PLUS: speed+=20.0;         /* SETS GRID SPACING DISTANCE */
           break;
case MINUS: speed-=20.0;        /* SETS GRID SPACING DISTANCE */
           break;

default: ;

} /*END SWITCH */
dispi(0);

tuvis(1);
printf("\nTEST LIGHTING ");
ch=getche();
printf("\n SPEED = %.1f, X = %.1f, Y = %.1f, Z = %.1f",speed,ltx,lty,ltz);
} /* WHILE END */
}/*END*/

,
/*****/

```

```
/*THIS SET OF FUNCTIONS DEAL WITH SENDING AND RECEIVING  COMMAND LISTS */
```

```
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <stdio.h>
#include "std.h"
#include <string.h>
#include <ctype.h>
#include "commands.h"
#define issign(c)      (((c)=='-' || '+') ? (1) : (0))
#define YES 1
#define NO 0
#define MAXLIN 81
#define WHITE 255
#define LIM 40
#define COMMAND "file.b"
#define COMMENT "test"
```

```
***** */
```

```
setupf()          /* THIS FUNCTION CREATES A SETUP COMMAND LIST */
{
extern int vx, vy, vz;
extern int cl, cli, twx0, twx1, twy0, twy1, twe0, twe1 ;
extern int twv, vx1, vx2, vy1, vy2, proj, lten1, lten2;
extern int ltnum, ltmax, displ, disp2, col,randisk;
extern double x,y,z, speed,dist;
extern double amb, ltx, lty,ltz, vwptx, vwpty, vwptz;
extern double winx1,winx2,winyl,winyl,dish,disy;
FILE *s1;
FILE *s2;
if(randisk==0)          /* OPENS SETUP.D */
    s1 = fopen("setup.d", "w");
if(randisk==1)
    s1 = fopen("d:setup.d", "w");
fprintf(s1,"ca \n");
fprintf(s1,"clears %d \n",cl);          /* CLEARS SCREEN */
fprintf(s1,"cleari %d \n",cli);        /* CLEARS BUFFER */
fprintf(s1, "twpos %d, %d, %d, %d, %d \n", twx0, twx1, twy0, twy1, twe0, twe1);
/* SET TEXT WINDOW */
fprintf(s1,"twvis %d \n",twv);          /* SET TEXT WINDOW VISIBLE */
fprintf(s1,"displa 0 \n");              /* SET TO GRAPHICS DISPLAY */
fprintf(s1,"viewport %d, %d, %d, %d \n",vx1,vx2,vy1,vy2); /* SET VIEWPORT */
fprintf(s1,"window %.1f, %.1f, %.1f, %.1f \n", winx1,winx2,winyl,winyl);
/* SET WINDOW */
fprintf(s1,"project %d \n",proj);       /* SET PROJECTION ANGLE */
fprintf(s1,"distan %.1f\n",dist);      /* SET DISTANCE */
fprintf(s1,"disth %.1f\n",dish);       /* SET HITHER CLIPPING*/
fprintf(s1,"distv %.1f\n",disv);       /* SET YON CLIPPING */
fprintf(s1,"cliph 1\n");                /* ENABLE HITHER PLANE */
```



```

fprintf(s1,"clipy 1\n");          /* ENABLE YON PLANE */
fprintf(s1,"liten %d, %d \n",lten1,lten2); /* ENABLE LIGHT SOURCE */
fprintf(s1,"light %d, %.2f, %d, %.1f, %.1f, %.1f\n",ltnum, amb, ltmax, ltx, lty, ltz);
                                     /* SET LIGHT DIRECTION */
                                     /* SET SHADING */

fprintf(s1,"dispat 0 1\n");
fprintf(s1,"dispat 4 5\n");      /* SET FACETING */
fprintf(s1,"dispat 1 0\n");      /* SET SHADING */
fprintf(s1,"vwidth\n");          /* RESET VWINDOW */
fprintf(s1,"vwrtpt %.1f, %.1f, %.1f\n", vwptx, vwpty, vwptz); /* SET VIEW REFERENCE POINT */
fprintf(s1,"vwrotx %d\n",vwrx);  /* SET VWINDOW ROTATION */
fprintf(s1,"vwroty %d\n",vwry);
fprintf(s1,"vwrotz %d\n",vwrz);
fprintf(s1,"lutint 6\n");        /* SET LUT */
fprintf(s1,"color 224\n");
fprintf(s1,"mdiden\n");          /* RESET MODELING MATRIX */
fclose(s1);
}

:*****/
:*****/
readlist()                        /* READS FILE LIST */
{
extern int ramdisk;
static char input*;
FILE *f1;
char ct=0;
displa(1);

if (ramdisk==0)
    if( ( f1 = fopen("filelist.i","r") ) != NULL)
if (ramdisk==1)
    if( ( f1 = fopen("d:filelist.i","r") ) != NULL)

while (fgets( input, LIM, f1)!= NULL)
{
    printf(input);                /* PRINTS STRING FROM FILE */
    ct++;
    if(ct==25)
    {
        ct=0;
        printf("\n\n----- PRESS ANY KEY TO CONTINUE ----- \n\n");
        getche();
    }
}

fclose(f1);                       /* CLOSES FILE */
}

```

```

/*****
sendfile()                      /* SEND COMMAND.D TO FILE */
{
extern int ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6;
extern int randisk;
extern double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
extern char sign;
static char input00;
FILE *f2;
FILE *f1;
FILE *f3;
char name00;
char comm00;
char rsp;
int dex;
int tester;
static char filvar12;
top:
tester=0;
printf("\nWHAT IS THE FILE NAME TO SEND ?");
gets(name);
printf("Addr=%5u char='%c'=%3d\n",          /* TESTS NAME */
        &name00, name00, name00);
if (name00 < 97)
{
    printf("\nBAD NAME TRY AGAIN ?");
    strcpy(name,COMMAND);
    tester=1;
}
if (tester==1)
    goto top;
strcpy(filvar,name);
if(( f3 = fopen(filvar,"r")) !=NULL)        /* CHECKS FOR EXISTING FILE */
{
    fclose(f3);
    printf("%s EXISTS -- do you wish to over write { y/n } ? ",filvar);
    rsp=getche();
    if(rsp == 'y')
    {
        tempc(filvar);
        goto lable;
    }
    else
        goto end;
}

```

```

lable:
sizfvar(filvar);
printf("\nANY COMMENTS ABOUT THE FILE ?");          /* ADDS COMMENTS TO FILE */
gets(comm);
printf("Addr=%5u char='%c'=%3d\n",
      &comm, comm, comm);
if (comm< 97)
{
    strcpy(name.COMMENT);
}
f1 = fopen("filelist.i","a");
fprintf(f1,"%s      %s      \n",filvar,comm);
fclose(f1);
f2 = fopen(filvar,"w");
if(randisk==0)                                       /* OPENS FILE */
    f3 = fopen("command.d","r");
if(randisk==1)
    f3 = fopen("d:command.d","r");
while (fgets(input,LIM,f3)!=NULL)                   /* WRITES COMMAND.D TO FILE */
{
    fprintf(f2,"%s",input);
}
fprintf(f2,"mdiden\n");                             /* CLOSSES FILE */
fclose(f1);
fclose(f2);
end;;
}
/*****/
/*****/
sizfvar(filvar)                                     /* MAKES ALL STRINGS 12 CHARACTERS */
char filvar10;
{
    strcat(filvar,SPACES);
    fit(filvar,12);
}
/*****/
/*****/
sendfap(input)
char input10;
{
    static char str10;
    char string10;
    strcpy(str1,input);
    fit(input,6);
    strcpy(string,&str1);
    tok(string);
    func2(input);
}

```

```

/*****
tempc(filvar)                                /* CREATES A TEMPORARY FILE FOR COMPARING FILE NAMES*/
char filvar40;
{
char *string2=" ";
char *token;
char *token2;
char rsp;
FILE *f4;
FILE *f1;
static char fvar40;
static char str140;
static char *input40;
f4=fopen("filelist.i","r");
f1=fopen("temp.i","w");
while (fgets(inout.LIN,f4)!=NULL)
{
strcpy(str1,input);
strcpy(fvar,filvar);
strcat(fvar,SPACES);
fit(fvar,12);
fit(str1,12);
token =strtok(str1,string2);
token2 =strtok(fvar,string2);
if((strcmp(token,token2) !=0))
fprintf(f1,"%s",input);
printf("\n%s---%s",token,token2);
}
fclose(f4);
fclose(f1);
f4=fopen("filelist.i","w");
f1=fopen("temp.i","r");
while (fgets(inout.LIN,f1)!=NULL)
fprintf(f4,"%s",inout);
fclose(f4);
fclose(f1);
}
*****/

```

```

/*****
    recifile()                /* RECEIVES A FILE FROM THE DISK */
    {
        extern int wdo,ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6;
        extern int randisk;
        extern double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
        extern char sign;
        extern double x,y,z;
        static char input40;
        int rnum;
        FILE *f2;
        FILE *f1;
        FILE *f3;
        static char name12;
        char filvar10;
        rnum=0;

        if(rnum==1)
            goto label;
        printf("\nWHAT IS THE FILE NAME TO RECIEVE ?");
        gets(name);
        cleari(0);
        if((f3= fopen(name,"r"))==NULL)
        {
            printf("\nFILE NOT FOUND:");
            fclose(f3);
            goto end;
        }
        f2 = fopen(name,"r");
        label:
        if(rnum==1)
            f2 =fopen("command.i","r");
        rnum=0;
        if(randisk==0)
            f1 = fopen("command.d","w");
        if(randisk==1)
            f1 = fopen("d:command.d","w");

        while (fgets(input,LIM,f2)!=NULL)    /*THIS WOULD BE QUICKER IF IT WAS PASSED THROUGH IN GETFILE*/
        {
            fprintf(f1,"%s",input);
            printf("\n RECEIVING %s",input);
        }
        fclose(f1);
        fclose(f2);
        reset();
        getfile();
        end:;
    }/* end main */

```

