AN ABSTRACT OF THE THESIS OF

Ratchatin Chancharoen for the degree of Master of Science in Mechanical Engineering presented on June 30, 1994.

Title: Computer Assisted Design of Planar Workholders

Abstract approved: Redacted for Privacy

Eugene F. Fichter

There are many situations in which contact between two bodies is at a single point so there can be no tensile (force). This form of contact only provides a single degree of restraint, and is defined as a unidirectional point contact. The analysis of Reuleaux shows that four suitably placed frictionless point contacts are required to completely restrain an object in a plane.

The objective of the thesis is to allow user-placement of three contacts and to find acceptable range for placement of the fourth. If the fourth contact is anywhere in the range, the four forces fulfill the requirement for total planar restraint, i.e. all translations and rotations in a plane are prevented. This project considers the directions and placement allowed for the fourth force but does not take into consideration the magnitudes or friction coefficients of any of the four forces.

In this research a program named Planar Restraint Design Assistant (PRDA) was developed to analyze positions of the four restraint forces. A planar object is first specified by the user; the program accepts AutoCAD images for more complex objects. For three given restraint forces PRDA determines a range of the fourth force such that total restraint is achieved. In addition, PRDA allows the user to arrange three restraint forces until a desired range of the fourth restraint force is obtained. Results are shown in visual form with accompanying graphs for numerical

interpretation. A program user's guide and a program learning guide are provided as shown in the report with illustrative examples. Program source code in QuickBASIC is included in the report.

COMPUTER ASSISTED DESIGN OF PLANAR WORKHOLDERS

by

Ratchatin Chancharoen

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June 30, 1994

Commencement June 1995

APPROVED:

Redacted for Privacy

Professor of Mechanical Engineering in charge of major

Redacted for Privacy

Head of department of Mechanical Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 30, 1994

Typed by researcher for Ratchatin Chancharoen

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

COMPUTER ASSISTED DESIGN OF PLANAR WORKHOLDERS

CHAPTER 1
INTRODUCTION

In many practical systems, the connection between the bodies is achieved by the use of classical kinematic pairs, and the resulting freedoms are easily analyzed. In contrast, any required relative freedoms between the two bodies can be synthesized by the use of appropriate kinematic pairs.

However, there are many situations in which contact between two bodies does not take the form of combinations of classical kinematic pairs, but rather of a number of contact points where there can be no tensile connection. This form of contact only provides a single degree of restraint in one sense, and is defined as a unidirectional point contact.

The analysis of Reuleaux (1876) shows that four unidirectional point contacts are required to prevent translation and rotation simultaneously in a plane. The purpose of this thesis is to find the fourth restraint that results in total restraint when three point contacts are given.

As a result of this research, a program named PRDA (Planar Restraint Design Assistant) was developed on the basis of the moment method. PRDA performs an analysis of positions of four restraint forces for two-dimensional restraint. This thesis consists of five chapters including this one.

Chapter 2 describes the fundamental theory of planar restraint. The requirements for total restraint are presented as well as equations used in the program.

Chapter 3 presents the user interface describing all parameters used in the program and how to set them. In addition, the meanings of the results are discussed and examples including design of jig for manufacturing process are included.

Chapter 4 describes the structure of the program along with how it was developed. All function keys are presented as well as how the program displays the results.

Chapter 5 describes assumptions made in this thesis and program limitations. Recommendations for further study are also presented.

Appendix A presents program learning guide, showing an example of how to set each parameter as well as how to get the results. The purpose of this section is to help unfamiliar users.

Appendix B presents source code of PRDA, written in QuickBasic along with source codes of GRP and HELP, callable from PRDA.

CHAPTER 2
THEORY OF PLANAR RESTRAINT

The theory of planar restraint of a body by multiple frictionless point contacts is discussed in this section. Reuleaux (1876) described a graphical method for evaluating translational and rotational restraints in planar cases using unidirectional point contacts. Kerr and Sanger (1983) developed an analytical theory of planar restraint. Both analyses show that four contact points are required for total restraint in a plane. This thesis uses a method based on moment calculations (called the Moment method) that is related to both Reuleaux and Kerr and Sanger methods. This new method evaluates whether a fourth contact results in restraint when the other three contacts are given.

2.1 Reuleaux's method

The following paragraphs summarize Reuleaux's method as presented on pages 98 through 114 of his book ***The Kinematics of Machinery*** which was originally published in 1876. Translational restraint and rotational restraint are treated separately in this method. Total restraint is achieved by preventing translation and rotation simultaneously.

Translational restraint

In the following discussion, it is assumed that only translation is possible. Rotation is prevented by some unspecified agencies.

- **Single point of restraint**

In Figure 201, body *A* is partially restrained by contact at one point with a second body *B*. Body *A* cannot translate in any direction that has a component in the *aN'* direction. The range of possible translational directions is indicated by arrows.

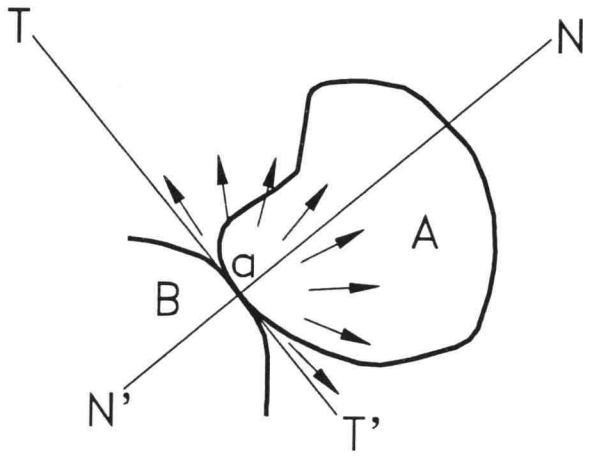

Figure 201 Translational restraint by a contact point

- **Complete translational restraint**

Complete translational restraint is attained by arranging contacts so that every possible translation direction is eliminated. Three contact points are needed to eliminate all possible translational directions (Figure 202).

Tangents at three contact points must form a triangle around at least part of the object to prevent translation. In the other word, the angle between any two consecutive normals at contacts must be less than 180°.

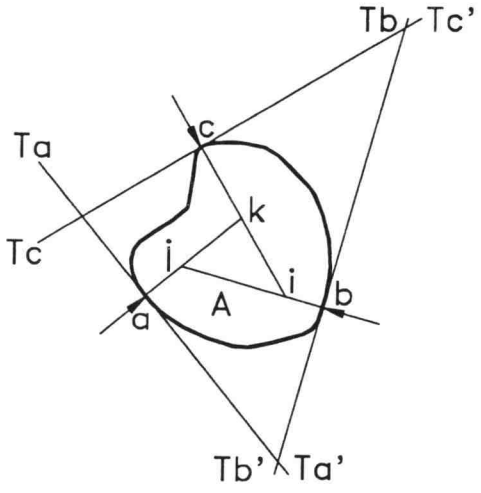

Figure 202 Complete translational restraint

Rotational restraint

Figure 203 is the same picture as Figure 201 but now rotation is allowed. Only rotations that do not move point *a* toward body *B* are allowed. Counterclockwise rotations are possible only if the center of rotation is above *NN'* and to the right of *TT'*. Clockwise rotations are possible only if the center of rotation is below *NN'* and to the right of *TT'*.

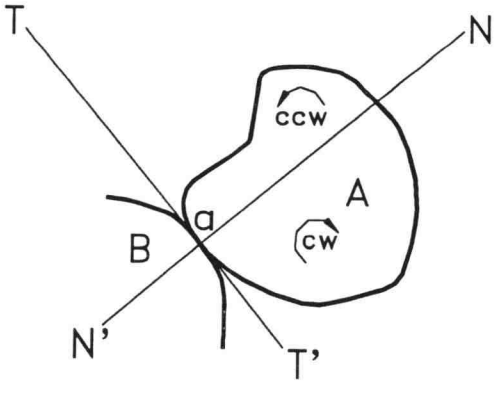

Figure 203 Rotational restraint by a contact point

Imagine standing at the contact point facing body *A* and looking along the contact normal. The center of rotation cannot be behind you. In front of you on your right are all candidate positions for clockwise centers of rotation while on your left are all candidate positions for counterclockwise centers of rotation.

Total restraint

The condition for complete translational restraint is met in Figure 204 and the combined effect of the three contact points eliminates rotation about most points in the plane. No point outside the triangle of tangents can be a center of rotation, nor can most points inside this triangle.

Consider point *C* (Figure 204) for instance; contact point *a* only allows clockwise rotation about this point but contact point *c* only allows counterclockwise rotation about this point. Thus no rotation is possible about point *C*.

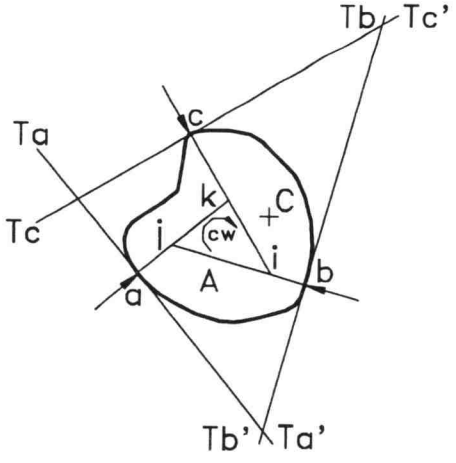

Figure 204 Partial restraint of three contact points

However, it is never possible to completely prevent rotation with three frictionless point contacts since there is always an area (outlined in bold lines in Figure 204) where all contacts allow rotation in the same direction.

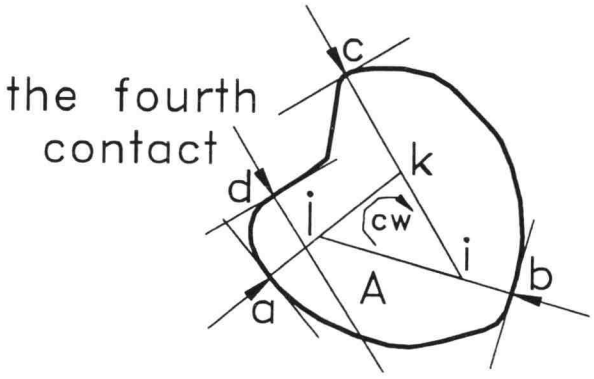

Figure 205 Total restraint

A fourth contact point (Figure 205) can be used to cover this area with its region of contrary rotation. Here contacts *a*, *b* and *c* all allow clockwise rotation about points within triangle *ijk* while contact *d* only allows counterclockwise rotation about points in this triangle.

2.2 Moment method

Reuleaux's method shows that four suitably placed frictionless contact points are required for total restraint. The objective of the moment method is to find a fourth restraint when the other three contacts are given.

In Figure 206, the object is partially restrained by three frictionless contact points with their restraint forces F_1, F_2 and F_3. For total restraint, the fourth force

must eliminate the remaining possible translational directions and the remaining possible centers of rotation simultaneously. This force can be found as any force that produces the contrary moments to forces F_1, F_2 and F_3 about intersection points i, j and k. Detail of this is discussed below.

Lines of forces F_1, F_2 and F_3 separate a plane into 7 regions, five of which are the regions of rotational restraint. However, all contacts, F_1, F_2 and F_3, allow counterclockwise rotation about points in region *2* and clockwise rotation about points in region *5*. Therefore, the fourth force must prevent these rotations.

However, if the fourth force prevents counterclockwise rotation about intersection point j and also clockwise rotation about intersection points i and k, it automatically prevents counterclockwise rotation about every point in region *2* and clockwise rotation about every point in region *5*. As a result, no point can be a center of rotation.

Consider the fourth force *A* for instance; this force produces the contrary moment to F_1 about point i and prevents clockwise rotation about this point. In addition, force *A* produces the contrary moments to F_2 and F_3 about points j and k. Thus force *A* is a candidate for the fourth force.

Force *A* also eliminates the remaining possible translational direction since tangents of contacts at *A*, F_1 and F_3 form a triangle around the object.

Consider the fourth force as *B* instead; this force produces the contrary moments to F_1 and F_2 about points i and j. However, all forces allow clockwise rotation about point k because force *B* produces moment in the same

direction as F_3 about this point. Therefore, force *B* is not a candidate for the fourth force.

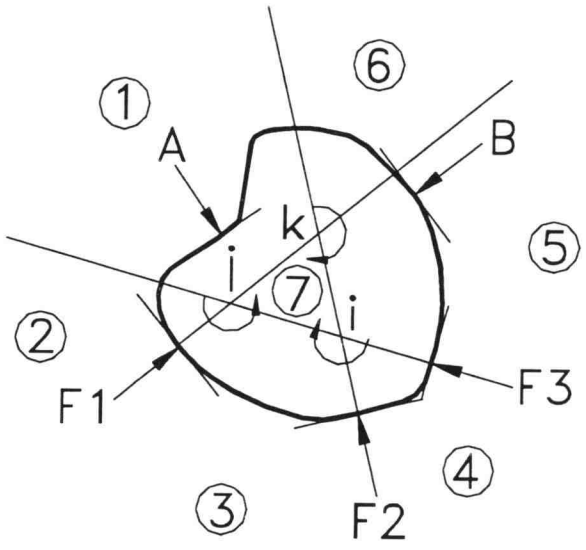

Figure 206 The intersection points of restraint forces

When three restraint forces are given, we can find the corners of a triangle formed by lines of these forces. The corners are the intersections of any two of these lines and are obtained by using equations below.

Given: Positions and directions of each restraint force ($\boldsymbol{x_i}$, $\boldsymbol{y_i}$) and $\boldsymbol{\theta_i}$ (i = 1, 2, 3)

The line of force i can be written as

$$x \cdot \sin(\theta_i) - y \cdot \cos(\theta_i) = A_i$$

where:

$$A_i = x_i \cdot \sin(\theta_i) - y_i \cdot \cos(\theta_i)$$

The intersection of force *i* and *j* is the solution of simultaneous equations of forces *i* and *j* as follows:

$$X\mathrm{int}_{ij} \cdot \sin(\theta_i) - Y\mathrm{int}_{ij} \cdot \cos(\theta_i) = A_i$$
$$X\mathrm{int}_{ij} \cdot \sin(\theta_j) - Y\mathrm{int}_{ij} \cdot \cos(\theta_j) = A_j$$

Where: ***Xint$_{ij}$*** and ***Yint$_{ij}$*** are X and Y coordinates of the intersection point of lines *i* and *j*

Hence, the intersection is

$$X\mathrm{int}_{ij} = \frac{A_j \cdot \cos(\theta_i) - A_i \cdot \cos(\theta_j)}{\sin(\theta_j - \theta_i)}$$

$$Y\mathrm{int}_{ij} = \frac{A_j \cdot \sin(\theta_i) - A_i \cdot \sin(\theta_j)}{\sin(\theta_j - \theta_i)}$$

The moment caused by force *k* about intersection *ij* is obtained by using equations below. Note that forces *i* and *j* pass through this intersection and produce no moment.