```

/*****
    fit(string,size)
    char string40;
    int size;
    {
    if (strlen(string) > size)
        *(string + size) = '\0';
    }
*****/
/*****
    tok(string1)                /* SEPARATES A STRING OF NUMBERS */
    char string140;
    {
    char *string2 = " ,";
    char *var1,*var2,*var3,*var4,*var5,*var6;
    extern int  ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6;
    extern double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
    extern char sign;
    char *token;
    token = strtok(string1,string2);
    var1=token;
    convert(var1);
    if (sign == 'i')
    {
        ivar1=ivar;
    }
    if (sign == 'f')
    {
        fvar1=fvar;
    }
    token = strtok(NULL,string2);
    var2=token;
    convert(var2);
    if (sign == 'i')
    {
        ivar2=ivar;
    }
    if (sign == 'f')
    {
        fvar2=fvar;
    }
    token = strtok(NULL,string2);
    var3=token;
    convert(var3);
    if (sign == 'i')
    {
        ivar3=ivar;
    }

```

```

if (sign == 'f')
{
    fvar3=fvar;
}
token =strtok(NULL,string2);
var4=token;
convert(var4);
if (sign == 'i')
{
    ivar3=ivar;
}
if (sign == 'f')
{
    fvar4=fvar;
}
token =strtok(NULL,string2);
var5=token;
convert(var5);
if (sign == 'i')
{
    ivar5=ivar;
}
if (sign == 'f')
{
    fvar5=fvar;
}
token =strtok(NULL,string2);
var6=token;
convert(var6);
if (sign == 'i')
{
    ivar6=ivar;
}
if (sign == 'f')
{
    fvar6=fvar;
}
}

```

```

/*****/

```

```

#include <math.h>                /* SEPARATES DIGITS AND CHARACTERS */
#define CHNUM 10
convert(number)
char number?0;
{
extern int ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6,vcount;
extern double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
extern char sign;
char ch;
/* static char numbersizeCHNUM;*/
int value;
double valf;
int digit = YES;
int num =0;
while (( ch= number[num++]) != '\0' && digit == YES)
    if (!isdigit(ch) && !isspace(ch))
        digit =NO;
    if (digit == YES)
        {
            value = atoi(number);
            ivar=value;
            sign = 'i';
        }
    else
        {
            valf =atof(number);

            fvar=valf;
            sign = 'f';
        }
}
./*****/

```



```

/*****/
    strtoint(number)          /* CHANGES A STRING TO FLOATING POINT */
    char number10;
    {
    char ch;
    double value;
    int digit = YES;
    int count = 0;
    static char minus="-";
    static char plus10="+";
    extern double returnval;
    while( (ch = numbercount++) != '\0' && digit == YES)
    if( !isign(ch) && !isdigit(ch) && !isspace(ch) )
        digit = NO;
    if( digit == YES)
    {
        printf("st3 %s", number);
        value= atof(number);
        printf("final %.2f", value);
    }
    else
        printf("\n NOT NUMBER ");
    returnval=value;
    }
/*****/
    strtoint(number)
    char numbers122;
    {
    char ch;
    int value;
    int digit = YES;
    int count = 0;
    extern int returnval1;
    while( (ch = numbercount++) != '\0' && digit == YES)
    if( !isign(ch) && !isdigit(ch) && !isspace(ch) )
        digit = NO;

    if( digit == YES)
    {
        value= atoi(number);
        printf("\nThe number was %.2f", value);
    }
    else
        printf("\n NOT NUMBER ");
    returnval1=value;
    }

```

```
/* THESE FUNCTIONS ARE RELATED TO CURSOR SHAPE AND FILE IMAGE */
```

```
#include "std.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
/*****/
cursor()
{
extern int menur,wdow,col,curs,planon,curson;
extern double x,y,z,orgx,orgy,orgz,orgxd,orgyd,orgzd;
extern double height,width,length;
extern int tess,vwx1, vwx2, vwy1, vwy2;
extern double winx1, winx2, winy1, winy2;
char string0;
window(winx1, winx2, winy1, winy2);      /* WINDOW SETUP */
if(wdow==0)
viewport(vwx1, vwx2, vwy1, vwy2);
color(188);
linfun(2);
dispat(2,0);                             /* WIREFRAME */
if(curs!=0)                               /* TEST NO FOR CROSSHAIR ONLY */
{
mover3(orgxd,orgyd,orgzd);               /* TRANSLATES FROM CENTROID */
MOVER3 (-(5.0), 0.0,0.0);                /* DRAWS CROSS HAIR */
DRAWER3 ((10.0),0.0,0.0);
MOVER3 (-5.0, (-5.0 ), 0.0);
DRAWER3 (0.0,10.0,0.0);
color(88);
MOVER3 (0.0,-5.0,-5.0);
DRAWER3 (0.0,0.0,10.0);
MOVER3 (0.0,0.0,-5.0);
mover3(-orgxd,-orgyd,-orgzd);            /* TRANSLATES TO CENTROID */
}
color(32);
linfun(2);
if(curs==0)                               /* TEST FOR CROSS HAIR ONLY */
{
curs0();
}
if (curs==1)
{
cursblock();                             /* DRAWS BLOCK */
linfun(0);
color(col);
}
```

```

if (curs==2)
{
    color(32);                /* SET XOR ON */
    linfun(2);                /*DRAWS COLUMN */
    column(0, width/2.0, height);
    linfun(0);                /* SET REPLACE DRAWING ON */
    color(col);
}
if (curs==3)
{
    color(32);
    linfun(2);                /* DRAWS TUBE */
    tube(6,length/2.0,width/2.0, height);
    linfun(0);
    color(col);
}
if (curs==4)
{
    color(32);
    linfun(2);
    sphere(2, height/2.0);    /* DRAWS SPHERE */
    linfun(0);
    color(col);
}
if (curs==5)
{
    color(32);
    linfun(2);
    torus(3, length/2.0, width/2.0); /* DRAWS TORUS */
    linfun(0);
    color(col);
}
if (curs==6)
{
    color(32);
    linfun(2);
    cone(1,width/2.0,length, height); /* DRAWS CONE */
    linfun(0);
    color(col);
}
if (curs==7)
{
    color(32);
    linfun(2);
    wedge(5, width, height,length); /* DRAWS WEDGE */
    linfun(0);
    color(col);
}

```

```

if (curs==8)
{
    dispat(4,tess);
    color(32);
    linfun(2);
    mdorg(orgx,orgy,orgz);          /* DRAWS COMMAND LIST 1 */
    clrwn(1);
    linfun(0);
    color(col);
    dispat(4,tess);
}
}
/* ***** */      /* THESE FUNCTION PERFORM THE SAME FUNCTIONS AS CURSOR , BUT CHANGE THE BACK LINES
*/
cursorpl()          /* SO THE XOR MODE WILL NOT CAUSE THEM TO DISAPPEAR */
{
    extern int wdw,col,tess,curs,planon,curson;
    extern double x,y,z;
    extern double height,width,length;
    color(32);
    linfun(2);
    dispat(2,0);
    planon=1;
    if(curs==0)
    {
        curs0();
    }
    if (curs==1)
    {
        cursblock();
        linfun(0);
        color(col);
    }
    if (curs==2)
    {
        dispat(4,tess);
        cone(1, 0.0, width/2.0, height);
        linfun(0);
        color(col);
    }
}

```

```

if (curs==3)
{
    dispat(4,tess);
    cone(1, length/2.0, width/2.0, height);
    linfun(0);
    color(col);
    linfun(0);
    color(col);
}

if (curs==4)
{
    dispat(4,tess);
    cone(1, 0.0, width/2.0, height);
    linfun(0);
    color(col);
}

if (curs==5)
{
    dispat(4,tess);
    torus(3, length/2.0, width/2.0);
    linfun(0);
    color(col);
}

if (curs==6)
{
    dispat(4,tess);
    cone(1,width/2.0,length/2.0, height);
    linfun(0);
    color(col);
}

if (curs==7)
{
    dispat(4,tess);
    wedge(5, width, height,length);
    linfun(0);
    color(col);
}
}

```

```

/*****/ /* THESE FUNCTION PERFORM THE SAME FUNCTIONS AS CURSOR , BUT CHANG THE BACK LINES */
/* SO THE XOR MODE WILL NOT CAUSE THEM TO DISAPPEAR
/*****/
    cursorel()
    {
    extern int wrow,col,tess,curs,planon,curson;
    extern double x,y,z,orgxd,orgyd,orgzd;
    extern double height,width,length;
    color(32);
    linfun(2);
    dispat(2,0);
    dispat(3,1);
    mover3(orgxd,orgyd,orgzd);

    MOVER3 (-5.0, 0.0,0.0);
    DRAW3 ((10.0),0.0,0.0);
    MOVER3 (-5.0, (-5.0 ), 0.0);
    DRAW3 (0.0,10.0,0.0);
    color(88);
    MOVER3 (0.0,-5.0,-5.0);

    DRAW3 (0.0,0.0,10.0);
    MOVER3 (0.0,0.0,-5.0);

    mover3(-orgxd,-orgyd,-orgzd);

    color(32);

    if (curs==1)
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3((-width), 0.0, 0.0);
        drawr3(0.0,height, 0.0);
        drawr3(width,0.0,0.0);
        drawr3(0.0,-height, 0.0);
        mover3(-width/2.0,height/2.0 ,length/2.0);
        linfun(0);
        color(col);
    }

```

```

if (curs==2)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3((-width), 0.0, 0.0);
    drawr3(0.0,height, 0.0);
    drawr3(width,0.0,0.0);
    drawr3(0.0,-height, 0.0);
    mover3(-width/2.0,height/2.0 ,length/2.0);
    linfun(0);
    color(col);
}
if (curs==3)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3((-width), 0.0, 0.0);
    drawr3(0.0,height, 0.0);
    drawr3(width,0.0,0.0);
    drawr3(0.0,-height, 0.0);
    mover3(-width/2.0,height/2.0 ,length/2.0);
    linfun(0);
    color(col);
}
if (curs==4)
{
    dispat(4,tess);
    torus(3,0.0, height/2.0);
    linfun(0);
    color(col);
}
if (curs==5)
{
    dispat(4,tess);
    torus(3, length/2.0, width/2.0);
    linfun(0);
    color(col);
}
if (curs==6)
{
    dispat(4,tess);
    cone(1,width/2.0,length/2.0, height);
    linfun(0);
    color(col);
}

```

```

if (curs==7)
{
    dispat(4,tess);
    wedge(5, width, height,length);
    linfun(0);
    color(col);
}
if (curs==8)
{
    dispat(4,tess);
    clrun(1);
    linfun(0);
    color(col);
}
}
/*****/
curs0()                                /* DRAWS CURSOR */
{
    extern int col,curs,planon;
    extern double x,y,z;
    extern double height,width,length;
    /*
    MOVE3 (-(width/2.0), 0.0,0.0);
    DRAW3 ((width/2.0),0.0,0.0);
    MOVE3 (0.0, (height/2.0), 0.0);
    DRAW3 (0.0,(-(height/2.0)),0.0);
    MOVE3 (0.0, 0.0, length/2.0);
    DRAW3 (0.0, 0.0, -(length/2.0));
    */

    MOVE3 (-(10.0/2.0), 0.0,0.0);
    DRAW3 ((10.0/2.0),0.0,0.0);
    MOVE3 (0.0, (10.0 /2.0), 0.0);
    DRAW3 (0.0,(-(10.0/2.0)),0.0);
    MOVE3 (0.0, 0.0, 10.0/2.0);
    DRAW3 (0.0, 0.0, -(10.0/2.0));

    move3(0.0,0.0,0.0);
    linfun(0);
    color(col);
}

```



```

/*****/                                     /* THESE FUNCTION TRANSLATE THE CURSOR POSITION */

left()
{
extern double orgxd,orgx,orgy,orgz,x,y,z;
extern double orxt,orgyt,orgzt,speed,orgspace;
extern int curs,orgv,wdow,exton;
  setscrn();                                     /* DRAWS CURSOR */
  if(orgv==1)
    orgxd=orgxd-speed;                         /* TRANSLATES MODELING ORIGIN FROM CENTROID*/

  if(orgv==0)
  {
    orgx= orgx-speed;                         /* TRANSLATES MODELING ORIGIN */
    x= x-speed;                             /* DECREASES X VALUE BY GRID SIZE */
    mdtran(- speed, 0.0, 0.0);
  }

  setscrn();                                     /* DRAWS CURSOR */
}
/*****/

right()                                       /* TRANSLATES TO THE RIGHT */
{
extern double orgxd, x,y,z, orgx,orgy,orgz ;
extern double speed;
extern int exton,wdow,orgv;
setscrn();
  if(orgv==1)
    orgxd=orgxd+speed;
  if(orgv==0)
  {
    orgx=orgx+speed;
    x= x+speed;
    mdtran(speed, 0.0, 0.0);
  }
  setscrn();
}
/*****/

```

```
up()                                /* TRANSLATES UP */
{
```

```
extern double x,y,z, orgx,orgy,orgz;
extern double speed;
extern int wdow,orgv;
extern int planon;
extern double orgxd,orgyd,orgzd;
setscrn();
if(planon != 1)
```

```
{
  if(orgv==1)
    orgyd=orgyd+speed;
  if(orgv==0)
  {
    orgy=orgy+speed;
    y+= speed;
    mdtran(0.0,speed,0.0);
  }
  setscrn();
}
```

```
}/*****/
```

```
down()                                /* TRANSLATES DOWN */
{
```

```
extern double x,y,z, orgx,orgy,orgz;
extern double speed;
extern int wdow,orgv;
extern int planon;
extern double orgxd,orgyd,orgzd;
```

```
if(planon != 1)
{
  setscrn();
  if(orgv==1)
    orgyd=orgyd-speed;
  if(orgv==0)
  {
    orgy=orgy-speed;
    y-= speed;
    mdtran(0.0,-speed,0.0);
  }
  setscrn();
}
```

```
}/*****/
```

```

upleft()                                /* TRANSLATES UPLEFT */
{
extern double x,y,z, orgx,orgy,orgz;
extern double speed;
extern int wdow,orgv;
extern int planon;
extern double orgxd,orgyd,orgzd;

if(planon != 1)
{
setscrn();
if(orgv==1)
{
orgyd=orgyd+speed;
orgx=orgx-speed;
}
if(orgv==0)
{
if(orgv==0)
orgy=orgy+speed;
orgx=orgx-speed;
y+=speed;
x-=speed;
mdtran(-speed, speed, 0.0);
}
setscrn();
}
}

./*****/                                /* TRANSLATES UPRIGHT */
upright()
{
extern double x,y,z, orgx,orgy,orgz;
extern double speed;
extern int wdow,orgv;
extern int planon;
extern double orgxd,orgyd,orgzd;

if(planon != 1)
{
setscrn();
if(orgv==1)
{
orgxd=orgxd+speed;
orgyd=orgyd+speed;
}
}
}

```

```

if(orgv==0)
{
    orgy=orgy+speed;
    orgx=orgx+speed;
    y+=speed;
    x+=speed;
    mdtran(speed, speed, 0.0);
}
setscrn();
}

```

```

/*****

```

```

/* TRANSLATES DOWNRIGHT */

```

```

downright()
{
    extern double x,y,z, orgx,orgy,orgz;
    extern double speed;
    extern int wdow,orgv;
    extern int planon;
    extern double orgxd,orgyd,orgzd;

```

```

if(planon != 1)
{
    setscrn();
    if(orgv==1)
    {
        orgxd=orgxd+speed;
        orgyd=orgyd-speed;
    }
    if(orgv==0)
    {
        orgy=orgy-speed;
        orgx=orgx+speed;
        y-=speed;
        x+=speed;
        mdtran(speed,-speed, 0.0);
    }
    setscrn();
}
}

```

```

/*****

```

```

downleft()                                /* TRANSLATES DOWNLEFT */
{
    extern double x,y,z, orgx,orgy,orgz;
    extern double speed;
    extern int wdw,orgv;
    extern int planon;
    extern double orgxd,orgyd,orgzd;

    if(planon != 1)
    {
        setscrn();
        if(orgv==1)
        {
            orgxd=orgxd-speed;
            orgyd=orgyd-speed;
        }
        if(orgv==0)
        {
            orgy=orgy-speed;
            orgx=orgx-speed;
            y-=speed;
            x-=speed;
            mdtran(-speed,-speed, 0.0);
        }
        setscrn();
    }
}

/*****/
forward()                                /* TRANSLATES FORWARD */
{
    extern double x,y,z, orgx,orgy,orgz;
    extern double speed;
    extern int wdw,orgv;
    extern int planon;
    extern double orgxd,orgyd,orgzd;
    if(planon != 2)
    {
        setscrn();
        if(orgv==1)
            orgzd=orgzd+speed;
        if(orgv==0)
        {
            orgz=orgz+speed;
            z= z+speed;
            mdtran(0.0, 0.0,speed);
        }
        setscrn();
    }
}

```

```

/*****/
back()                                /* TRANSLATES FORWARD */
/* TRANSLATES FORWARD */             /* TRANSLATES FORWARD */
{
extern double x,y,z, orgx,orgy,orgz;
extern double speed;
extern int wdog,orgv;
extern int planon;
extern double orgxd,orgyd,orgzd;
    if(planon != 2)
    {
        setscrn();
        if(orgv==1)
            orgzd=orgzd-speed;
        if(orgv==0)
        {
            orgz=orgz-speed;
            z-=speed;
            mdtran(0.0, 0.0,-speed);
        }
        setscrn();
    }
}
/*****/

```

/* THESE FUNCTION WILL DRAW A BLOCK SO THAT THE XOR MODE WILL NOT ERASE BACK LINES */

```
#include "std.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
/*****/
cursblock()
{
extern int wrow,col,curs,planon,cursor,rx,ry,rz;
extern double x,y,z;
extern double height,width,length;
if(planon==0)
{
block(4,width,height,length);
}
if(planon==1||planon==2) /* IF BLOCK IS DRAW IN PLAN OR ELEVATION THE BLOCK MUST BE R
EDRAWN */
{
if(((rx==0)||!(rx==0))&&(rz==0)) /* IF NO ANGLE IS GIVEN */
{
mover3((width/2.0),(-height/2.0),(-length/2.0));
drawr3(0.0,(height),0.0);
drawr3((-width),0.0,0.0);
drawr3(0.0,(-height),0.0);
drawr3(0.0,0.0,length);
drawr3(width,0.0,0.0);
drawr3(0.0,0.0,-length);
mover3(-width/2.0,height/2.0,length/2.0);
}
if(planon==1)
{
cursblockp(); /* IF PLAN IS ON */
}
if(planon==2)
{
cursblocke(); /* IF ELEVATION ON */
}
} /*if planon lor 2 */
}/*main*/
```

```

/*****/
cursblocke()                                /* DRAWS BLOCK IN ELEVATION */
{
extern int wdow,col,curs,planon,curson,rx,ry,rz;
extern double x,y,z;
extern double height,width,length;
    if(rx<90)
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(0.0, (height), 0.0);
        mover3((-width), 0.0, 0.0);
        drawr3(0.0,(-height),0.0);
        mover3((width), 0.0, 0.0);
        mover3((-width), 0.0, 0.0);
        drawr3(0.0, 0.0, length);
        mover3(width,0.0,0.0);
        drawr3(0.0, 0.0, -length);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
    if(rx>270)
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));

        drawr3(0.0, 0.0, length);
        drawr3(0.0, (height), 0.0);
        mover3((-width), 0.0, 0.0);
        drawr3(0.0,(-height),0.0);
        drawr3(0.0, 0.0, -length);
        mover3(width,0.0, 0.0);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
    cursblocke2();
}

/*****/
cursblocke2()
{
extern int wdow,col,curs,planon,curson,rx,ry,rz;
extern double x,y,z;
extern double height,width,length;

    if((rx==90)!='(rx==270))
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(0.0,0.0,(length));
        drawr3(-width,0.0, 0.0);
    }
}

```



```

        drawr3(0.0,0.0,-length);
        drawr3(width,0.0, 0.0);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
if((rx>90)&&(rx<180))
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0, 0.0, length);
    mover3(-width,0.0, 0.0);
    drawr3(0.0, 0.0, -length);
    mover3(0.0, 0.0, length);
    drawr3(0.0,( height),0.0);
    mover3(width,0.0, 0.0);
    drawr3(0.0,( -height),0.0);
    mover3(0.0,0.0, -length);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}
if(rx==180)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0,( height),0.0);
    drawr3(-width,0.0, 0.0);
    drawr3(0.0,(-height),0.0);
    drawr3(width,0.0, 0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}

if((rx>180)&&(rx<270))
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0, (height), 0.0);
    mover3((-width), 0.0, 0.0);
    drawr3(0.0,( -height),0.0);
    drawr3(0.0, 0.0, length);
    mover3(width,0.0, 0.0);
    drawr3(0.0, 0.0, -length);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}

```