Given: Center of moment (***Xint$_{ij}$***, ***Yint$_{ij}$***)
Positions and directions of force *k* (***X$_k$***, ***Y$_k$***) and θ_k

The equation of line *k* can be written as

$$x \cdot \sin(\theta_k) - y \cdot \cos(\theta_k) = x_k \cdot \sin(\theta_k) - y_k \cdot \cos(\theta_k)$$

rewrite,

$$x \cdot l + y \cdot m = p$$

where:

l = $\sin(\theta_k)$
m = $-\cos(\theta_k)$
p = $x_k * \sin(\theta_k) - y_k * \cos(\theta_k)$

The equation of line (called line k-ij) passing through point ($Xint_{ij}$, $Yint_{ij}$), in direction perpendicular to line k can be written as

$$x \cdot m - y \cdot l = X\mathrm{int}_{ij} \cdot m - Y\mathrm{int}_{ij} \cdot l$$

The line k-ij intersects line k at

$$\begin{aligned} X_T \cdot l + Y_T \cdot m &= x_k \cdot l + y_k \cdot m \\ X_T \cdot m - Y_T \cdot l &= X\mathrm{int}_{ij} \cdot m - Y\mathrm{int}_{ij} \cdot l \end{aligned}$$

Where: X_T and Y_T are X and Y coordinates of the intersection point of lines k-ij and k

Hence, the intersection is

$$\begin{aligned} X_T &= m \cdot l \cdot (y_k - Y\mathrm{int}_{ij}) + m^2 \cdot X\mathrm{int}_{ij} + l^2 \cdot x_k \\ Y_T &= m \cdot l \cdot (x_k - X\mathrm{int}_{ij}) + l^2 \cdot Y\mathrm{int}_{ij} + m^2 \cdot y_k \end{aligned}$$

The distance between the center of moment and line k is then the distance between ($Xint_{ij}$, $Yint_{ij}$) and (X_T, Y_T), and is

$$Dist = \sqrt{(X_T - X\mathrm{int}_{ij})^2 + (Y_T - Y\mathrm{int}_{ij})^2}$$

Substitute X_T and Y_T, and simplify

$$Dist = l \cdot (x_k - X\mathrm{int}_{ij}) + m \cdot (y_k - Y\mathrm{int}_{ij})$$

Hence, the moment of line k about ($Xint_{ij}$, $Yint_{ij}$) is

$$moment = l \cdot (x_k - X\mathrm{int}_{ij}) + m \cdot (y_k - Y\mathrm{int}_{ij})$$

or,

$$moment = (x_k - X\mathrm{int}_{ij}) * \sin\theta_k - (y_k - Y\mathrm{int}_{ij}) * \cos\theta_k$$

When a direction of the fourth force is given, we can find a set of positions of the fourth force such that movement is prevented. When a position of the fourth force is given, we can find a set of acceptable directions for it.

When a direction of the fourth force is given

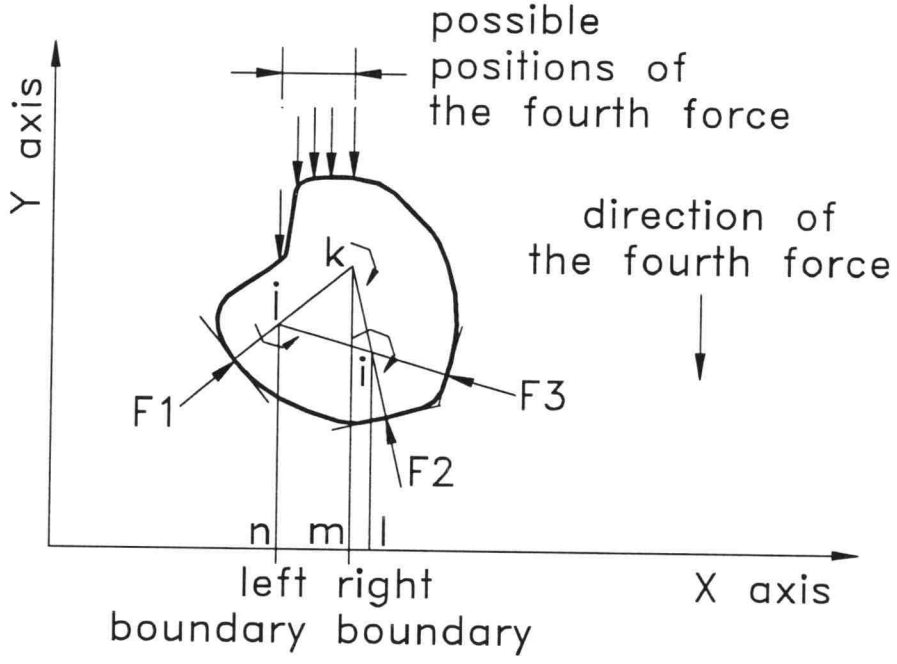

Figure 207 Possible positions of the fourth force

In Figure 207, the object is again restrained by three frictionless contact points with their restraint forces F_1, F_2 and F_3. A direction of the fourth force is as shown. The sense of the moment about every intersection caused by three contacts can be obtained by inspection or using equation shown earlier. In this case, directions of moments about point i, j and k are clockwise, counterclockwise and clockwise.

When intersection point *i* is considered, the fourth force must produce the contrary moment (counterclockwise in this case) to F_1 about this point. Therefore, the fourth force must be on the left of point *i*. When intersection points *j* and *k* are considered, the fourth force must be also on the right of point *j* and on the left of point *k*. The overlap among these areas can be obtained as discussed below.

Lines *il*, *jn* and *km* are drawn from the intersections to intersect X axis in the direction of the fourth force. Note that points *l*, *m* and *n* are on the X axis. In order to achieve the above requirements, the fourth force must be between points *n* and *m* which are set as the left and right boundaries. In this thesis, only surface forces will be concerned. Thus, the fourth force must be applied on the object surface. Consequently, the possible positions of the fourth force are found as any position on the surface that line of force falls inside the boundary. If the fourth force is at any of these positions, the four forces fulfill the requirement for total restraint.

When a position of the fourth force is given

In Figure 208, the object is restrained by three frictionless contact points with their restraint forces F_1, F_2 and F_3. The position of the fourth force is at point *O* as shown. In this case, directions of moments about point *i*, *j* and *k* are clockwise, counterclockwise and clockwise.

Lines *Oi*, *Oj* and *Ok* are drawn from the position of the fourth force to every intersection. When intersection point *i* is considered, the fourth force must be below line *Oi* to produce counterclockwise moment about this point. When

points *j* and *k* are considered, the fourth force must be also above line *Oj* and below line *Ok*. The overlap among these areas is enclosed by the angle *iOj* in which the fourth force must be applied. Limits of possible directions of the fourth force are shown as two arrows. If the fourth force is in any of these directions, the four forces fulfill the requirement for total restraint.

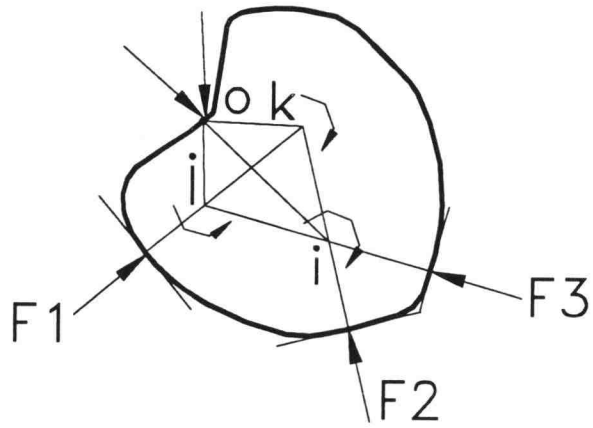

Figure 208 Possible directions of the fourth force

CHAPTER 3
HOW TO USE THE PROGRAM

3.1 Objective and program's overview

The Planar Restraint Design Assistant (PRDA) was written for determining positions of forces that restrain an object in a plane. The problem consists of an object and four restraint forces, three of which are specified and the fourth (called the external force) is to be determined. If direction of the external force is known, the program will find positions where the external force can be applied. If the user is not satisfied with these positions of the external force, any of the three specified forces can be altered until a satisfactory result is found.

If position of the external force is given, the program will find acceptable directions for it. As with the first situation, unsatisfactory results can be modified by altering the first three forces.

In the program, there are four windows. The upper left window shows the filename, the active force currently under user control and angles of forces relative to object surface normal. The middle window displays the object and restraint forces. The external force mode and calculation mode are shown in the middle left window. The lower window shows menus and program outputs. PRDA also provides the following features.

- A short message describing parameters needed is provided in most screen.

- User is allowed to use [ESC] to go back one step in most situations (except when input is a number).
- All parameters that can be set are always displayed on screen.
- Help is provided and can be reached from the main menu.

3.2 Defining an object

User must define an object before using other functions of the program. The program considers the object as a series of points on the object surface. The number of points in the series is automatically adjusted to between 160 and 200. The user can define the object in several ways as described below.

Creating an object by typing in a series of points

The program allows definition of object by entering a series of points. The first point can be located anywhere on the object surface. X and Y coordinates are separated by [ENTER]. After entering the coordinates of the first point, the program asks for the next point. Enter points in order around the object and end the sequence with [ENTER] (an empty line). As points are entered, lines are drawn on the screen and rescaled as necessary to fit in window.

In the "Define Object" menu, press [E] (Enter from keyboard) or [1] (Choice 1) to key in a series of points. The series is stored with a user specified filename.

Reading an object definition from a data file

The file must contain the coordinates from the first to the last point. There are two formats, "TST" and "SRF". In the TST format, the coordinates of the first point are stored in the first line. X and Y coordinates are separated by comma or space. The next line contains the coordinates of the next point and so on. The user can view the TST formatted file, "Random.tst" for example, by using "DOS EDITOR". The SRF formatted file is a binary file with each coordinate stored as a four-byte real number. An example of the SRF formatted file is "Random.srf".

In the "Define Object" menu, press [R] (Read from file) or [2] (Choice 2) to command the program to read data from a file. The list of TST and SRF formatted files will show at the bottom of screen. Users can use arrow up or down keys to highlight the desired filename and then press [ENTER] to read it. The program also allows users to type filename. After filename is entered, the program draws the object in the middle window and goes to the next routine.

Defining an object by reading a picture file

The program also has the capability to read a gray-scale image file. After the picture is read into the program, the program finds a contour line of the object and then transforms it to a series of points. The series is also saved in the file named "Obj.srf". Rename this SRF formatted file using DOS command to use it directly next time.

In a gray-scale image file, rows are stored in order from top to bottom and each row is stored in order from left to right. There is one byte per pixel. Thus the gray level

ranges from 0 to 255. There is no header and no punctuation. The user has to tell the program the size of a picture before loading.

By making use of this, the user can draw an object in a drawing program, AutoCAD® for example, and output it as a PCX file. There are many commercial programs that can convert a picture to a gray-scale image. An example of the commercial software is PMAN.

In the "Define Object" menu, press [R] (Read from file) or [2] (Choice 2) to read from a file. The user must type filename. The picture file extension must be neither TST nor SRF. After filename is entered, the program asks for the size (max 125×125) of the picture.

3.3 Mode of the external force

The program performs two types of calculation. One is for a given direction of external force while the other is for a given position of external force (see Chapter 2). With a given direction of external force (mode 1), the program will find locations where it could be applied. On the contrary, with a given position of external force (mode 2), the program will find directions it could be applied in.

After an object is defined at the beginning of the program, the program goes to the "Define External Force" routine. When "Mode 1" is selected, the program asks for the external force direction. After the user defines the external force, the program goes to the main menu.

3.4 Main menu

There are six choices in the main menu. The first two, "Define Object" and "Define External Force", have already been discussed in section 3.2 and 3.3. These choices can be changed at any time from the main menu. Choice 3 is to perform calculation and show result. Choice 4 is to set value for friction. Choice 5 is for help. These choices are detailed below. Finally, choice 6 is to finish the program and return to DOS prompt.

3.5 Positions of restraint forces

Three restraint forces are distinguished by colors, green cyan and red. Whenever a new object has been selected, position of green restraint force is set at the first point and positions of cyan and red restraint forces are equally spaced around the object.

Users can set these positions in the "Analyze force and show result" routine. Every restraint force direction is initially set perpendicular to object's surface. Users can change these directions in the "Set Parameters" routine.

3.6 Finding positions of restraint forces

The "Analyze force and show result" routine finds positions of all restraint forces. In this routine, user can move each restraint force along the object surface and place it in the designed position. The program then determines the positions or directions of the external force (depending on the external force mode). The user can reposition each

restraint force to get the desired external force positions or directions.

There are many commands in the "Analyze force and show result" routine to assist the user to find each restraint force position. Get into the "Analyze force and show result" routine by pressing [F] (Force) or [3] (Choice 3) in the main menu. The details of all function keys are discussed below.

Moving each restraint force along an object surface

Only one restraint force at a time can be moved around the object. The active force color that shows in the upper left window and a circle at the tail of the force vector indicates the force that can be moved.

The user can move the active force by pressing [+] or [-]. The key [+] will move the active force in one direction while the key [-] will move it in the other direction.

Changing the active force

Only the active force can move along the object surface. Users can change the active force by pressing [F].

Increasing or decreasing the movement step size

[PAGE UP] will increase the step and [PAGE DOWN] will decrease it.

Switching an object from inside to outside

The program allows users to change internal/external boundary by pressing [alt]+[F]. It initially considers the boundary as the outside of the object profile. This function changes boundary to inside the object. In this case, all forces are applied from the inner side of the profile.