```

}

```

```
*****/
```

```
  cursblockp()                      /* DRAWS FOR BLOCK IN PLAN */
```

```
{
```

```
  extern int rcntx,rcntz ;
```

```
  extern int wdw,col,curs,planon,curson,rx,ry,rz;
```

```
  extern double x,y,z;
```

```
  extern double height,width,length;
```

```
  if(rx!=0 && rz==0)
```

```
{
```

```
    cursblockx();                  /* CHECKS FOR X ROTATION */
```

```
}
```

```
  if(rz!=0 && rx==0)
```

```
{
```

```
    cursblockz();                  /* CHECKS FOR Y ROTATION */
```

```
}
```

```
  if(rz!=0 && rx!=0 && ry!=0)
```

```
{
```

```
    cursblockxyz();                /* CHECKS FOR Z ROTATION */
```

```
}
```

```
  if((rz!=0 && rx!=0)&&ry==0)
```

```
{
```

```
    if(rcntx<rcntz)
```

```
        cursblockxz();
```

```
    if(rcntx>rcntz)
```

```
        cursblockzx();
```

```
}
```

```
}
```

```
*****/
```

```
  cursblockx()                      /* EXECUTES X AXIS BLOCK */
```

```
{
```

```
  extern int wdw,col,curs,planon,curson,rx,ry,rz;
```

```
  extern double x,y,z;
```

```
  extern double height,width,length;
```

```
  extern int rcntx,rcntz ;
```

```
  if(rx<90)
```

```
{
```

```
    mover3((width/2.0),(-height/2.0), (-length/2.0));
```

```
    drawr3(0.0, (height), 0.0);
```

```
    mover3((-width), 0.0, 0.0);
```

```
    drawr3(0.0, (-height),0.0);
```

```
    mover3(0.0,( height),0.0);
```

```
    drawr3(0.0, 0.0, length);
```

```
    mover3(width,0.0, 0.0);
```

```
    drawr3(0.0, 0.0, -length);
```

```
    mover3(0.0, (-height),0.0);
```

```

        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
    if((rx==90)|| (rx==270))
    {
        mover3((width/2.0),(- height/2.0), (-length/2.0));
        drawr3(0.0,(height),0.0); drawr3(-width,0.0, 0.0);
        drawr3(0.0,(-height),0.0);
        drawr3(width,0.0, 0.0);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }

```

```

    if((rx>90)&&(rx<180))
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        mover3(0.0, (height), 0.0);
        drawr3(0.0, 0.0, length);
        mover3(-width,0.0,0.0); drawr3(0.0, 0.0, -length);
        mover3(0.0,0.0, length);
        drawr3(0.0,(-height),0.0);
        mover3(width,0.0, 0.0);
        drawr3(0.0,(height),0.0);
        mover3(0.0,-height, -length);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }

```

```

        cursblockxl();

```

```

    }

```

```

/*****/

```

```

cursblockxl()

```

```

{
    extern int wdw,col,curs,planon,curson,rx,ry,rz;
    extern double x,y,z;
    extern double height,width,length;
    extern int rcntx,rcntz ;

```

```

    if(rx==180)
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(0.0,0.0,(length));
        drawr3(-width,0.0, 0.0);
        drawr3(0.0,0.0,-length); drawr3(width,0.0, 0.0);
        mover3(-width/2.0, height/2.0,length/2.0);
        block(4,width,height,length);
    }

```

```

if((rx>180)&&(rx<270))
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0,0.0, length);
    mover3(-width,0.0, 0.0);
    drawr3(0.0, 0.0, -length);
    mover3(0.0, 0.0,length);
    drawr3(0.0,( height),0.0);
    mover3(width,0.0, 0.0);
    drawr3(0.0,(-height),0.0);
    mover3(0.0,0.0, -length);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}

```

```

if(rx>270)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    mover3(0.0,( height),0.0);
    drawr3(0.0, 0.0, length);
    drawr3(0.0, (-height), 0.0);
    mover3((-width), 0.0, 0.0);
    drawr3(0.0,( height),0.0);
    drawr3(0.0, 0.0, -length);
    mover3(width,0.0, 0.0);
    mover3(0.0,( -height),0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}

```

```

}
/*****/

```

```

cursblockz() /* DRAWS BLOCK FOR Z AXIS ROTATION */

```

```

{
    extern int wdow,col,curs,planon,curson,rx,ry,rz;
    extern double x,y,z;
    extern double height,width,length;
    extern int rcntx,rcntz ;
    if (rz < 90 ;| (rz >180)&&(rz< 270))
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(0.0, (height), 0.0);
        drawr3(-width,0.0, 0.0);
        mover3(0.0, 0.0, length);
        drawr3(width,0.0, 0.0);
        drawr3(0.0,( -height),0.0);
        mover3(0.0, 0.0, -length);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
}

```

```

if (rz==90 || rz==270 )
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0, 0.0,length);
    drawr3(0.0,height, 0.0);
    drawr3(0.0, 0.0,-length);
    drawr3(0.0,-height, 0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}

```

```

if ((rz > 90 && rz < 180)|| (rz >270 && rz < 360))
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0, (height), 0.0);
    mover3(0.0, 0.0, length);
    drawr3(0.0, -height, 0.0);
    drawr3(-width,0.0, 0.0);
    mover3(0.0, 0.0,-length);
    drawr3(width,0.0, 0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}

```

```

if(rz==180)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0, (height), 0.0);
    drawr3((-width), 0.0, 0.0);
    drawr3(0.0,( -height),0.0);
    drawr3(0.0, 0.0, length);
    drawr3(width,0.0,0.0);
    drawr3(0.0, 0.0, -length);
    mover3(-width/2.0,height/2.0 ,length/2.0);
}

```

```

}

```

```

/*****/

```

```

cursblockz()          /* TEST FOR A COMBINATION OF X Z ROTATION */

```

```

{
    extern int wrow,col,curs,planon,curson,rx,ry,rz;
    extern double x,y,z;
    extern double height,width,length;
    extern int rcntx,rcntz ;

```

```

if( ((rz<90) ||( rz>90&& rz<180) || (rz>180 && rz<270)|| (rz>270))&&((rx<90) ||( rx>90&& rx<180) || (r
x>180 &&rx<270)|| (rx>270)) )
{
    block(4,width,height,length);
}

```

```

if( ((rx<90) || (rx>90&& rx<180) || (rx>180 &&rx<270) || (rx>270))&& (rz==90 || rz==270))
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0,height, 0.0);
    drawr3(0.0,0.0,length);
    drawr3(0.0,-height, 0.0);
    drawr3(0.0,0.0,-length);
    mover3(-width/2.0, height/2.0, length/2.0);
}
if( (rx>90 || (rx>180 &&rx<270))&& rz==180)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(-width, 0.0, 0.0);
    drawr3(0.0,height,length);
    drawr3(width, 0.0, 0.0);
    drawr3(0.0,-height,-length);
    mover3(0.0,height,0.0);
    drawr3(-width,0.0,0.0);
    mover3(0.0,-height,length);
    drawr3(width,0.0,0.0);
    mover3(0.0,0.0,-length);
    mover3(-width/2.0, height/2.0, length/2.0);
}
if(((rx>90&&rx<180) || rx>270)&& rz==180)
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    mover3(0.0,height,0.0);
    drawr3(-width, 0.0, 0.0);
    drawr3(0.0,-height,length);
    drawr3(width, 0.0, 0.0);
    drawr3(0.0,height,-length);
    mover3(0.0,-height,0.0);
    drawr3(-width,0.0,0.0);
    mover3(0.0,height,length);
    drawr3(width,0.0,0.0);
    mover3(0.0,-height,-length);
    mover3(-width/2.0, height/2.0, length/2.0);
}
cursblockxz2();
}

```

```

/*****/
cursblockxz2()
{
extern int wrow,col,curs,planon,curson,rx,ry,rz;
extern double x,y,z;
extern double height,width,length;
extern int rcntx,rcntz ;
if((rx==90;'rx==270) && (rz==90 ;' rz==270 ))
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
drawr3(0.0,height, 0.0);
drawr3(0.0,0.0,length);
drawr3(0.0,-height, 0.0);
drawr3(0.0,0.0,-length);
mover3(-width/2.0, height/2.0, length/2.0);
}
if((rx==90;'rx==270) && (rz==180 ))
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
drawr3(0.0,height,0.0);
drawr3(-width,0.0,0.0);
drawr3(0.0,-height, 0.0);
drawr3(width,0.0,0.0);
mover3(-width/2.0, height/2.0, length/2.0);
}
if((rx==90) && ((rz>90 && rz<180);' (rz>270 && rz<360 )))
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
mover3(0.0,height, 0.0);
drawr3(-width,0.0,length);
mover3(0.0,-height, 0.0);
drawr3(width,0.0, -length);
mover3(-width/2.0, height/2.0, length/2.0);
block(4,width,height,length);
}
if((rx==90) && (rz<90 ;' (rz>180 && rz<270)))
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
mover3(0.0,height, 0.0);
drawr3(-width,0.0,0.0);
mover3(0.0,-height, 0.0);
drawr3(width,0.0, 0.0);
drawr3(0.0,0.0,length);
mover3(0.0,height, 0.0);
drawr3(0.0,0.0,-length);
mover3(0.0,-height, 0.0);
mover3(-width/2.0, height/2.0, length/2.0);
block(4,width,height,length);
}
cursblockxz3();
}

```

```

/*****
cursblockxz3()
{
    extern int wrow,col,curs,planon,curson,rx,ry,rz;
    extern double x,y,z;
    extern double height,width,length;
    extern int rcntx,rcntz ;
    if((rx==180) && (rz<90 ;| (rz>180 && rz<270)))
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        mover3(0.0,height,0.0);
        drawr3(-width,-height,0.0);
        mover3(0.0,0.0,length);
        drawr3(width,height,0.0);
        mover3(0.0,-height,-length);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
    if ( rx==180 && ((rz>90 && rz<180) ;| rz>270) )
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(-width,height,0.0);
        mover3(0.0,0.0,length);
        drawr3(width,-height,0.0);
        mover3(0.0,0.0,-length);
        mover3(-width/2.0, height/2.0, length/2.0);
        block(4,width,height,length);
    }
    if(rx==180 && (rz==90 ;| rz==270))
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(0.0,height,0.0);
        drawr3(0.0,0.0,length);
        drawr3(0.0,-height,0.0);
        drawr3(0.0,0.0,-length);
        mover3(-width/2.0, height/2.0, length/2.0);
    }
    if(rx==180 && rz==180)
    {
        mover3((width/2.0),(-height/2.0), (-length/2.0));
        drawr3(-width,0.0,0.0);
        drawr3(0.0,0.0,length);
        drawr3(width,0.0,0.0);
        drawr3(0.0,0.0,-length);
        mover3(-width/2.0, height/2.0, length/2.0);
    }
    cursblockxz4();
}

```



```

/*****
  cursblockrz4()
  {
    extern int wrow,col,curs,planon,cursor.rx,ry,rz;
    extern double x,y,z;
    extern double height,width,length;
    extern int rcntx,rcntz ;
    if((rx==270) && ((rz>90 && rz<180)!! (rz>270 && rz<360 )))
    {
      mover3((width/2.0),(-height/2.0), (-length/2.0));
      mover3(0.0,height, 0.0);
      drawr3(-width,0.0,0.0);
      mover3(0.0,-height, 0.0);
      drawr3(width,0.0, 0.0);
      drawr3(0.0,0.0,length);
      mover3(0.0,height, 0.0);
      drawr3(0.0,0.0,-length);
      mover3(0.0,-height, 0.0);
      mover3(-width/2.0, height/2.0, length/2.0);
      block(4,width,height,length);
    }
    if((rx==270) && (rz<90 !! (rz>180 && rz<270)))
    {
      mover3((width/2.0),(-height/2.0), (-length/2.0));
      mover3(0.0,height, 0.0);
      drawr3(-width,0.0,length);
      mover3(0.0,-height, 0.0);
      drawr3(width,0.0, -length);
      mover3(-width/2.0, height/2.0, length/2.0);
      block(4,width,height,length);
    }
  }
*****/

```

```

cursblockzx()          /* FILTER FOR Z X ROTATION */
{
  extern int wrow,col,curs,planon,cursor,rx,ry,rz;
  extern double x,y,z;
  extern double height,width,length;
  extern int rcntx,rcntz ;
  if( ((rz<90) ;!( rz>90&& rz<180) ;! (rz>180 && rz<270);!(rz>270))&&((rx<90) ;!( rx>90&& rx<180) ;! (r
    x>180 &&rx<270);! (rx>270)) )
  {
    block(4,width,height,length);
  }
  if( ((rz<90) ;! (rz>90&& rz<180) ;! (rz>180 &&rz<270) ;!(rz>270))&& (rx==90 ;! rx==270))
  {
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0,height, 0.0);
    drawr3(-width,0.0,0.0);
    drawr3(0.0,-height, 0.0);
    drawr3(width,0.0,0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
  }
  if(( rz<90 ;! (rz>180 && rz<270)) && rx==180)
  {
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(-width,height, 0.0);
    drawr3(0.0,0.0,length);
    drawr3(width,-height,0.0);
    drawr3(0.0,0.0,-length);
    mover3(-width/2.0, height/2.0, length/2.0);
  }
  if( (rz>90&& (rz<180 ;! rz>270)) && rx==180)
  {
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    mover3(0.0, height, 0.0);
    drawr3(-width,-height, 0.0);
    drawr3(0.0,0.0,length);
    drawr3(width,height,0.0);
    drawr3(0.0,0.0,-length);
    mover3(0.0, -height, 0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
  }
  cursblockzx90();
}
/*****/

```

```

cursblockzx90()      /* FILTER FOR Z X=90 ROTATION */
{
extern int wdw, col, curs, planon, curson, rx, ry, rz;
extern double x, y, z;
extern double height, width, length;
extern int rcntx, rcntz ;
if(rz==90&&(rx<90;!(rx>180&&rx<270)) )
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    mover3(-width,0.0, 0.0);
    drawr3(width, 0.0,length);
    mover3(0.0,height,0.0);
    drawr3(-width,0.0,-length);
    mover3(width, -height, 0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}
if(rz==90&&((rx>90&&rx<180);!( rx>270)) )
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(-width, 0.0,length);
    mover3(0.0,height,0.0);
    drawr3(width,0.0,-length);
    mover3(0.0, -height, 0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
    block(4,width,height,length);
}
if((rz==90;!rz==270)&&(rx==90;!rx==270) )
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(-width, 0.0,0.0);
    drawr3(0.0,height,0.0);
    drawr3(width, 0.0,0.0);
    drawr3(0.0,-height,0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
}
if((rz==90;!rz==270)&&(rx==180) )
{
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3(0.0,0.0,length);
    drawr3(0.0,height,0.0);
    drawr3(0.0,0.0,-length);
    drawr3(0.0,-height,0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
}
cursblockzx270();
}
/*****/

```

```

cursblockzx270()
{
extern int wdw, col, curs, planon, curson, rx, ry, rz;
extern double x, y, z;
extern double height, width, length;
extern int rcntx, rcntz ;
if(rz==270&&(rx>90&&rx<180);!( rx>270)) )
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
mover3(-width,0.0, 0.0);
drawr3(width, 0.0,length);
mover3(0.0,height,0.0);
drawr3(-width,0.0,-length);
mover3(width, -height, 0.0);
mover3(-width/2.0, height/2.0, length/2.0);
block(4,width,height,length);
}
if(rz==270&&(rx<90;!(rx>180&&rx<270)) )
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
drawr3(-width, 0.0,length);
mover3(0.0,height,0.0);
drawr3(width,0.0,-length);
mover3(0.0, -height, 0.0);
mover3(-width/2.0, height/2.0, length/2.0);
block(4,width,height,length);
}
cursblockzx180();
}
/*****/
cursblockzx180()
{
extern int wdw, col, curs, planon, curson, rx, ry, rz;
extern double x, y, z;
extern double height, width, length;
extern int rcntx, rcntz ;
if(rz==180&&(rx>90;!(rx>180&&rx<270)) )
{
mover3((width/2.0),(-height/2.0), (-length/2.0));
mover3(0.0,height,0.0);
drawr3( 0.0,-height,length);
mover3(-width,0.0,0.0);
drawr3( 0.0,height,-length);
mover3(width,-height,0.0,0.0);
mover3(-width/2.0, height/2.0, length/2.0);
block(4,width,height,length);
}
}

```

```

if(rz==180&&((rx>90&&rx<180);!( rx>270)) )
{
  mover3((width/2.0),(-height/2.0), (-length/2.0));
  drawr3( 0.0,height,length);
  mover3(-width,0.0,0.0);
  drawr3( 0.0,-height,-length);
  mover3(width,0.0,0.0);
  mover3(-width/2.0, height/2.0, length/2.0);
  block(4,width,height,length);
}
if(rz==180&&(rx==90;rx==270) )
{
  mover3((width/2.0),(-height/2.0), (-length/2.0));
  drawr3( 0.0,height,0.0);
  drawr3(-width,0.0,0.0);
  drawr3( 0.0,-height,0.0);
  drawr3(width,0.0,0.0);
  mover3(-width/2.0, height/2.0, length/2.0);
}
if(rz==180&&rx==180 )
{
  mover3((width/2.0),(-height/2.0), (-length/2.0));
  drawr3( -width,0.0,0.0);
  drawr3(0.0,0.0,length);
  drawr3(width, 0.0,0.0);
  drawr3(0.0,0.0,-length);
  mover3(-width/2.0, height/2.0, length/2.0);
}
}

```

```

/*****/

```

```

cursblockxyz()    /* IF X Y Z ROTATION SET */
{
  extern int wrow,col,curs,planon,curson,rx,ry,rz;
  extern double x,y,z;
  extern double height,width,length;
  extern int rcntx,rcntz ;
  if((rx==90;||rx==270)&&rz==180 )
  {
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3( 0.0,height,0.0);
    drawr3(-width,0.0,0.0);
    drawr3( 0.0,-height,0.0);
    drawr3(width,0.0,0.0);
    mover3(-width/2.0, height/2.0, length/2.0);
  }
  if((rx==90;||rx==270)&&(rz==90;||rz==270))
  {
    mover3((width/2.0),(-height/2.0), (-length/2.0));
    drawr3( -width,height,0.0);
    mover3(0.0,0.0,length);
    drawr3(width,-height,0.0);
    mover3( 0.0,0.0,-length);
    mover3(-width/2.0, height/2.0, length/2.0);
  }
  block(4,width,height,length);
}

```

```

./*****/

```

* THESE FUNCTION CREATE SWEEP SOLIDS */

```
#include "std.h"
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
#include <stdio.h>
#include "keys.h"
#define SSOLID "ssolid"
#define COMMAND "ssolid.bak"
#define COMMENT " "
#define LIM 40
```

./*****/

noveline()

```
{
extern double x, y, z,xd, yd, zd,speed;
extern int exton,vectotal;
double nx,ny,nz,xl,y1,lx1, ly1, lz1,lx2, ly2, lz2;
extern int curs,firstcall;
int ch;
int outf,run,def,cnt;
FILE *f1;
mcenter();
clears(0);
def=0;
cnt=0;
vectotal=0;
curs=0;
firstcall=0;
color(29);
move3(0.0,500.0,0.0);
draw3(0.0,-500.0,0.0);
move3(500.0,0.0,0.0);
draw3(-500.0,0.0,0.0);
mdiden();
f1=fopen("points.d","w");
    nx =0.0 ;
    ny =0.0 ;
    nz= 0.0 ;
    outf=0;
    run=0;
```

/* DRAWS QUADRANTS */

/* RESET MODELING MATRIX */

top:

```
    ch = getche();
    outf=0;
    run=0;
    lx1=x;
    ly1=y;
    lz1=z;
```

/* CALCULATES DIFFERENCE OF POINTS */