Calculating when restraint forces are positioned

When three restraint forces are placed in the desired positions, user can press [C] to calculate position or orientation of the fourth force. The result depends on mode of the external force.

For the external force mode 1, the result will be where the external force could be applied with specified direction. These positions are shown by yellow arrows in the middle window with the object profile.

For the external force mode 2, external force directions will be determined. The program will find the possible directions for all points on the object and show result in a graph at the bottom of screen. On the graph, the horizontal axis represents location of points on the object while the vertical axis is direction of the external force relative to object surface normal.

For a given object and given positions of restraint forces, the range of possible external force directions is between the curves formed by yellow and blue dots. Vertical cyan line shows position of the external force on the object. The angle between the two arrows at the point of application of the external force on the object (brown

square) also shows the range of directions of the external force.

Changing run mode between static and dynamic

The program starts in "Dynamic Run Mode" which recalculates external force position whenever the active force is moved. In "Static Run Mode", [C] must be pressed each time the external force is to be calculated. The active restraint force moves faster in Static run mode, but trends of changes in restraint forces can be more easily seen in Dynamic run mode. Run mode will switch to Static when [alt]+[S] is pressed and Dynamic when [alt]+[D] is pressed.

Displaying a list of function keys when it disappears

When the external force mode is 2, and graph is shown on screen, the list of function keys can be displayed again by pressing [M]. To get back to the graph, press [C] (Calculate).

Error message

If three forces intersect at a point, they cannot restrain the object from rotation. In this case, the program displays an error message as shown below.

"When three restraint forces intersect at a point, the object can rotate about the point of intersection. Please try another configuration of the restraint forces."

Leaving the "Analyze force and show result" routine

Press [X] or [ESC] to exit this routine. The program will return to the main menu.

3.7 Setting Parameters

Users can define the angle between each restraint force and surface normal. Furthermore, users can also define maximum angle between the external force and surface normal. At the main menu, press [P] (Parameter) or [4] (Choice 4) to get into the "Set Parameters" routine. The lower window will show functions in this routine. The details of all functions are discussed below.

Defining the maximum angle of an external force

- **For external force mode 1**

The maximum angle of the external force limits possible positions of the external force to those with absolute value of angle to surface normal less than or equal to limit.

- **For external force mode 2**

If there is friction, the external force direction could be any direction in the friction cone. Angle limit is shown as a white band on the graph and restricts the angular range shown at the external force position on the object. To

set the maximum angle of the external force, press [1] in the "Set Parameters" menu. Input must be between 0° and 90°.

Defining the angle between each restraint force and the object surface normal

The default value is zero degree (perpendicular to the object surface). Since three restraint forces are distinguished by color, each restraint force angle is indicated by the color of the arrow, shown in the "Set parameters" menu.

Choices 2 through 4 in "Set Parameters" menu is to set these angles. Input must be between -90° and 90°.

Leaving the 'Set parameters' routine

Press [R] (Return) or [5] (Choice 5) to exit this routine. The program will return to the main menu.

3.8 Help

By pressing [H] in the main menu, help routine will come on screen. The routine displays a text file describing the details of the program. Use arrow up, arrow down, Page up or page down to move through the document.

3.9 Leaving the program

Press [X] in the main menu to finish the program.

3.10 Examples

Example 1

A jig is to be designed to hold a landing gear door hinge during polishing process. In the process, the applied force direction is -135 degrees while its location can be anywhere on the flat edge (Figure 301).

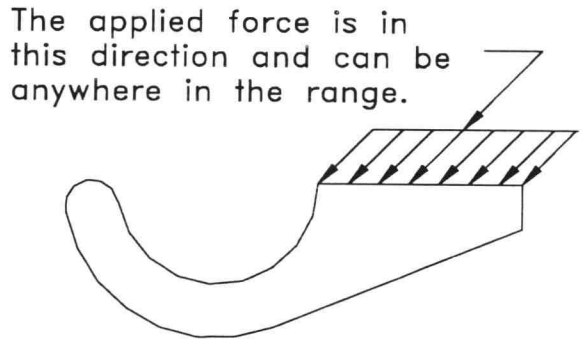

Figure 301 The profile of the hinge

A profile of the landing gear door hinge was initially drawn in AutoCAD and then transformed to a gray-scale image with a size of 144X96 (using PMAN). The object was defined as the hinge by entering a file as Hinge.gry (a gray-scale image file) and its size. The external force was set to mode 1 with the direction of -135.

Three restraint forces were placed at the positions shown in Figure 302. With this configuration of the restraint forces, the program found the possible positions of the external force as shown.

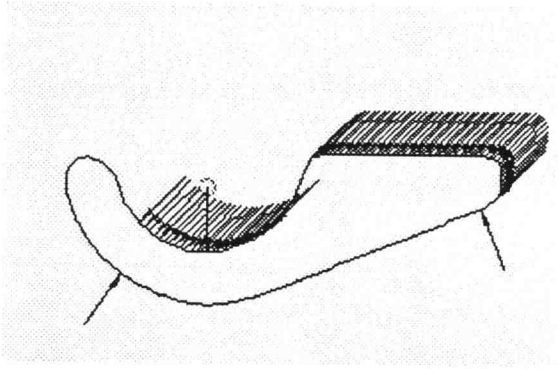

Figure 302 The result from the program

Although the four forces including the applied force, can prevent the hinge from moving, three restraint forces alone cannot do it. This problem is important because the applied force is not always present. The fourth frictionless contact point is added to solve this as discussed below.

The fourth contact can be applied at any position perpendicular to the surface but not where the applied force can be. When external force mode is 2 and external force angle limit is 0°, the position of the fourth contact was found as shown in Figure 303.

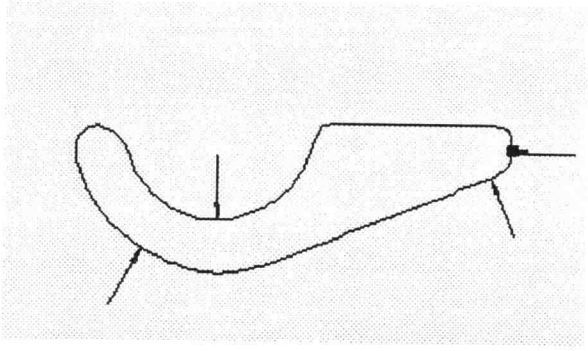

Figure 303 Position of the fourth contact

The first three restraints play important roles during the polishing while the fourth restraint is for any unexpected force that might exists when the applied force is absented. The jig is as shown in Figure 304. Friction at contacts is unnecessary for clamping the hinge.

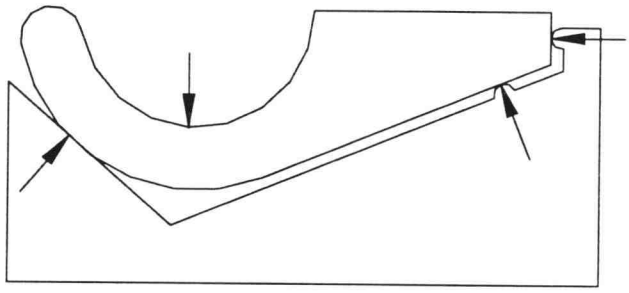

Figure 304 The positions of point contacts of the jig

Example 2

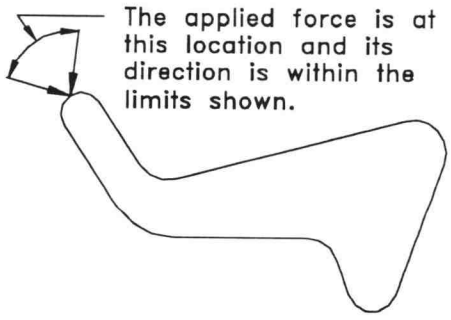

Figure 305 The profile of the clutch lever

Positions of restraints are to be chosen to hold a clutch lever against the applied force. The location of the applied force is as shown in Figure 305 but its direction can be anywhere within limits shown.

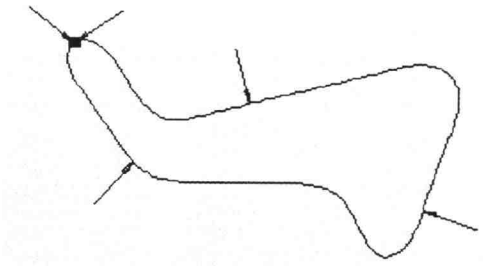

Figure 306 The result from the program

In the "Analyze force and show results" menu, the position of the external force (indicated by two arrows with a small box at their heads) was moved to the position shown in Figure 306. With positions of three restraint forces as shown, program found limits of the external force direction. However, this range of the external force directions does not satisfy the problem.

Many configurations of the three restraint forces were tested until a satisfactory result was obtained as shown in Figure 307. The four forces including the external force prevent the lever from sliding and turning.

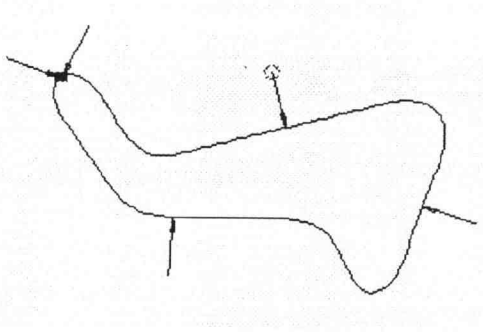

Figure 307 The program shows that these restraints can hold the lever against the applied force

CHAPTER 4
HOW THE PROGRAM WORKS

The program source code was written in Microsoft QuickBasic and runs directly on MS-DOS with IBM or its compatible computers. The program consists of three subprograms as follows:

1. PRDA
2. GRP
3. HELP

The program starts with PRDA (Planar Restraint Design Assistant) which can call the others, GRP and HELP. Details of these subprograms are described below.

4.1 PRDA

This program consists of 7 major subroutines as shown below:

1. Main
2. SetObject
3. SetEF
4. SetFriction
5. Analysis
6. Help
7. Calculation

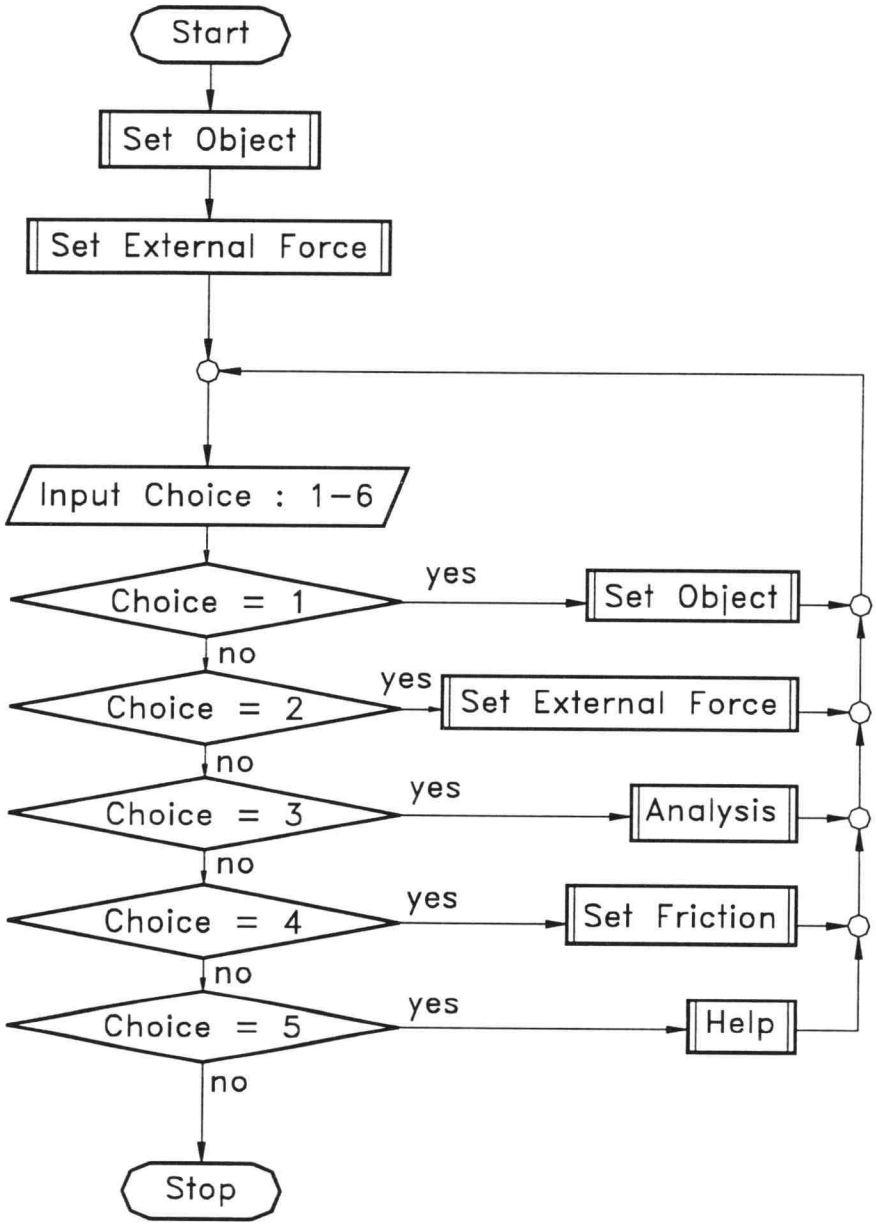

Figure 401 Block diagram of Main

Main

The PRDA program starts in Main in which all functions and subroutines are declared. The program begins with calling SetObject and SetEF and then goes to the main menu. The menu asks for a choice with the list of operations shown at the bottom of screen. When a choice is chosen, the program calls the corresponding routine and returns to the main menu after a called subroutine is finished. If choice 6 is selected, the program is terminated and returns to DOS prompt. These subroutines and their operations are discussed below. Block diagram of Main is shown in Figure 401.