```

lx2=lx1-x;
ly2=ly1-y;
lz2=lz1-z;
linfun(2);
while(run==0)
{
    /* TRANSLATES CURSOR */
    switch(ch)
    {

        case '+' : speed++;
                    break;
        case '-' : speed--;
                    break;
        case '4' : left();
                    break;
        case '6' : right();
                    break;
        case '8' : up();
                    break;
        case '2' : down();
                    break;
        case '7' : upleft();
                    break;
        case '9' : upright();
                    break;
        case '3' : downright();
                    break;
        case '1' : downleft();
                    break;
        case CR   : run=1;
                    outf=1;
                    fprintf(f1, "\n      %.2f, %.2f", x, y);
                    printf("\nPRINT *** %.2f, %.2f *** ", x, y);
                    if(vectotal==0)
                    {
                        x1=x;
                        y1=y;
                    }
                    cnt++;
                    vectotal++;

                    break;
        case ESC  : run=2;
                    outf=1;
                    break;

        default   : run=1;
                    def=1; ;
    } /* switch */
}

```



```

if(def==1)
    goto tdef;
if(outf==1)
    goto out;
lx2=lx1-x;          /* CALCULATES POINT DIFFERENCE */
ly2=ly1-y;
lz2=lz1-z;
nx = -lx2;
ny = -ly2;
nz = -lz2;
linfun(2);
if(vectotal>0)
drawr3(lx2,ly2,lz2);    /* DRAWS LINE */
printf(
"\nx = %.2f y = %.2f z = %.2f grid = %.2f "
, x, y, z, speed);
ch = getche();
out:
linfun(2);
if((vectotal>1) && (cnt!=0))
{
    if(cnt!=1)
    {
        mover3(nx,ny,nz);    /* DRAWS LINE */
        drawr3(lx2,ly2,lz2);
    }
    cnt++;
}
/*PRINTS VALUES */
printf(
"-x = %.2f -y = %.2f -z = %.2f grid = %.2f "
, lx2, ly2, lz2,speed);
tdef:
def=0;
}
if(run==1)
    goto top;
fprintf(f1,"\n      %.2f, %.2f",x1,y1);
vectotal++;          /* COUNT VECTOR NUMBER */
fclose(f1);
if(exton==0)
    lathe();          /* EXIT TO LATHR */
if(exton==1)
    extrnd();          /* EXIT TO EXTRND */
}

```

```

/*****/
lathe()
/* CONSTRUCT A TURNED SOLID */
{
extern double fvar, fvar1,fvar2;
extern int exton,randisk;
static char input40;
int ct = 0;
FILE *f1;
FILE *f2;
char string40;
float rot;
double *xptr;
double xlist125;
double *yptr;
double ylist125;
double var1;
double var2;
extern int ssnum, vectotal;
extern double rotnum;
f1 = fopen("points.d","r");
f2 = fopen("ssolid.d","w");
ssnum=1;
rotnum=360.0;
printf("\n TEST VECTOTAL= %d ",vectotal);
fprintf(f2,"ssolid 1, %d,",vectotal);
/* BEGIN SSOLID COMMAND */
printf("\nENTER ROTATION ANGLE");
while(scanf("%f",&rot) !=1)
{
printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
scanf("%s");
}
while(getchar()!='\n');
printf("\nTEST TEST ROTATION = %.2f",rot);
rotnum=rot;
while (fgets(input,LIM,f1)!=NULL)
fprintf(f2,"%s",input);
fprintf(f2,"\n: %.2f ",rotnum);
fclose(f2);
fclose(f1);
creatss();
getssol();
}

/*****/

```

```

/*****
extrud()
{
    extern double fvar, fvar1,fvar2,depth1,depth2,depth3;
    extern int randisk;
    static char input40;
    int ct = 0;

    FILE *f1;
    FILE *f2;
    char string40;
    double *xptr;
    double xlist125;
    double *yptr;
    double ylist125;
    double var11;
    double var21;
    float ht;
    extern int ssnum, vectotal;
    extern double rotnum;
    f1 = fopen("points.d","r");
    f2 = fopen("ssolid.d","w");
    ssnum=0;
    rotnum=360.0;
    depth1=0.0;
    depth2=0.0;
    depth3=100.0;
    printf("\n TEST VECTOTAL= %d ",vectotal);
    fprintf(f2,"ssolid 0, %d,",vectotal);

        while (fgets(input,LIM,f1)!=NULL)
            fprintf(f2,"%s",input);
        printf("\nENTER HEIGHT");
    while(scanf("%f",&ht) !=1)
    {
        printf("PLEASE USE NUMERIC INPUT; TRY AGAIN.\n");
        scanf("%s");
    }
    while(getchar()!='\n');
    printf("\nTEST TEST ROTATION = %.2f",ht);
    depth3=ht;
    fprintf(f2,"\n,      %.2f,%.2f,%.2f",depth1,depth2,depth3);
    fclose(f2);
    fclose(f1);
    creatss();
    getssol();
}
*****/

```

```

/*****/
getssol()
{
extern int wdown,exton;
extern int count,planon;
extern double depth1,depth2,depth3, x, y, z;
extern double rotnum,fvar, fvar1,fvar2,fvar3;
extern int ivar, ivar1, ivar2;
extern int ssnum,vectotal;

static char input40;
static char str140;
int ct = 0;

char string40;
extern double xlist225;
extern double ylist225;
extern double *xptra;
extern double *yptra;
int sscount,total,ssct,vtot;
FILE *fl;

yptra=ylist;
xptra=xlist;

vtot=0;
ssct=0;
total=0;
sscount=0;
vectotal=0;
fl = fopen("ssolid.d","r");
while (fgets(input,LIM,fl)!=NULL) /* RECEIVES COMMANDS */
{
strcpy(str1,input);
fscanf(input,6);
if (((strcmp(input,SSOLID) == 0)!!(vtot!=0)!!(sscount==1))
{
strcpy(string,&str1);
tok(string);
if(vectotal==0) /* TEST FOR LAST VECTOR */
{
ssnum=ivar1; /* ASSIGNS VECTOR TOTALS */
vectotal=ivar2;
vtot=ivar2;
}
}
}

```

```

if(sscount== -1)
{
    /* READS DEPTH */
    depth1=fvar1;
    depth2=fvar2;
    depth3=fvar3;
    rotnum=fvar1;
    clbeg(1);
    /* SETS SSOLID COMMAND TO COMMAND LIST */
if(exton==0)
    ssolid(ssnum,vectotal,xlist,ylist,rotnum);
if(exton==1)
    ssolid(ssnum,vectotal,xlist,ylist,depth1,depth2,depth3);
    clend();
    vectotal=0;
    rotnum=0.0;
    vtot=0;
    ssct=0;
    total=0;
    sscount=-1;
}
if((total<vtot)&&(sscount>0))
{
    xlist: set=fvar1;
    ylist: set=fvar2;
    total=ssct;
    ssct++;
    if(total==vtot-1)
    {
        sscount=-2;
        xptr=xlist;
        yptr=ylist;
        vtot=0;
    }
}
    sscount++;
}
}
}

```

```

/*****

```

```

creatss()
{
extern int ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6,dsp;
extern int randisk;
extern double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
extern char sign;
static char input40;
FILE *f2;
FILE *f1;
FILE *f3;
char name0;
char comm0;
char rsp;
int dex;
int cnt,tester;
static char filvar12;
cnt=0;
top:
tester=0;
printf("\nWHAT IS THE NAME OF OBJECT ?");
gets(name);
printf(" char= %c =%3d\n",
      name0, name0);
printf("STRING==%c,%c,%c,%c,%c,%c",
      name0, name1, name2, name3, name4, name5 );
printf("\nNAME==%s",name);
if (name0< 97)
{
printf("\nBAD NAME TRY AGAIN ?");
strcpy(name,COMMAND);
tester=1;
if(cnt>3)
{
tester=0;
name0= 97;
}
}
if (tester==1)
goto top;
strcpy(filvar,name);
if(( f3 = fopen(filvar,"r")) !=NULL)
{
fclose(f3);
printf(
"%s EXISTS -- do you wish to over write { y/n } ? ",filvar);
rsp=getche();
}
}

```

```

    if(rsp == 'y')
    {
        tempc(filvar);
        goto lable;
    }
    else
        goto end;
}
lable:
sizfvar(filvar);
printf("\nANY COMMENTS ABOUT THE FILE ?");
gets(comm);
printf("Addr=%5u char='%c'=%3d\n",
    &comm0, comm0, comm0);
if (comm0 < 97)
{
    strcpy(name.COMMENT);
}
f1 = fopen("objectlist.i","a");
fprintf(f1,"%s      '%s'  \n",filvar,comm);
fclose(f1);
f2 = fopen(filvar,"w");
if(randisk==0)
    f3 = fopen("ssolid.d","r");
if(randisk==1)
    f3 = fopen("d:ssolid.d","r");
while (fgets(input,LIM,f3)!=NULL)
{
    fprintf(f2,"%s",input);
}
fclose(f1);
fclose(f2);
end;;
}

```

./<*****>/

```

recissol()
{
extern int wdw,ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6,dsp,rnum;
extern int randisk;
extern double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
extern char sign;
extern double x,y,z;
static char input40;
FILE *f2;
FILE *f1;
FILE *f3;
int tester;
static char name12;
char filvar10;
int cnt;
cnt=0;
top:
printf("\nWHAT IS THE OBJECT TO RECIEVE ?");
    gets(name);
printf(" char= %c %3d\n",
    name0, name0);
printf("STRING==%c,%c,%c,%c,%c,%c",
    name0, name1, name2, name3, name4, name5 );
if (name0< 97)
{
    printf("\nBAD NAME TRY AGAIN ?");
    strcpy(name,COMMAND);
    tester=1;
    cnt++;
    if(cnt>3)
    {
        tester=0;
        printf(" NAME==%s",name);
    }
}
if (tester==1)
    goto top;
if((f3= fopen(name,"r"))==NULL)
{
    printf("\nOBJECT NOT FOUND:");
    fclose(f3);
    goto end;
}
f2 = fopen(name,"r");
if(randisk==0)
    f1 = fopen("ssolid.d","w");
if(randisk==1)
    f1 = fopen("d:ssolid.d","w");

```



```
while (fgets(input,LIM,f2)!=NULL)
    fprintf(f1,"%s",input);
fclose(f1);
fclose(f2);
getssol();
end;
}/* end main */
/*****/
```