SetObject

The SetObject subroutine is to define object's contour. The program considers the contour as a series of points (all forces must be applied at these points). As a result, users can define the series in four ways as follows:

1. Enter from keyboard
2. Read from TST (ASCII) formatted file
3. Read from SRF (Binary) formatted file
4. Call GRP program to convert a gray-scale image to a series; details in GRP

The subroutine first asks the user how to define object, entering from keyboard or reading from a file. For entering from keyboard, the routine asks for coordinates of the first point, the second point, and so on until input is empty. For reading from a file, the routine asks for a file name and then checks for its existence. If this file does not exist, the program goes back to ask how to define object again. The routine also shows TST and SRF formatted files at the bottom of screen. If user enters the file name from

keyboard, the program checks its format. When the extension of the file is neither SRF nor TST, the program considers the file as a gray-scale image and afterwards calls the GRP program to convert it to a series.

When the file is a gray-scale image file, the routine asks for size of the image and then saves the image filename and its size to a file named "File.dat" to be shared with the GRP program. Subsequently, GRP reads "File.dat" for the file to be converted. After the image is transformed to a series of points, GRP saves the series as "Obj.srf" and returns to PRDA. Finally, PRDA opens "Obj.srf" for a series of points to be used in its program.

Since all forces must be applied at a point in the series, number of points should be as large as possible. The minimum number of points was selected as 160. When number of points is fewer than the minimum, the routine generates points so that it exceeds the minimum.

In this process, the program finds the perimeter of the object. A variable, Step, is defined as this length divided by the minimum number of points. When distance between any two consecutive points is greater than Step, points are generated between those two points in such a way that the distance between any two consecutive points after adjustment does not exceed Step. As a result, the number of points is more than the minimum of 160 and the generated points are evenly spaced.

The upper limit on number of points was chosen as 200. When number of points is more than the maximum, the program finds a new series of 200 points. If the number of points before adjustment is C, point i of the new series is point $i*C/200$ of the original series. In this way $C-200$ points are eliminated.

After the series of points is adjusted, the routine draws the object and then calls the SetAngle subroutine. This routine finds angle perpendicular to the object surface for all points in the series. The angle of point *i* can be found as discussed on the following page.

Given: Coordinates of points *i-1*, *i* and *i+1*

Find: The direction perpendicular to the object surface of point *i*

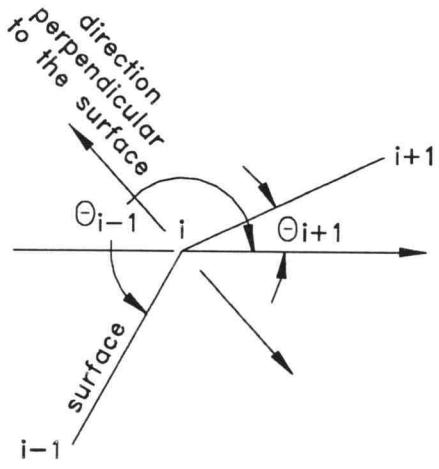

Figure 402 Direction perpendicular to the surface of point *i*

The directions from point *i* to point *i+1* and from point *i* to point *i-1* are found as

$$\theta_{i+1} = \tan^{-1}\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$$

$$\theta_{i-1} = \tan^{-1}\left(\frac{y_{i-1} - y_i}{x_{i-1} - x_i}\right)$$

where: θ_{i+1} is the direction from point *i* to point *i+1*

θ_{i-1} is the direction from point *i* to point *i-1*

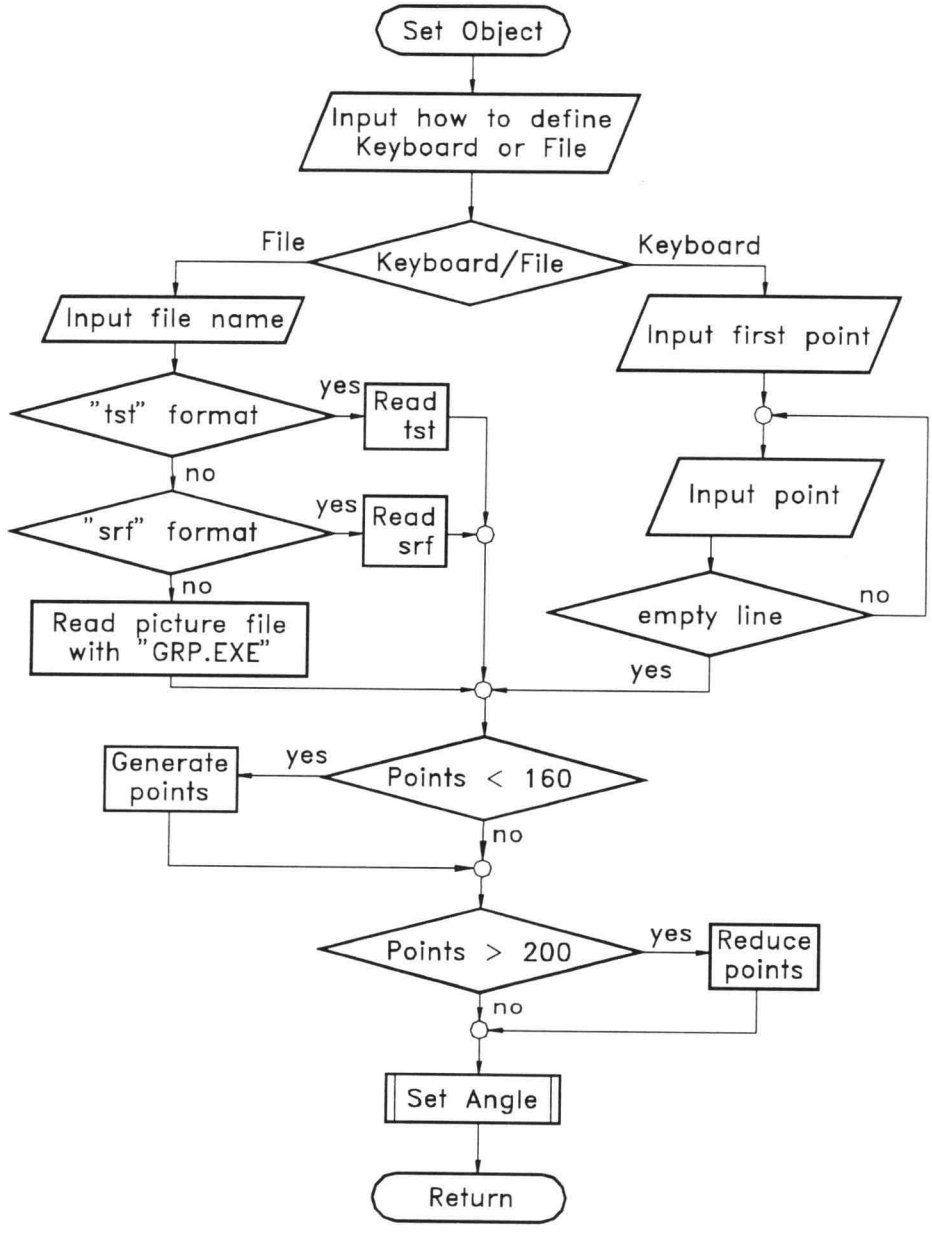

Figure 403 Block diagram of SetObject subroutine

The direction of point *i* is the middle direction between θ_{i+1} and θ_{i-1} and can be found as

$$\theta_i^* = \frac{\theta_{i+1} + \theta_{i-1}}{2}$$

where: θ^*_i is the direction perpendicular to the object surface of point *i*

However, there are two directions that can be the direction of point *i*. The direction of point *i* is the one that close to the direction of point *i-1* (the difference is less than 180°). Therefore, the directions of all points depend on the direction of the first point which is initially set as towards the outside of the object.

Finally, the program returns to the main menu after the angles are set. Block diagram of the SetObject subroutine is shown in Figure 403.

SetEF (Set External Force)

The SetEF subroutine allows user to set mode of external force (see Chapter 2). The routine starts with asking for mode of the external force. The active character keys are [1], [2] and [Esc]. If [1] (specify the external force direction) is selected, the routine asks for the external force direction and subsequently checks input as follows:

- If input is not a number, ask for new input
- If input is empty, set input to zero

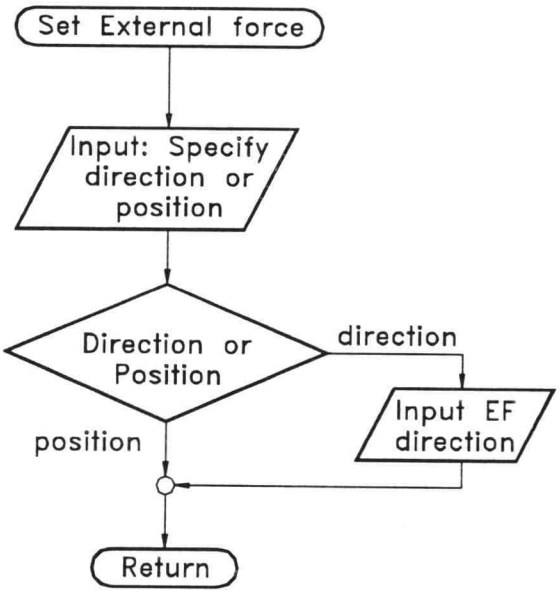

Figure 404 Block diagram of SetEF subroutine

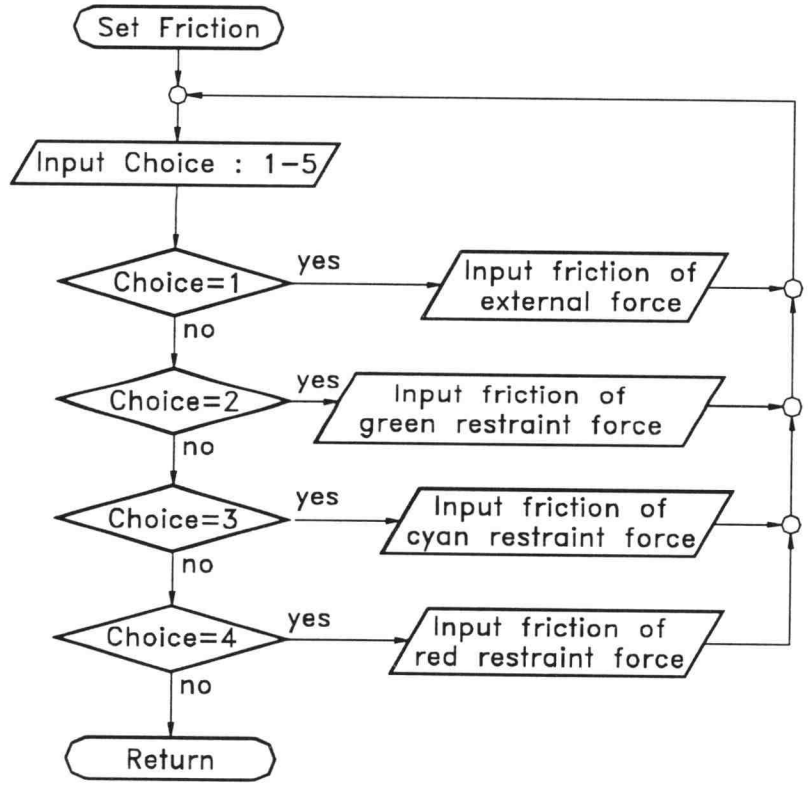

Figure 405 Block diagram of SetFriction subroutine

On the other hand, if [2] (specify the external force position) is selected, the routine asks no more question and returns to the main menu. If [Esc] is pressed instead, the program returns to the main menu. Block diagram of SetEF subroutine is shown in Figure 404.

SetFriction

The SetFriction subroutine allows users to set angle between each restraint force and object surface normal. The routine begins with asking for choice from those shown at the bottom of screen. The active character keys are [1], [2], [3], [4], [5], [R] and [Esc]. When [1] to [4] are selected, the routine asks for the angle of force corresponding to the selected choice. The routine also checks input as follows:

- If input is not a number, ask for new input
- If input is not between 0° and 90° (choice 1), ask for new input
- If input is not within ± 90° (choice 2-4), ask for new input
- If input is empty, set input to zero

After the angle is set, the routine goes back to ask for choice again. The last three characters, [5] [R] and [Esc], all finish the routine and return to the main menu. Block diagram of the SetFriction is shown in Figure 405.

Analysis

The purpose of the Analysis subroutine is to find positions of restraint forces. The routine starts with waiting for a command key. Whenever a key is pressed, the routine will do the operation corresponding to that key and subsequently check run mode. If the run mode is Static, the program goes back to wait for a command key again. If the run mode is Dynamic, the program determines results by calling the Calculation subroutine and afterwards, goes back to wait for a command key. The command keys and their operations are:

X and **Esc**: Back to the main menu.

C: Find results by calling Calculation subroutine.

<alt> F: Call SetAngle subroutine to redefine a set of angles. The new angles are 180 Degrees away from the old ones. This function is to redefine internal/external boundary of an object.

PgUp: Increase the step by one.

PgDn: Decrease the step by one.

+: Change position of the active restraint force in positive direction by the step.

-: Change position of the active restraint force in negative direction by the step.

F: change the active force as shown below.

If the active force is green, change it to cyan.

If the active force is cyan, change it to red.

If the active force is red and mode of the external force is 1, change it to green.

If the active force is red and mode of the external force is 2, change it to brown (representing position of external force).

If the active force is brown, change it to green.

<alt> D: Set run mode to Dynamic.

<alt> S: Set run mode to Static.