```

/* THESE FUNCTIONS ARE SPECIALIZED FUNCTIONS; REMOVE;EXTRACT; SUBTRACT ; LOAD COMMAND LIST */
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <stdio.h>
#include "std.h"
#include <sys/types.h>
#include <io.h>
#include <string.h>
#include "keys.h"
#include "c2.h"
char sign;
int ivar,ivar1,ivar2,ivar3,ivar4,ivar5,ivar6,count, clcount;
double fvar,fvar1,fvar2,fvar3,fvar4,fvar5,fvar6;
double xcon, ycon, zcon;
/*****/
remobj()

{ /*BEGIN*/
extern int proj, dish,vwx, vwy, vwz;
extern int vwx1v, vwx2v, vwy1v, vwy2v;
extern int randisk ,wdow;
extern int count,planon;
extern double vwptx,vwpty,vwptz;
extern double winx1, winx2, winy1, winy2;      /* SETS WINDOW SIZE */
extern double height,length, width, x,y,z;
extern double depth1,depth2,depth3,rotnum,fvar, fvar1,fvar2,fvar3;
extern int ivar, ivar1, ivar2;
extern int vectotal,total;
extern int sscount,ssnum,total,ssct,vtot;
static char input00;
static char str100;
int ct = 0;
extern int vali;
FILE *f1;
FILE *f2;
FILE *f3;
char string00;
int locator =0;
extern double px1, py1, pz1;
extern double px2, py2, pz2;
extern double px3, py3, pz3;
extern double px4, py4, pz4;
extern double px5, py5, pz5;
extern double px6, py6, pz6;
extern double px7, py7, pz7;
extern double px8,py8, pz8;

```

```

xyzsend();                /* STORES XYZ COORDINATES */
vali=0;
cleari(0);
clrun(249);
defpts();                 /* DEFINES THE PERIMETER POINTS */
                           /* SWEEP SOLID RESET VALUES */

vectotal=0;
vtot=0;
ssct=0;
total=0;
sscount=0;

if (randisk==1)
{
    /* OPEN FILES */
    f2 = fopen("d:command.i","w");
    f1 = fopen("d:command.d","r");
}
if (randisk==0)
{
    f2 = fopen("command.i","w");
    f1 = fopen("command.d","r");
}
dispat(0.1);
window(winx1. winx2. winv1. winv2):    /* SETS WINDOW SIZE */
if(wdow==1)
{
    vwrotx(vwx):
    vwroty(vwy):
    vwrotz(vwz):
    window(winx1. winx2. winv1. winv2):    /* SETS WINDOW SIZE */
    vwoort(vwx1v. vwx2v. vwy1v. vwy2v):
    project(proj);
    disth(dish);
    vwvpt(vwptx,vwpty,vwptz);
}
while (fgets(input,LIM,f1)!=NULL)        /* READS IN FILE */
{
    strcpy(str1,input);                  /* STORES 1ST 6 CHARACTERS */
    fit(input,6);
    ssolidr(input,str1);                 /* TESTS FOR SSOLID COMMAND */
    if((strcmp(SSOLID,input)==0)||!(strcmp(BLANK,input)==0))
    {
        fprintf(f2,"%s",str1);          /* PRINTS SSOLID COMMAND */
    }
}

```

```

if(sscount==0)                                /* TESTS FOR END OF SSOLID COMMAND*/
{
    strcpy(string,&str1);                      /* TAKES LAST CHARACTERS FROM STRING */
    tok(string);                               /* SEPARATES NUMBERS */
    if(strcmp(LOCATE,input)==0)                /* TESTS LOCATION*/
    {
        /*      printpts();*/
        vali=0;
        filterpts();
    }
    if ((vali ==0)|| (strcmp(MDTRAN,input)==0)) /* PRINTS COMMANDS THAT PASS */
    {
        printf("\n%s*",str1);
        count=0;
        func2(input);
        fprintf(f2,"%s",str1);
    }
}
}/* WHILE END */
fclose(f1);                                   /* CLOSES FILE */
fclose(f2);
dispi(0);
if (randisk==1)
{
    f2 = fopen("d:command.i","r");
    f1 = fopen("d:command.d","w");
}
if (randisk==0)
{
    f2 = fopen("command.i","r");
    f1 = fopen("command.d","w");
}
while (fgets(input,LIM,f2)!=NULL)              /* SEND FILE TO COMMAND.D */
    fprintf(f1,"%s",input);
fclose(f1);
fclose(f2);

getfilel();                                  /* RESETS SCREEN */
}/* END*/
/*****/

```

```
/.*****/
```

```
extract()
TER
```

```
/* SAME PROCESS AS REMOVE EXCEPT OBJECT IN THE PERINE
```

```
/* ARE STORED*/
```

```
{ /*BEGIN*/
```

```
extern int proj, dish,vwx, vwy, vwz;
```

```
extern int vwxlv, vwx2v, vwy1v, vwy2v;
```

```
extern int randisk ,wdow;
```

```
extern int count,planon;
```

```
extern double vwptx,vwpty,vwptz;
```

```
extern double winx1, winx2, winy1, winy2; /* SETS WINDOW SIZE */
```

```
extern double height,length, width, x,y,z;
```

```
extern double depth1,depth2,depth3,rotnum,fvar, fvar1,fvar2,fvar3;
```

```
extern int ivar, ivar1, ivar2;
```

```
extern int vectotal,total;
```

```
extern int sscount,ssnum,total,ssct,vtot;
```

```
static char input[0];
```

```
static char str1[0];
```

```
int ct = 0;
```

```
extern int vali;
```

```
FILE *f1;
```

```
FILE *f2;
```

```
FILE *f3;
```

```
char string[0];
```

```
int locator =0;
```

```
extern double px1, py1, pz1;
```

```
extern double px2, py2, pz2;
```

```
extern double px3, py3, pz3;
```

```
extern double px4, py4, pz4;
```

```
extern double px5, py5, pz5;
```

```
extern double px6, py6, pz6;
```

```
extern double px7, py7, pz7;
```

```
extern double px8,py8, pz8;
```

```
xyzsend();
```

```
vali=0;
```

```
cleari(0);
```

```
clrwn(249);
```

```
defpnts();
```

```
vectotal=0;
```

```
vtot=0;
```

```
ssct=0;
```

```
total=0;
```

```
sscount=0;
```

```

if (randisk==1)
{
    f2 = fopen("d:command.i","w");
    f1 = fopen("d:command.d","r");
}
if (randisk==0)
{
    f2 = fopen("command.i","w");
    f1 = fopen("command.d","r");
}
dispat(0,1);
window(winx1, winx2, winy1, winy2);    /* SETS WINDOW SIZE */
if(wdow==1)
{
    vwrotx(vwx);
    vwroty(vwy);
    vwrotz(vwz);
    window(winx1, winx2, winy1, winy2);    /* SETS WINDOW SIZE */
    viewport(vwx1v, vwx2v, vwylv, vwyl2v);
}
while (fgets(input,LIM,f1)!=NULL)
{
    strcpy(str1,input);
    fit(input,6);
    ssolidr(input,str1);
    if((strcmp(SSOLID,input)==0)||!(strcmp(BLANK,input)==0))
    {
        fprintf(f2,"%s",str1);
    }
    if(sscount==0)
    {
        strcpy(string.&str1);
        tok(string);
        if(strcmp(LOCATE,input)==0)
        {
            vali=0;
            filterpts();
        }
        if ((vali ==1)||!(strcmp(NDTRAN,input)==0))    /* DIFFERENCE HERE */
        {
            printf("\n*%s*",str1);
            count=0;    /* STORES AND RUNS COMMANDS */
            func2(input);
            fprintf(f2,"%s",str1);
        }
    }
}
}/* WHILE END */

```

```

        fclose(f1);
        fclose(f2);
        dispi(0);
if (randisk==1)
    {
        f2 = fopen("d:command.i","r");
        f1 = fopen("d:command.d","w");
    }
if (randisk==0)
    {
        f2 = fopen("command.i","r");
        f1 = fopen("command.d","w");
    }
while (fgets(input,LIM,f2)!=NULL)
    fprintf(f1,"%s",input);
    fclose(f1);
    fclose(f2);

getfile!():

```

```

    }/* END*/

```

```

/*****

```

```
/* NOT USED FOR TESTING ONLY*/
```

```
printpts()
```

```
{
```

```
extern double px1, py1, pz1;
```

```
extern double px2, py2, pz2;
```

```
extern double px3, py3, pz3;
```

```
extern double px4, py4, pz4;
```

```
extern double px5, py5, pz5;
```

```
extern double px6, py6, pz6;
```

```
extern double px7, py7, pz7;
```

```
extern double px8,py8, pz8;
```

```
FILE *sl;
```

```
extern double fvar, fvar1,fvar2,fvar3;
```

```
sl = fopen("pts.i","a");
```

```
fprintf(sl," %.1f < %.1f * %.1f > %.1f * %.1f > %.1f\n",px1,fvar1,py1,fvar2,pz1,fvar3);
```

```
fprintf(sl," %.1f < %.1f * %.1f > %.1f * %.1f < %.1f\n ",px2,fvar1,py2,fvar2,pz2,fvar3);
```

```
fprintf(sl," %.1f > %.1f * %.1f > %.1f * %.1f < %.1f\n ",px3,fvar1,py3,fvar2,pz3,fvar3);
```

```
fprintf(sl," %.1f > %.1f * %.1f > %.1f * %.1f > %.1f\n ",px4,fvar1,py4,fvar2,pz4,fvar3);
```

```
fprintf(sl," %.1f < %.1f * %.1f < %.1f * %.1f > %.1f\n ",px5,fvar1,py5,fvar2,pz5,fvar3);
```

```
fprintf(sl," %.1f < %.1f * %.1f < %.1f * %.1f < %.1f\n ",px6,fvar1,py6,fvar2,pz6,fvar3);
```

```
fprintf(sl," %.1f > %.1f * %.1f < %.1f * %.1f < %.1f\n ",px7,fvar1,py7,fvar2,pz7,fvar3);
```

```
fprintf(sl," %.1f > %.1f * %.1f < %.1f * %.1f > %.1f\n ",px8,fvar1,py8,fvar2,pz8,fvar3);
```