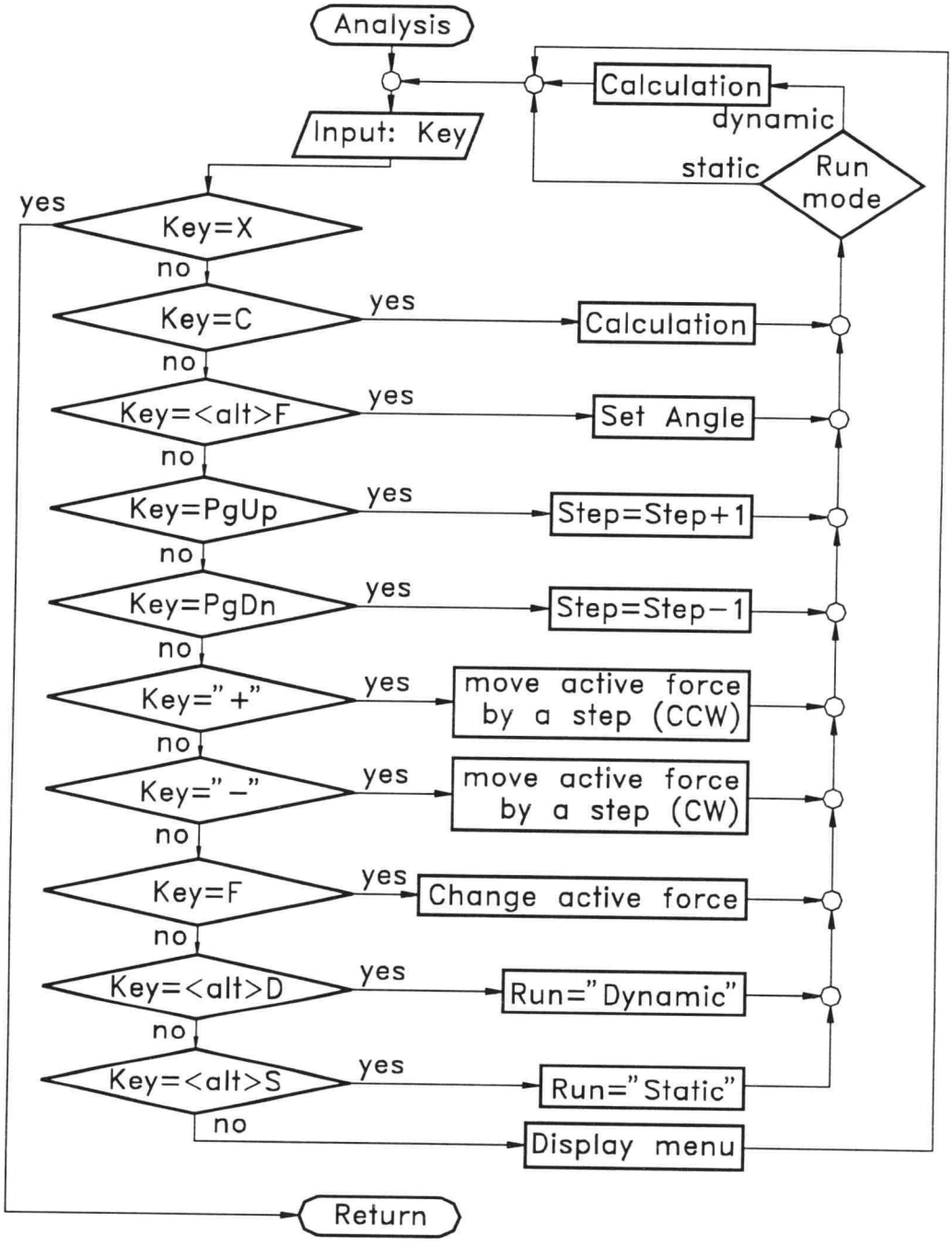

Figure 406 Block diagram of Analysis subroutine

Other keys: Display list of command keys.

When this routine is finished, the program returns to the main menu. Block diagram of the Analysis subroutine is shown in Figure 406.

Help

The Help subroutine calls the Help program; details in the Help program. Help displays "Program User's Guide" on screen along with commands to move through the document. When Help is finished, the program returns to the main menu.

Calculation

The Calculation subroutine uses parameters set from other subroutines to determine results. Block diagram of the Calculation subroutine is shown in Figure 407. The routine starts with determining all intersections of restraint forces (see Chapter 2).

After all intersections are obtained, the program checks whether three restraint forces intersect at a point or not. The program considers that three intersections are at the same point when they are too close (depending on size of the object). If an object can fit in a box of width W and height H, three intersections are too close when the distance between any two intersection is less than variable L which should be large enough to cover error from transformation of an image to a series of points. However, if it is too large, the program considers three forces

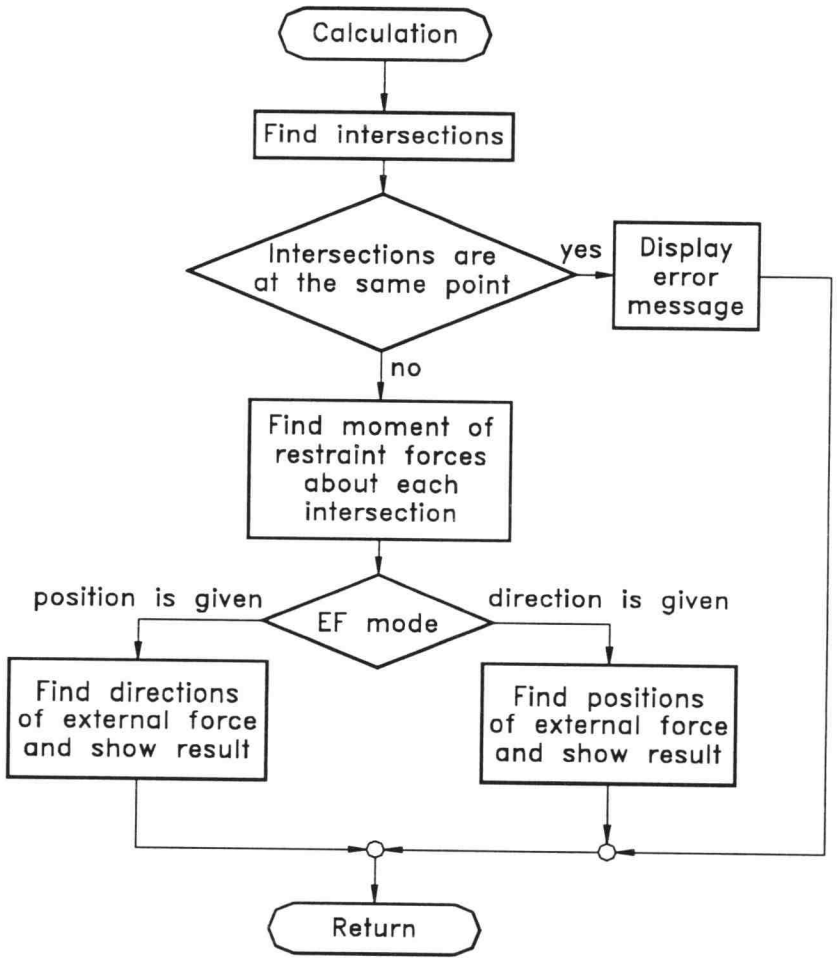

Figure 407 Block diagram of Calculation subroutine

intersect at a point even when the distance between any two of the intersection points is large but smaller than L.

The number of L came from the test of the circular object "Cir.tst", initially drawn in AutoCAD®. Since this object is not exactly circular because of error from the transformation process, L should be the smallest number such that the program considers three intersections are too close for all configurations of the three restraint forces and is chosen as the greater of $0.037615W^2$ and $0.080909H^2$. If the three intersections are too close, the program displays an error message (see Chapter 3) and goes to the end of the routine. If not, the program finds the moment of the three restraint forces about all intersections (see Chapter 2).

Results depend on mode of the external force; details in Chapter 2. The program checks mode of the external force and subsequently determines results as discussed below.

- **When external force direction is specified (mode 1)**

As discussed in Chapter 2 (Moment method), the possible positions of the external force are any point on the surface that is between the left and right boundaries. The program checks all points in the series for all possible external force positions. Furthermore, the direction of the external force must be also within the limit of angle of the external force (see Chapter 3). Consequently, the possible external force positions are found as any point in the series that falls inside the boundary and in the limit of angle of the external force. The program draws yellow arrows, representing external force positions, in the middle window of screen with the object contour. If the program finds no

possible external force position, the message "NO SOLUTION" is shown in the middle window.

- **When external force position is specified (mode 2)**

As discussed in Chapter 2 (Moment method), the external force direction must be such that it produces dissimilar moment from the moment produced by three restraint forces. In addition, the external force direction must be also within the limit of angle of the external force (see Chapter 3). The program determines a range of external force directions not only at the specified point but at all points in the series. In the middle window of screen, the program displays two arrows representing maximun and minimum directions of the external force at the specified position on the object profile. The range of external force directions of each point is also shown in a graph at the bottom of screen. The direction of the external force is in the object surface coordinate frame (intrinsic) where zero degree direction is the inward pointing normal to the object surface. If the program finds no possible external force position, the message "NO SOLUTION" is shown in the middle window.

4.2 GRP

The GRP program was developed to be used with the PRDA program. The purpose is to convert a gray-scale image (see Chapter 3) to a series of points on the surface. GRP gets information about an image from a file named "File.dat" and outputs a series of points to a file named "Obj.srf". The purpose of these two files is to communicate with PRDA.

0	0	0	0	0	0	0
0	0	15	15	15	0	0
0	15	15	15	15	15	0
0	15	15	10	15	15	0
0	15	15	15	30	15	0
0	0	0	15	15	15	0
0	0	0	0	0	0	0

Figure 408 A gray-scale image

B	B	B	B	B	B	B
B	B	O	O	O	B	B
B	O	O	O	O	O	B
B	O	O	O	O	O	B
B	O	O	O	O	O	B
B	B	B	O	O	O	B
B	B	B	B	B	B	B

Figure 409 The object pixels

	1	2	3	4	5	6	7
1	B	B	B	B	B	B	B
2	B	B	C	C	C	B	B
3	B	C	C	O	C	C	B
4	B	C	O	O	O	C	B
5	B	C	C	C	O	C	B
6	B	B	B	C	C	C	B
7	B	B	B	B	B	B	B

Figure 410 The contour pixels

First, the program opens "File.dat" for the filename of the gray-scale image and its size. If this file does not exist, the program returns to PRDA. In the image, the color of the upper left pixel is considered the color of the background. The program compares color of each pixel to the background color to find object pixels. Color of the object is any color different from the background color. The contour pixels are found as any object pixel next to a background pixel(s). An example of the way to find the contour pixels is shown below.

Given: a gray-scale image as shown in Figure 408

The background color is the number in the upper left pixel, 0, in this case. The object pixel is, therefore, any non-zero pixel. Since any pixel with color different from the background color is an object pixel, there must be no noise in the image. In Figure 409, the background pixels are marked "***B***" and the object pixels are marked "***O***". This transformation is done with a linear search through the file.

Contour pixels are any object pixel next to the background pixel(s) connecting to it at least a corner. In Figure 410, the contour pixels are marked "***C***" while "***B***" and "***O***" are again for the background and object pixels.

The program searches for any contour pixel to define the first point of the series. The search begins from the middle left pixel and goes to the right. If there is no contour pixel in this row, the program continues searching in the next lower row and goes to the first row of the image after the last row is searched. The program stops search whenever any contour pixel is found. This pixel is the first point of the series. The second point of the series is found as shown on the following page.

Given: The first point is at pixel number 5 (Figure 411)

1	2	3
4	5	6
7	8	9

Figure 411 How the program picks the second point

The second point is the contour pixel next to the first point in the priority as follows:

1. Pixel number 2
2. Pixel number 4
3. Pixel number 6
4. Pixel number 8
5. Pixel number 1
6. Pixel number 3
7. Pixel number 7
8. Pixel number 9

The third point is the contour pixel next to the second point but not the same pixel as the first point. How the program picks the third point is discussed as follows:

Given: The second point is at pixel number 5 (Figure 412)

1	2	3
4	5	6
7	8	9

Figure 412 How the program picks the third point

When the first point is at the side (pixel number 2, 4, 6, 8), the priority of the pixel to be the third point (if it is a contour pixel) is as follows. The pixel opposite to the first point is the first priority. The other two pixels at the sides that are neither the first point nor opposite to the first point will be the second and the third priorities. Two pixels at the corners that are not next to the first point will be the fourth and fifth priorities. The other two pixels at the corners and next to the first point are the sixth and seventh priorities. The example is shown below.

When the first point is at pixel number 2, the priority of the pixel to be the third point is as follows:

1. Pixel number 8
2. Pixel number 4
3. Pixel number 6
4. Pixel number 7
5. Pixel number 9
6. Pixel number 1
7. Pixel number 3

When the first point is at the corner (pixel number 1, 3, 7, 9), the priority of the pixel to be the third point is as follows. Two pixels at the sides that are not next to the first point will be the first and the second priorities. The pixel opposite to the first point is the third priority. Two pixels at the other corners are the fourth and fifth priorities. The other two pixels on the sides and next to the first point are unable to be the third point. The example is shown below.

When the first point is at pixel number 7, the priority of the pixel to be the third point is as shown on the following page.

Point Number	Coordinate	
	X	Y
1	2	4
2	2	3
3	3	3
4	3	2
5	4	2
6	5	2
7	5	3
8	6	3
9	6	4
10	6	5
11	6	6
12	5	6
13	4	6
14	4	5
15	3	5
16	2	5

Figure 413 A series of points found from the contour

1. Pixel number 6
2. Pixel number 2
3. Pixel number 3
4. Pixel number 9
5. Pixel number 1

Each subsequent point can be found in a similar way. The series is complete when the point is found to be the same as the first point. Finally, the series is saved to the file named "Obj.srf".

For the image of the contour shown in Figure 410, the series can be found as shown in Figure 413. However, there may be more than one object in the image. In these cases, the contour will be found from the object that is discovered first by the search and only this object will be saved in "Obj.srf".

4.3 Help

The Help program is callable from the PRDA program to view a file named "Readme.txt" (see Chapter 3). First, the program opens "Readme.txt" and saves all lines in the file to an array variable. The program displays only 17 lines on screen at a time but allows users to change lines which are displayed using the command keys as shown below.

PgUp: Display the next 16 lines
PgDn: Display the previous 16 lines
↑: Display the previous line
↓: Display the next line
P: Copy "readme.prn" to printer
Esc: end the program

The program waits for the above command keys. Whenever a command key is pressed, the program performs operation corresponding to that key and then goes back to wait for a command key again until [ESC] is pressed to finish the program. When Help is finished, the program returns to PRDA.

CHAPTER 5
DISCUSSION AND CONCLUSIONS

In this thesis, PRDA was developed to find a range where the fourth force can be applied when three restraint forces are given. The program provides two modes of the fourth force, given its direction and given its position. For a given fourth force direction, PRDA shows a range of its positions in visual form with an object. For a given fourth force position, PRDA shows a graph of range of its directions, useful to find starting positions of restraint forces. However, the assumptions used in this thesis are:

- The object is restrained by point contacts represented by unidirectional restraint force.
- The restraint force is compression that has no limit on magnitude.
- The object is rigid
- The movement out of a plane is prevented by some unspecified agencies.

In addition, the program has limitations as follows:

- All forces must be applied at points in a series.
- Number of points in a series is not more than 200.
- The maximum size of an image file is approximately 125(row)×125(column) tested on 486 DX 50, 4 MB RAM.

Recommendations for future development of PRDA are:

- Perform an analysis of magnitudes of restraint forces. For a desired range of the fourth force, there may be many possible configurations of the three restraint forces.

Therefore, force magnitudes could be used to find the best configuration among the possible ones. In addition, the maximum of force magnitude can be used to design the strength of the three contacts.

The magnitude of restraint force can be determined using the moment equation shown in this report. However, positions and directions of all forces including the fourth force must be given. The program may allow the user to position the fourth force and subsequently finds magnitudes of the three restraint forces. Result can be shown in a graph having three bars for three restraint forces. Note that three restraint forces are distinguished by color. Hence, color of a bar can be used to indicate the corresponding restraint force while the height of the bar is the magnitude of the restraint force.

- Allow to create an object by reading the other formats of image files, Bitmap for example. The current version of PRDA has the capability to read only a gray-scale image file. Therefore, the user must save an image in this format. It is more convenient if PRDA can read the other formats of image files. Imagine the user uses the copy command in AutoCAD for Windows to copy an object to Clipboard and then paste it to PRDA directly.

- Allow to set friction for each restraint force. In the current version of PRDA, the user can specify the direction of restraint force relative to surface normal but it is still unidirectional force. It is more useful if restraint force can be in any direction within a friction cone specified by the user and PRDA finds results considering all possible directions of restraint force.

BIBLIOGRAPHY

1. Reuleaux, F.,(1876). The Kinematics of Machinery, Dover, New York

2. Kerr, D.R., Sanger, D.J., (1983). The analysis of kinematic restraint, Theory of Machines and Mechanisms

3. Hunt, K.H., (1978). Kinematic Geometry of Mechanisms, Clarendon Press, Oxford

4. Meriam, J.L.,Kraige, L.G., (1992) Engineering Mechanics, Wiley, New York

5. Walker, W.F., (1969). Beginner's Guide to Jig and Tool Design, Hart Publishing Company, Inc.,ew York

6. Erwin Kreyszig, (1988). Advanced Engineering Mathmatics, John Wiley & Sons, Inc., New York

APPENDICES

Appendix 1 Program Larning Guide

The Planar Restraint Design Assistant (PRDA) is written for determining positions of forces that restrain an object in a plane. The problem consists of an object and four restraint forces, three of which are specified and the fourth (called the external force) is to be determined. If direction of the external force is known, the program will find positions where the external force can be applied. If the user is not satisfied with these positions of the external force, any of the three specified forces can be altered until a satisfactory result is found.

If position of the external force is given, the program will find acceptable directions for it. As with the first situation, unsatisfactory results can be modified by altering the first three forces.

Starting PRDA

Change to the drive and the directory where PRDA resides on your computer. Assume that PRDA is stored in the directory TEMP on the C: drive. Type PRDA to run the program in this way:

C:\TEMP> PRDA <Enter>

You will see four windows on the screen. The upper left window shows the file name, the active force and angles of forces. The middle window is for displaying an object and restraint forces. The external force mode and calculation mode are shown in the middle left window. The lower window will appear as shown in Figure A1.

Defining an object

You can define the object two ways as stated in the window. The first is by drawing the object on screen. You have to know the coordinates of points at corners of the object and enter these points from keyboard. The alternative is by calling the data file that contains points on the object's surface. ***Square.tst*** is the example file that comes with the program (Figure A2).

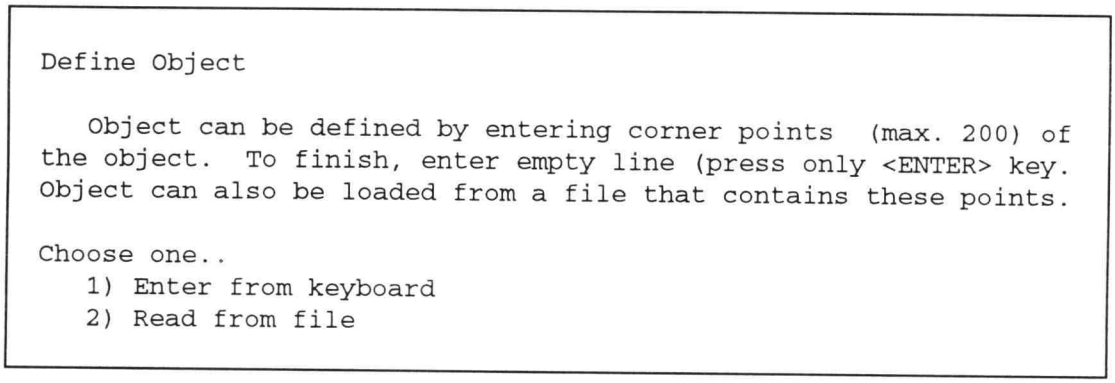

Figure A1 Define Object menu

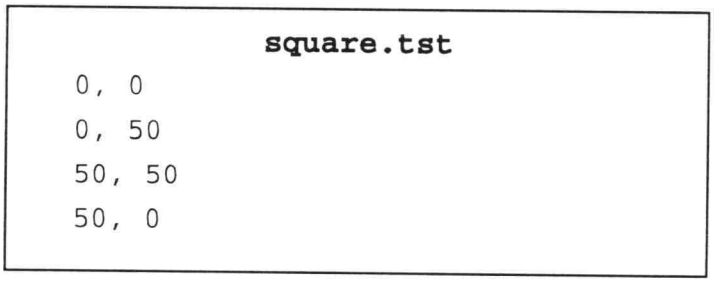

Figure A2 Example of TST file containing corners of a square

When the lower window appears as shown in Figure A1, do the following:

Press **"2"** to read an object from the data file

The lower window now lists data files and the program asks the name from those to be read. Do the following:

Type **"Random.tst" <Enter>**

The object will show in the middle window. Defining an object from keyboard and other file types are discussed elsewhere.

Defining external force

After the user defines an object, the program will go to "Define External Force" routine. The program then asks for external force mode (Figure A3).

```
External Force (EF)

     External force can be specified in either of 2 ways. Direction
of  EF can be  specified and program displays possible positions or
position can be specified and program displays possible directions.

Enter EF mode number  Mode 1: Specify EF direction
                      Mode 2: Specify EF position
```

Figure A3 Define External Force menu

The program performs two types of calculation. With a given direction of external force (mode 1), the program will find locations where it could be applied. With a given position of external force (mode 2), the program will find

directions it could be applied in. In mode 2, directions are shown both on the object and in graph at the bottom of screen. Do the following to specify the direction of external force:

Press **"1"** to tell the program that EF direction will be specified

Type **"270" <Enter>** for the direction in degrees of external force

Main menu

```
Main menu:  Please enter your choice?

1) Define Object:  To change an object
2) Define External Force:  To change EF mode
3) Analyze Force and show results:  Do calculation
4) Set Parameters:  To assign values for friction
5) Help:  To show details of each function
6) Exit:  End this program
```

Figure A4 The main menu

After the object and EF mode are defined, the program will go to the main menu which has six choices. The first two, "Define Object" and "Define External Force", have already been discussed. The user can go to "Define Object" to redefine the object or go to "Define External Force" to redefine the external force mode of calculation. Note that the middle left window will show the external force mode. Choice 3 and 4 will be discussed in this section. Choice 5,

"Help", is to call help routine. The last choice is to terminate the program. The lower window of the screen will appear as shown in Figure A4.

Analyzing restraint forces

Next, we will find the positions of restraint forces that will prevent the object from sliding or turning. Choice 3 in the main menu is to assist the user finding positions of restraint forces. Do the following:

Press **"3"** to begin finding the result.

The screen will be as shown in Figure A5. The commands used in this routine are also shown in the lower window of screen.

The program initially picks positions of restraint forces which can be changed in this routine. Directions of restraint forces are first set to be perpendicular to object's surface. Users can change these directions by using "Set Parameters" routine described elsewhere.

In this routine, the user can move each restraint force along the object's surface. The restraint force being moved (active force) has a small circle at its tail. Do the following:

Press **"+"** and hold until the green restraint force is at the position shown in Figure A6.

When pressing "+", the active force will move along the object's surface. The user can move this active force in the other direction by pressing "-" instead of "+".

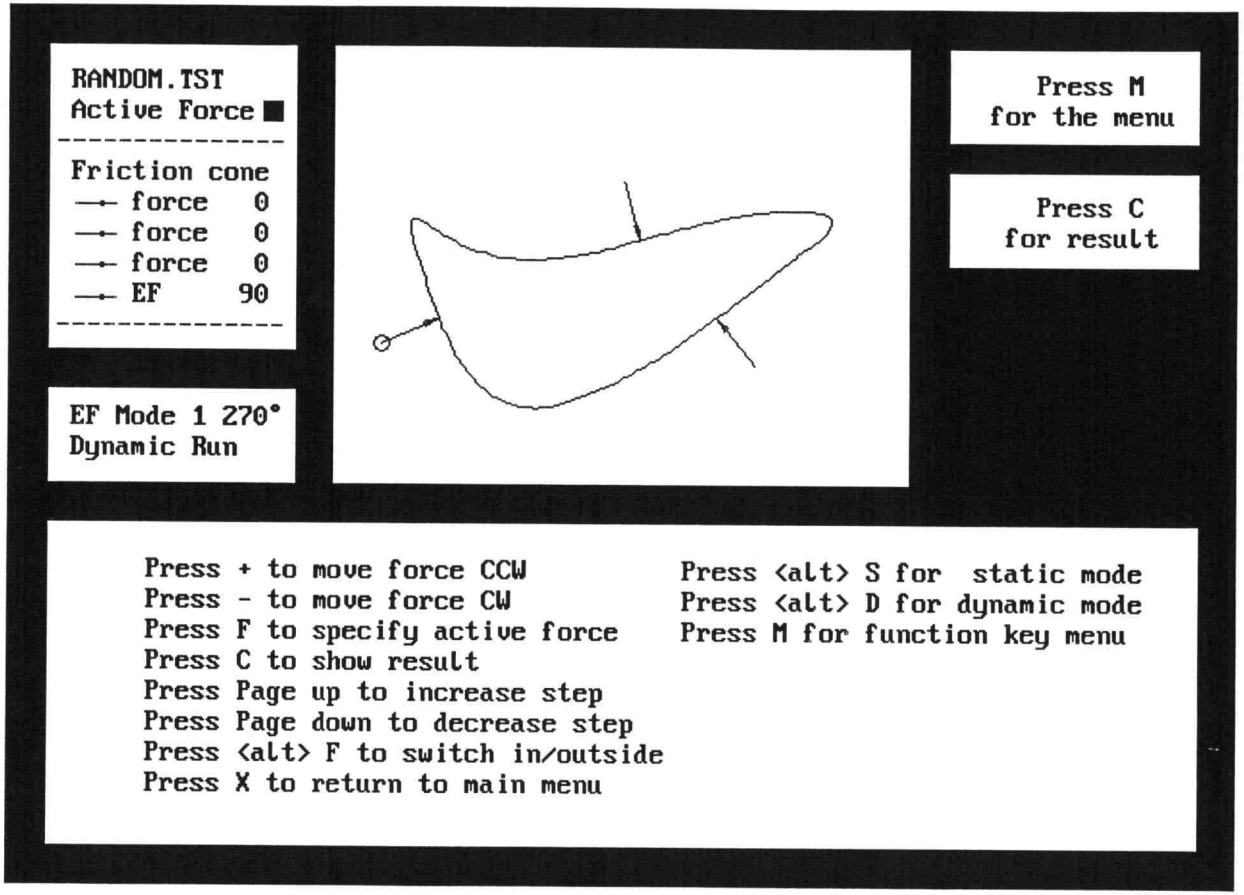

Figure A5 Analyze force and show result menu

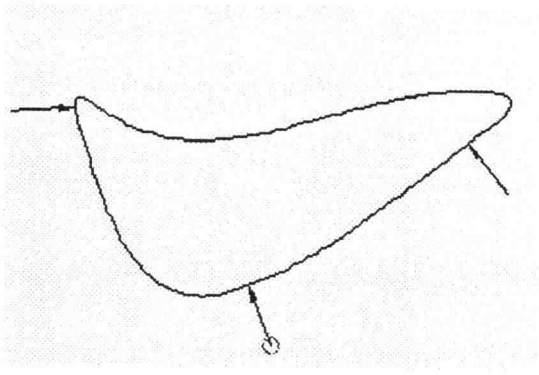

Figure A6 The screen showing the object and restraint forces

Press **"F"** to change the active force

Only one restraint force can move at a time. Since there are three restraint forces, the user must tell the program which force is active by pressing "F". Restraint forces are distinguished by color. The active restraint force is shown by the color of a small square box in the upper left window and by a circle at the tail of the force arrow.

Press **"+"** and hold until the cyan restraint force is positioned as desired

Press **"F"** to change the active force

Press **"+"** and hold until the red restraint force is positioned as desired

Dynamic or Static run mode

The program starts in "Dynamic Run Mode" which recalculates external force positions (or directions)

whenever the active force is moved. Users may reposition each restraint force until desired locations of external force are obtained. In "Static Run Mode", the active force can be moved without calculating the results. Do the following to change the calculation mode from "Dynamic Run" to "Static Run Mode".

Press **<Alt> S**

Now, user can move the active restraint force faster. When three restraint forces are positioned, press **"C"** to determine locations of external force. Do the following:

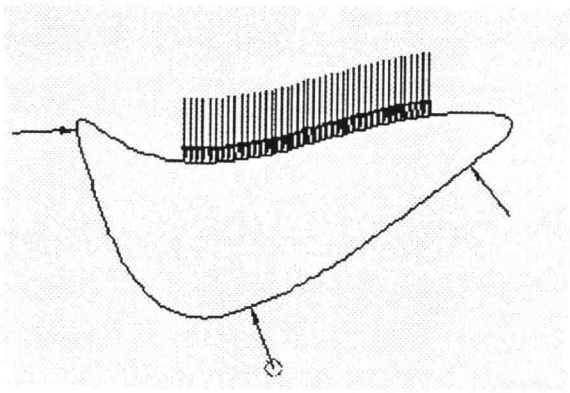

Figure A7 The possible locations of the external force

Press **"C"** to show the result.