```
fprintf(sl,"*****\n ");
```

```
fclose(sl);
```

```
}
```

```
/*******/
```



```

defpnts()                /* DEFINES THE 8 PERIMETER POINTS */
{
extern double px1, py1, pz1;
extern double px2, py2, pz2;
extern double px3, py3, pz3;
extern double px4, py4, pz4;
extern double px5, py5, pz5;
extern double px6, py6, pz6;
extern double px7, py7, pz7;
extern double px8, py8, pz8;

extern double height, length, width, x, y, z;
FILE *s1;

px1=(-width/2.0)+x;
py1=(height/2.0)+y;
pz1=(length/2.0)+z;
px2=(-width/2.0)+x;
py2=(height/2.0)+y;
pz2=(-length/2.0)+z;

px3=(width/2.0)+x;
py3=(height/2.0)+y;
pz3=(-length/2.0)+z;

px4=(width/2.0)+x;
py4=(height/2.0)+y;
pz4=(length/2.0)+z;

px5=(-width/2.0)+x;
py5=(-height/2.0)+y;
pz5=(length/2.0)+z;

px6=(-width/2.0)+x;
py6=(-height/2.0)+y;
pz6=(-length/2.0)+z;

px7=(width/2.0)+x;
py7=(-height/2.0)+y;
pz7=(-length/2.0)+z;

px8=(width/2.0)+x;
py8=(-height/2.0)+y;
pz8=(length/2.0)+z;
}
/*****/

```

```

/*****/
filterpts()                /* FILTERS EACH POINT */
{
extern int vali;
extern double fvar, fvar1,fvar2,fvar3;
extern double px1, py1, pz1;
extern double px2, py2, pz2;
extern double px3, py3, pz3;
extern double px4, py4, pz4;
extern double px5, py5, pz5;
extern double px6, py6, pz6;
extern double px7, py7, pz7;
extern double px8,py8, pz8;
    if((px1<fvar1) && (py1>fvar2) && (pz1>fvar3))
    {
        if((px2<fvar1) && (py2>fvar2) && (pz2<fvar3))
        {
            /* printf("PT2 OK "); */
            if((px3>fvar1) && (py3>fvar2) && (pz3<fvar3))
            {
                /* printf("PT3 OK "); */
                if((px4>fvar1) && (py4>fvar2) && (pz4>fvar3))
                {
                    /* printf("PT4 OK "); */
                    if((px5<fvar1) && (py5<fvar2) && (pz5>fvar3))
                    {
                        /* printf("PT5 OK "); */
                        if((px6<fvar1) && (py6<fvar2) && (pz6<fvar3))
                        {
                            /* printf("PT6 OK "); */
                            if((px7>fvar1) && (py7<fvar2) && (pz7<fvar3))
                            {
                                /* printf("PT7 OK ");*/
                                if((px8>fvar1) && (py8>fvar2) && (pz8>fvar3))
                                {
                                    vali=1;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
/*****/

```

```

/*****
setcl2()                               /* SETS COMMAND LIST TO BOARD*/

{ /*BEGIN*/
extern int shade,vwx, vwy, vwz;
extern int vwxlv, vwx2v, vwxlv, vwy2v;
extern int randisk ,wdow;
extern int count,planon;
extern double winx1, winx2, winy1, winy2;      /* SETS WINDOW SIZE */
extern int vectotal,total,faceton;
extern int sscount,ssnum,total,ssct,vtot;
static char input40;
static char str140;
int ct = 0;
extern int vali;
FILE *f1;
FILE *f2;
char string40;
int locator =0;
shade=0;
xyzsend();
vali=0;
vectotal=0;
vtot=0;
ssct=0;
total=0;
sscount=0;

```

```

if (randisk==1)
{
    f2 = fopen("d:command.i","w");
    f1 = fopen("d:command.d","r");
}
if (randisk==0)
{
    f2 = fopen("command.i","w");
    f1 = fopen("command.d","r");
}
mdiden();                                /* RESETS MODELING MATRIX */

window(winx1, winx2, winy1, winy2);      /* SETS WINDOW SIZE */
if(wdow==1)
{
    vwrotx(vwx);
    vwroty(vwy);
    vwrotz(vwz);
    window(winx1, winx2, winy1, winy2);
    viewport(vwxlv, vwx2v, vwylv, vwyl2v);
}
printf("\nCLRUN TEST LINE 475 ok util.c ");
setupvars();                             /* RUNS SETUP COMMANDS */
printf("\nCLRUN TEST LINE 459 ok util.c ");
cldel(2);                                /* DELETES EXISTING COMMAND LIST */
dispat(2,0);                             /* SETS TO WIRE FRAME */
clbeg(2);                                /* BEGINS COMMAND LIST */
printf("\nCLRUN TEST LINE 462 ok util.c ");
while (fgets(input,LIM,f1)!=NULL)        /* READS FILE */
{
    strcpy(str1,input);
    fit(input,6);
    ssolidr(input,str1);
    if(sscount==0)
    {
        strcpy(string,&str1);
        tok(string);
        count=0;
        if(faceton==1)
        {
            if(strcmp(DISPAT,input)!=0)
                func2(input);
        }
        if(faceton==0)
        {
            func2(input);
        }
    }
}
}/* WHILE END */

```

```

mdiden();
clend();
fclose(f1);
fclose(f2);
clrun(2);
/* ENDS COMMANDLIST */
/* CLOSES FILE */
/* RUNS COMMAND LIST*/

```

```

}/* END*/
/*****/
/*****/
subtract()
/* PERFORMS A BOOLEAN OPERATION */

```

```

{ /*BEGIN*/
extern int vwx, vwy, vwz;
extern int vwxlv, vwzlv, vwy1v, vwy2v;
extern int randisk, wdw;
extern int count, planon, ivar;
extern double winx1, winx2, winy1, winy2; /* SETS WINDOW SIZE */
extern double fvar2, fvar3, fvar4, height, length, width, x, y, z;
extern double rotnum, fvar, fvar1, fvar2, fvar3;
extern int ivar, ivar1, ivar2;
extern int vectotal, total;
extern int sscount, ssnum, total, ssct, vtot;
static char input40;
static char str140;
int ct = 0;
extern int vali;
FILE *f1;
FILE *f2;
FILE *f3;
char string40;
int locator = 0;
extern int bop;
int key;
int cnt;
if(bop==1)
/* TEST FOR FIRST PASS */
{
f3 = fopen("sub.i", "w");
fclose(f3);
ivar=4;
fvar2=width;
fvar3=height;
fvar4=length;
block(4, width, height, length);
key=SS;
xyzsend();
defpnts();
vali=0;
vectotal=0;

```

```

vtot=0;
ssct=0;
total=0;
sscount=0;
keyloop();                                /* ENTER KEYBOARD */
}
if(bop==0)                                /* COMPLETION TEST */
    goto out;
if(bop==2)
{
    if (randisk==1)
    {
        /* OPEN FILES */
        f3 = fopen("d:setsub.i","w");
        f2 = fopen("d:command.i","w");
        f1 = fopen("d:command.d","r");
    }
    if (randisk==0)
    {
        f3 = fopen("setsub.i","w");
        f2 = fopen("command.i","w");
        f1 = fopen("command.d","r");
    }
    fprintf(f3,"solbeg 1");                /* PRINTS SOL COMMANDS TO FILE */
    fprintf(f3,"\\nsolbop 0");
    while (fgets(input,LIM,f1)!=NULL)      /* READS STRING COMMANDS */
    {
        strcpy(str1,input);
        fit(input,6);
        ssolidr(input,str1);
        if((strcmp(SSOLID,input)==0)|| (strcmp(BLANK,input)==0))
        {
            fprintf(f2,"%s",str1);
        }
        if(sscount==0)
        {
            strcpy(string,&str1);
            tok(string);
            if(strcmp(LOCATE,input)==0)
            {
                /*      printpts();*/
                vali=0;
                filterpts();
            }
        }
    }
}

```

```

        if ((vali ==0)|| (strcmp(MDTRAN,input)==0))
        {
            printf("\n*%s*",str1);          /* DOES NOT EXECUTE ONLY PRINTS */
            count=0;
/*      func2(input); */
            fprintf(f2,"%s",str1);
        }
        if (vali ==1)
        {
            if (strcmp(LOCATE,input)==0)
            {
                strcpy(string,&str1);      /* TRANSFORMS LOCATION TO TRANSLATE COMMA
ND */
                fprintf(f3,"\nmdiden");      /* RESETS MODELING MATRIX */
                fprintf(f3,"\nmdtran %s",string); /* TRANSLATES SOLID */
            } /* END LOCATE IF */
            count=0;
            if (strcmp(MDTRAN,input)!=0)
            if (strcmp(LOCATE,input)!=0)      /* PRINTS ALL TRANSLATION COMMANDS*/
                fprintf(f3,"%s",str1);
            } /* END VALI IF */

        } /* END SSCOUNT IF */
    } /* WHILE END */
    fclose(f3);
    fclose(f1);          /* CLOSSES FILE */
    fclose(f2);
} /* END IF BOP STATEMENT */
if (bop==2)
    subtract2();
    out;;
} /* END*/

/*****/

```

```

subtract2()
/* TRANSFERS FILES TO COMMAND FILE */
{
    extern double x,y,z,xd,yd,zd;
    static char input40;
    static char str140;
    FILE *f1;
    FILE *f2;
    FILE *f3;
    FILE *f4;
    extern int bop, randisk;
    f3 = fopen("setsub.i","r");
    f4 = fopen("sub.i","r");
    bop=0;

    if (randisk==1)
    {
        f2 = fopen("d:command.i","r");
        f1 = fopen("d:command.d","w");
    }
    if (randisk==0)
    {
        f2 = fopen("command.i","r");
        f1 = fopen("command.d","w");
    }
    while (fgets(input,LIM,f2)!=NULL)
        fprintf(f1,"%s",input);
    while (fgets(input,LIM,f3)!=NULL)
        fprintf(f1,"%s",input);
    while (fgets(input,LIM,f4)!=NULL)
        fprintf(f1,"%s",input);

    fprintf(f1,"\nsolend ");
    fprintf(f1,"\nsolrun 1 ");
    fprintf(f1,"\nmdiden\n ");
    x=0.0;
    y=0.0;
    z=0.0;
    xd=0.0;
    yd=0.0;
    zd=0.0;
    fclose(f3);
    fclose(f1);
    fclose(f2);
    fclose(f4);

    getfile();
}/* END*/
/*****/

```

9

/* TRANSFERS FILES */

/* WRITES END COMMANDS */

/* CLOSES FILE */

/* RUNS COMMAND FILE */

**The vita has been removed from
the scanned document**