When "C" is pressed, the program will determine positions of external force (Figure A7) and show these positions by yellow arrows. When the external force is at any location shown by the yellow arrows, the four restraint forces will be able to prevent the object from moving.

To change running mode back to Dynamic, press <alt> D".

Setting friction parameters

The program initially assumes that each restraint force is perpendicular to the object's surface but friction may make this unnecessary. The user can specify the angle between each restraint force and the object surface normal. Do the following to specify the angle of restraint forces.

Press **"X"** to exit from "Analyze force and show results" routine

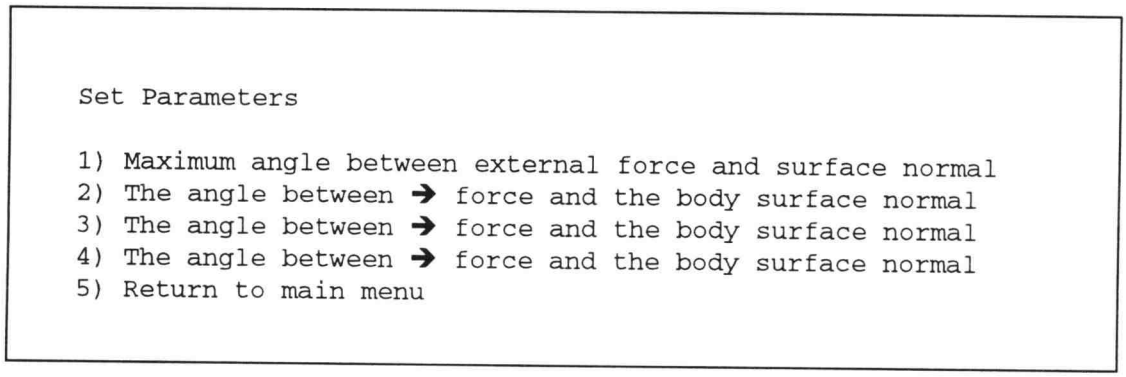

Figure A8 Set Parameters menu

The program will return to the main menu. The lower window of the screen will appear as shown in Figure A2.

Press **"4"** to set the angle of restraint forces

Choice 4, "Set Parameters", in the main menu allows user to assign values for angles between each restraint force and the object surface normal. After pressing "4" in the main menu, the lower window will appear as shown in Figure A8

In "Set Parameters" menu, there are five choices. The first is to assign value for limit of absolute value of angle between the external force and surface normal; must be positive and less than or equal to 90°. Choices 2 through 4 are to assign values for angles between restraint forces and the object surface normal; must be between -90° and 90°. Choice 5 is to exit the "Set parameters" menu and return to the main menu. Do the following to assign a value for the angle of each restraint force.

Press **"2"** to assign a value for the angle of green restraint force

Enter **"10" <Enter>** for angle between restraint force and body surface normal. Direction of the restraint force will be 10 degree counterclockwise from the object surface normal. Negative angles give rotations in the clockwise direction. Set angles for the other restraint forces in a similar way.

Press **"5"** to exit from "Set Parameters" menu

The program will return to the main menu. Go to "Analyze force and show results" routine again. Restraint forces are now at the specified angles. Press "X" when finished to return to the main menu.

Creating an object by entering a series of points

The current object was read from a data file containing points on the surface. The user can create an object by entering points on the surface in order around the object. Do the following to redefine the object:

Press **"1"** to redefine the object

When the lower window appears as shown in Figure A1, do the following:

Press **"1"** to create an object from keyboard

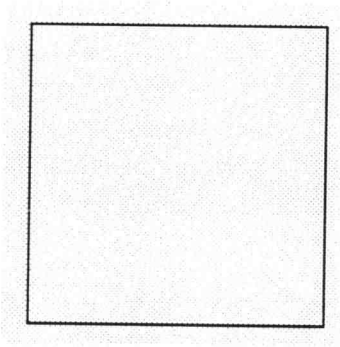

Figure A9 The square object

The lower window now asks for the coordinates of points on the object's surface. Enter the series of points as in Figure A2 to create a square object. Do the following:

Enter **"0" <Enter>** for X coordinate of the first point
Enter **"0" <Enter>** for Y coordinate of the first point
Enter **"0" <Enter>** for X coordinate of the second point
Enter **"50" <Enter>** for Y coordinate of the second point
Enter **"50" <Enter>** for X coordinate of the third point
Enter **"50" <Enter>** for Y coordinate of the third point
Enter **"50" <Enter>** for X coordinate of the fourth point
Enter **"0" <Enter>** for Y coordinate of the fourth point
Press **"<Enter>"** to end the series.

Now, the middle window shows the square object as in Figure A9 and the program asks for a filename for storing the series.

Enter **"test.tst" <Enter>** for the filename

Creating an object by reading a gray-scale image

After the object is defined completely, the program returns to the main menu. Go to "Define Object" again to redefine the object by reading a gray-scale image file.

In an image file, there are two colors, background and object. The program has the capability to convert an image to a series of points. Do the following:

Press **"1"** to redefine the object

When the lower window appears as shown in Figure A1, do the following:

Press **"2"** to read an image file

Type **"random.gry" <Enter>** for the name of the image

The lower window will appear as shown in Figure A10. Now, the program asks for the size of the image. This image has a size of 144×96. Do the following to enter the size of the image:

Type **"144" <Enter>** for number of image's columns
Type **"96" <Enter>** for number of image rows

When the transformation is completed. Do the following to return to the main menu:

Press **"<Enter>"**

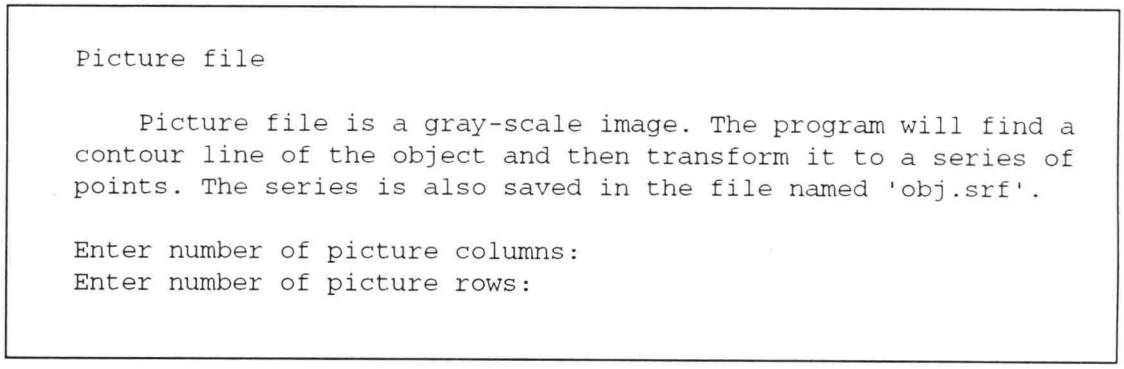

Figure A10 The image file screen

In the main menu, do the following to exit the program:

Press **"6"** to exit the program

Appendix 2 Program Source Code

Source Code of PRDA

```
'************************************************************
'* Title:       PRDA.BAS                                    *
'* Programmer:  Ratchatin Chancharoen                       *
'*              Graduate Student in Mechanical Eng. dpt.    *
'*              Oregon State University                     *
'* Date:        30 March 1994                               *
'* Purpose:     This program is to determine positions of   *
'*              forces that restrain object against the     *
'*              applied force. Calculation is based on the  *
'*              moment method.                              *
'************************************************************
DECLARE SUB parameters ()
DECLARE SUB prnfile (j!, k!, strg$)
DECLARE SUB help ()
DECLARE SUB StArrow (x!, y!, a!, c!)
DECLARE SUB SetFriction ()
DECLARE SUB SetEF ()
DECLARE SUB Calculation ()
DECLARE SUB Analysis ()
DECLARE SUB SetAngle ()
DECLARE SUB SetObject ()
DECLARE SUB appearance ()
DECLARE SUB arrow (x!, y!, a!, c!)
DECLARE SUB drwobj ()
DECLARE FUNCTION FitLimit! (maxa!, mina!, ul!, ll!)
DECLARE FUNCTION CheckIntersect! (x1!,y1!,x2!,y2!,x3!,y3!)
DECLARE FUNCTION chkpt! (ptx!, lx1!, lx2!, pty!, ly1!, ly2!)
DECLARE FUNCTION ReadNum! (Num$)
DECLARE FUNCTION fcolor! (force!)
DECLARE FUNCTION in$ (high!, low!, angle!)
DECLARE FUNCTION clsang$ (a!, b!)
DECLARE FUNCTION pst$ (x!, y!, a, p#)
DECLARE FUNCTION moment$ (a!, p#, x!, y!)
DECLARE FUNCTION pick! (a1!, a2!, la!)
DECLARE FUNCTION adjust! (a!)
DECLARE FUNCTION arctan! (x!, y!)
DECLARE FUNCTION max! (a!, b!)
DECLARE FUNCTION min! (a!, b!)

'shared variables
COMMON SHARED file$, chkrd$
COMMON SHARED rxx1, rxx2, ryy1, ryy2
COMMON SHARED status$, mod$
```

```
COMMON SHARED flip, c, ind, p, x, dyn, ans, xe, ye

'array variables
DIM SHARED x(1 TO 4), y(1 TO 4), a(1 TO 4)
DIM SHARED lf#(1 TO 4), mf#(1 TO 4), pf#(1 TO 4)
DIM SHARED insx(1 TO 3), insy(1 TO 3), torque$(1 TO 3)
DIM SHARED p(1 TO 3), px(1 TO 3), py(1 TO 3)
DIM SHARED cone(1 TO 4), colr(1 TO 20)
DIM SHARED dtx(1 TO 800), dty(1 TO 800), ang(1 TO 800)
DIM SHARED maxang(1 TO 200), minang(1 TO 200)
DIM SHARED high(1 TO 3), low(1 TO 3)

'main
     CALL parameters     'Set parameters at the begining
     CALL appearance
     CALL SetObject
     CALL SetEF

mbegin:
     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF

COLOR colr(2)  'main menu
LOCATE 19, 25: PRINT ": Please enter your choice ?"
LOCATE 21, 16: PRINT "1) Define Object"
LOCATE 22, 16: PRINT "2) Define External Force"
LOCATE 23, 16: PRINT "3) Analyze Force and show results"
LOCATE 24, 16: PRINT "4) Set Parameters"
LOCATE 25, 16: PRINT "5) Help"
LOCATE 26, 16: PRINT "6) Exit"

COLOR colr(3)
LOCATE 19, 16: PRINT "Main menu"
LOCATE 21, 16: PRINT "1)"
LOCATE 22, 16: PRINT "2)"
LOCATE 23, 16: PRINT "3)"
LOCATE 24, 16: PRINT "4)"
LOCATE 25, 16: PRINT "5)"
LOCATE 26, 16: PRINT "6)"

LOCATE 21, 26: PRINT "O"
LOCATE 22, 26: PRINT "E"
LOCATE 23, 27: PRINT "F"
LOCATE 24, 23: PRINT "P"
LOCATE 25, 19: PRINT "H"
LOCATE 26, 20: PRINT "x"

COLOR colr(1)
LOCATE 21, 32: PRINT ": To change an object"
LOCATE 22, 40: PRINT ": To change EF mode"
```

```
LOCATE 23, 49: PRINT ": Do calculation"
LOCATE 24, 33: PRINT ": To assign values for friction"
LOCATE 25, 23: PRINT ": To show details of each function"
LOCATE 26, 23: PRINT ": End this program"

DO    'main loop
      LET status$ = UCASE$(INKEY$)
      IF status$ = "O" THEN EXIT DO
      IF status$ = "F" THEN EXIT DO
      IF status$ = "E" THEN EXIT DO
      IF status$ = "H" THEN EXIT DO
      IF status$ = "P" THEN EXIT DO
      IF status$ = "1" THEN EXIT DO
      IF status$ = "2" THEN EXIT DO
      IF status$ = "3" THEN EXIT DO
      IF status$ = "4" THEN EXIT DO
      IF status$ = "5" THEN EXIT DO
      IF status$ = "6" THEN EXIT DO
LOOP WHILE status$ <> "X"

IF status$ = "O" OR status$ = "1" THEN CALL SetObject
IF status$ = "F" OR status$ = "3" THEN CALL Analysis
IF status$ = "H" OR status$ = "5" THEN CALL help
IF status$ = "E" OR status$ = "2" THEN CALL SetEF
IF status$ = "P" OR status$ = "4" THEN CALL SetFriction
IF status$ = "X" OR status$ = "6" THEN GOTO mend
GOTO mbegin

mend:

FUNCTION adjust (a)
'This function is to adjust the angle "a" to be in the range
'of 0 and 360 degree. For example, adjust(400)= 40

DO
      IF a >= 360 THEN LET a = a - 360
      IF a < 0 THEN LET a = a + 360
LOOP WHILE a >= 360 OR a < 0
LET adjust = a

END FUNCTION

SUB Analysis
'see Chapter 4
CONST pi = 3.141593

'setup
LET ind = INT(c / 2)
LET stp = 2
LET rmode = 1
```

```
      WINDOW SCREEN (0, 0)-(639, 479)
      COLOR colr(2)
      LOCATE 15, 5: PRINT "Dynamic Run"
      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (489, 19)-(621, 251), 6, BF
      LINE (490, 150)-(620, 250), 2, B
      LINE (490, 20)-STEP(130, 50), 0, BF
      LINE (490, 20)-STEP(130, 50), 2, B
      LOCATE 3, 68: PRINT "Press M"
      LOCATE 4, 65: PRINT "for the menu"
      LINE (490, 85)-STEP(130, 50), 0, BF
      LINE (490, 85)-STEP(130, 50), 2, B
      LOCATE 7, 68: PRINT "Press C"
      LOCATE 8, 66: PRINT "for result"
      WINDOW (rxx1, ryy1)-(rxx2, ryy2)
addf:
      IF dyn = 4 THEN LET dyn = 3
      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (21, 271)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 19, 10: PRINT "Press + to move force CCW"
LOCATE 20, 10: PRINT "Press - to move force CW"
LOCATE 21, 10: PRINT "Press F to specify active force"
LOCATE 22, 10: PRINT "Press C to show result"
LOCATE 23, 10: PRINT "Press Page up to increase step"
LOCATE 24, 10: PRINT "Press Page down to decrease step"
LOCATE 25, 10: PRINT "Press <alt> F to switch in/outside"
LOCATE 26, 10: PRINT "Press X to return to main menu"
LOCATE 19, 45: PRINT "Press <alt> S for  static mode"
LOCATE 20, 45: PRINT "Press <alt> D for dynamic mode"
LOCATE 21, 45: PRINT "Press M for function key menu"

COLOR colr(3)
LOCATE 19, 16: PRINT "+"
LOCATE 20, 16: PRINT "-"
LOCATE 21, 16: PRINT "F"
LOCATE 22, 16: PRINT "C"
LOCATE 23, 16: PRINT "Page up"
LOCATE 24, 16: PRINT "Page down"
LOCATE 25, 16: PRINT "<alt> F"
LOCATE 26, 16: PRINT "X"
LOCATE 19, 51: PRINT "<alt> S"
LOCATE 20, 51: PRINT "<alt> D"
LOCATE 21, 51: PRINT "M"

LINE (171, 21)-(469, 249), colr(14), BF
CALL drwobj
CALL arrow(x(1), y(1), a(1), colr(4))
CALL arrow(x(2), y(2), a(2), colr(5))
```

```
CALL arrow(x(3), y(3), a(3), colr(6))
FOR i = 1 TO 3
     FOR k = 1 TO c
          IF dtx(k) = x(i) AND dty(k) = y(i) THEN
               LET pp = k
          END IF
     NEXT k
     LET a(i) = ang(pp) + cone(i)
     LET lf#(i) = COS(a(i) * pi / 180)
     LET mf#(i) = SIN(a(i) * pi / 180)
     LET pf#(i) = mf#(i) * dtx(pp) - lf#(i) * dty(pp)
NEXT i
CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)

DO
   DO
       LET drt$ = INKEY$
       LOOP WHILE drt$ = ""
       LET n1 = ASC(drt$)
       LET n2 = ASC(RIGHT$(drt$, 1))

     IF UCASE$(drt$) = "X" THEN GOTO exitloop    'key=X

     IF ASC(drt$) = 27 THEN GOTO exitloop        'key=esc

     IF UCASE$(drt$) = "C" THEN                  'key=C

          WINDOW SCREEN (0, 0)-(639, 479)
          LINE (171, 21)-(469, 249), colr(14), BF
          CALL drwobj
          IF dyn <> 4 THEN
            CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
            CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)
          END IF
          CALL arrow(x(1), y(1), a(1), colr(4))
          CALL arrow(x(2), y(2), a(2), colr(5))
          CALL arrow(x(3), y(3), a(3), colr(6))
          CALL Calculation
          IF ans = 1 THEN GOTO addf

     ELSEIF n1 = 0 AND n2 = 33 THEN              'key=<alt> F

          WINDOW SCREEN (0, 0)-(639, 479)
          LINE (171, 21)-(469, 249), colr(14), BF
          CALL SetAngle
          FOR i = 1 TO 3
            FOR k = 1 TO c
                IF dtx(k) = x(i) AND dty(k) = y(i) THEN
                   LET pp = k
```

```
            END IF
          NEXT k
          LET a(i) = ang(pp) + cone(i)
          LET lf#(i) = COS(a(i) * pi / 180)
          LET mf#(i) = SIN(a(i) * pi / 180)
          LET pf#(i) = mf#(i) * dtx(pp) - lf#(i) * dty(pp)
        NEXT i
        CALL drwobj
        CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
        CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)
        CALL arrow(x(1), y(1), a(1), colr(4))
        CALL arrow(x(2), y(2), a(2), colr(5))
        CALL arrow(x(3), y(3), a(3), colr(6))

ELSEIF n1 = 0 AND n2 = 73 THEN              'key=page up

      LET stp = stp + 1

ELSEIF n1 = 0 AND n2 = 81 THEN              'key=page down

      LET stp = stp - 1
      IF stp < 1 THEN LET stp = 1

ELSEIF n1 = 43 THEN                         'key="+"

      IF dyn <> 4 THEN
       CALL arrow(dtx(p), dty(p), ang(p) + cone(dyn), 0)
       CIRCLE (xe, ye), (rxx2 - rxx1) / 150, 0
       LET p = p + stp
       IF p > c THEN LET p = p - c
       LET x(dyn) = dtx(p)
       LET y(dyn) = dty(p)
       LET a(dyn) = ang(p) + cone(dyn)
       LET lf#(dyn) = COS(a(dyn) * pi / 180)
       LET mf#(dyn) = SIN(a(dyn) * pi / 180)
       LET var1 = mf#(dyn) * dtx(p) - lf#(dyn) * dty(p)
       LET pf#(dyn) = var1
       CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
       CALL arrow(x(1), y(1), a(1), colr(4))
       CALL arrow(x(2), y(2), a(2), colr(5))
       CALL arrow(x(3), y(3), a(3), colr(6))
      ELSE
       LET ind = ind - stp
       IF ind < 1 THEN LET ind = c
       IF rmode = 0 THEN
         WINDOW SCREEN (0, 0)-(639, 479)
         LINE (171, 21)-(469, 249), colr(14), BF
         CALL drwobj
         CALL arrow(x(1), y(1), a(1), colr(4))
         CALL arrow(x(2), y(2), a(2), colr(5))
```

```
            CALL arrow(x(3), y(3), a(3), colr(6))
            CALL Calculation
            IF ans = 1 THEN GOTO addf
          END IF
        END IF

ELSEIF n1 = 45 THEN                                    'key="-"

        IF dyn <> 4 THEN
         CALL arrow(dtx(p), dty(p), ang(p) + cone(dyn), 0)
         CIRCLE (xe, ye), (rxx2 - rxx1) / 150, 0
         LET p = p - stp
         IF p < 1 THEN LET p = p + c
         LET x(dyn) = dtx(p)
         LET y(dyn) = dty(p)
         LET a(dyn) = ang(p) + cone(dyn)
         LET lf#(dyn) = COS(a(dyn) * pi / 180)
         LET mf#(dyn) = SIN(a(dyn) * pi / 180)
         LET var1 = mf#(dyn) * dtx(p) - lf#(dyn) * dty(p)
         LET pf#(dyn) = var1
         CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
         CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)
         CALL arrow(x(1), y(1), a(1), colr(4))
         CALL arrow(x(2), y(2), a(2), colr(5))
         CALL arrow(x(3), y(3), a(3), colr(6))
        ELSE
         LET ind = ind + stp
         IF ind > c THEN LET ind = 1
         IF rmode = 0 THEN
           WINDOW SCREEN (0, 0)-(639, 479)
           LINE (171, 21)-(469, 249), colr(14), BF
           CALL drwobj
           CALL arrow(x(1), y(1), a(1), colr(4))
           CALL arrow(x(2), y(2), a(2), colr(5))
           CALL arrow(x(3), y(3), a(3), colr(6))
           CALL Calculation
           IF ans = 1 THEN GOTO addf
         END IF
        END IF

ELSEIF UCASE$(drt$) = "F" THEN                         'key=F

       IF mod$ = "2" AND dyn = 3 THEN
         CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
         CIRCLE (xe, ye), (rxx2 - rxx1) / 150, 0
         LET dyn = 4
       ELSE
         IF dyn <> 4 THEN
          CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
          CIRCLE (xe, ye), (rxx2 - rxx1) / 150, 0
```

```
          LET dyn = dyn + 1
          IF dyn > 3 THEN LET dyn = 1
          CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
          CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)
        ELSE
          LET dyn = 1
          CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
          CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)
        END IF

        IF dyn = 1 THEN
          FOR k = 1 TO c
           IF dtx(k) = x(1) AND dty(k) = y(1) THEN
              LET p = k
           END IF
          NEXT k
        END IF

        IF dyn = 2 THEN
          FOR k = 1 TO c
           IF dtx(k) = x(2) AND dty(k) = y(2) THEN
             LET p = k
           END IF
          NEXT k
        END IF

        IF dyn = 3 THEN
          FOR k = 1 TO c
           IF dtx(k) = x(3) AND dty(k) = y(3) THEN
              LET p = k
           END IF
          NEXT k
        END IF

      END IF
      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (132, 50)-STEP(10, 10), fcolor(dyn), BF
      WINDOW (rxx1, ryy1)-(rxx2, ryy2)

ELSEIF n1 = 0 AND n2 = 31 THEN              'key=<alt> S

      LET rmode = 0
      WINDOW SCREEN (0, 0)-(639, 479)
      COLOR colr(2)
      LOCATE 15, 5: PRINT "Static Run "
      COLOR colr(3)
      WINDOW (rxx1, ryy1)-(rxx2, ryy2)

ELSEIF n1 = 0 AND n2 = 32 THEN              'key=<alt> D
```

```
            LET rmode = 1
            WINDOW SCREEN (0, 0)-(639, 479)
            COLOR colr(2)
            LOCATE 15, 5: PRINT "Dynamic Run"
            WINDOW (rxx1, ryy1)-(rxx2, ryy2)

      ELSE

            GOTO addf 'Display menu

      END IF

      IF rmode = 1 THEN
            WINDOW SCREEN (0, 0)-(639, 479)
            LINE (171, 21)-(469, 249), colr(14), BF
            CALL drwobj
            IF dyn <> 4 THEN
              CALL arrow(x(dyn), y(dyn), a(dyn), fcolor(dyn))
              CIRCLE (xe, ye), (rxx2 - rxx1) / 150, colr(2)
            END IF
            CALL arrow(x(1), y(1), a(1), colr(4))
            CALL arrow(x(2), y(2), a(2), colr(5))
            CALL arrow(x(3), y(3), a(3), colr(6))
            CALL Calculation
            IF ans = 1 THEN GOTO addf
      END IF

   LOOP WHILE UCASE$(drt$) <> "X"

exitloop:
'set screen
      WINDOW SCREEN (0, 0)-(639, 479)
      CALL appearance
      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (171, 21)-(469, 249), colr(14), BF
      CALL drwobj
      CALL arrow(x(1), y(1), a(1), colr(4))
      CALL arrow(x(2), y(2), a(2), colr(5))
      CALL arrow(x(3), y(3), a(3), colr(6))
      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (21, 271)-(619, 439), 0, BF
      WINDOW (rxx1, ryy1)-(rxx2, ryy2)

END SUB

SUB appearance
'Print parameters on screen along with the program screen

SCREEN 12
CLS
```

```
LINE (0, 0)-(640, 460), 6, BF
LINE (0, 0)-(639, 460), 2, B
LINE (170, 20)-(470, 250), 2, B
LINE (20, 20)-(150, 180), 2, B
LINE (20, 200)-(150, 250), 2, B
LINE (20, 270)-(620, 440), 2, B
LINE (171, 21)-(469, 249), colr(14), BF
LINE (21, 21)-(149, 179), 0, BF
LINE (21, 201)-(149, 249), 0, BF
LINE (21, 271)-(619, 439), 0, BF

LINE (490, 20)-STEP(130, 34), 2, B
LINE (490, 69)-STEP(130, 34), 2, B
LINE (490, 118)-STEP(130, 34), 2, B
LINE (490, 167)-STEP(130, 34), 2, B
LINE (490, 216)-STEP(130, 34), 2, B

COLOR colr(2)
LOCATE 3, 5: PRINT USING "\            \"; file$
LOCATE 7, 15: PRINT USING "###"; cone(1)
LOCATE 8, 15: PRINT USING "###"; cone(2)
LOCATE 9, 15: PRINT USING "###"; cone(3)
LOCATE 10, 15: PRINT USING "###"; cone(4)
LOCATE 14, 13: PRINT USING "\  \"; mod$
IF mod$ = "1" THEN
     LOCATE 14, 14: PRINT " "
     LOCATE 14, 15: PRINT USING "###"; a(4)
     LOCATE 14, 18: PRINT CHR$(248)
END IF
LOCATE 15, 5: PRINT "Dynamic Run"

COLOR colr(1)
LOCATE 4, 5: PRINT "Active Force"
LOCATE 5, 4: PRINT "---------------"
LOCATE 6, 5: PRINT "Friction cone"
LOCATE 7, 9: PRINT "force"
LOCATE 8, 9: PRINT "force"
LOCATE 9, 9: PRINT "force"
LOCATE 10, 9: PRINT "EF"
LOCATE 11, 4: PRINT "---------------"
LOCATE 14, 5: PRINT "EF Mode"

LINE (132, 50)-STEP(10, 10), fcolor(dyn), BF
CALL StArrow(55, 375, 0, colr(4))
CALL StArrow(55, 359, 0, colr(5))
CALL StArrow(55, 343, 0, colr(6))
CALL StArrow(55, 326, 0, colr(7))

END SUB
```

```
FUNCTION arctan (x, y)
'This function is to find arctangent of the Cartesian
'coordinates
CONST pi = 3.141593

IF y <> 0 THEN
IF (x / y) = 0 THEN
     IF y > 0 THEN LET T = 90
     IF y <= 0 THEN LET T = -90
END IF
END IF
IF x > 0 THEN LET T = ATN(y / x) * 180 / pi
IF x < 0 THEN LET T = ATN(y / x) * 180 / pi + 180
LET T = adjust(T)
LET arctan = T

END FUNCTION

SUB arrow (x, y, a, c)
'This subroutine is to draw the arrow of force
CONST pi = 3.141593

LET xe = x - (rxx2 - rxx1) / 20 * COS(a * pi / 180)
LET ye = y - (rxx2 - rxx1) / 20 * SIN(a * pi / 180)
LET xa1 = x + (rxx2 - rxx1) / 80 * COS(a * pi / 180 - 3)
LET xa2 = x + (rxx2 - rxx1) / 80 * COS(a * pi / 180 + 3)
LET ya1 = y + (rxx2 - rxx1) / 80 * SIN(a * pi / 180 - 3)
LET ya2 = y + (rxx2 - rxx1) / 80 * SIN(a * pi / 180 + 3)
LINE (xa1, ya1)-(xa2, ya2), c
LINE (x, y)-(xa1, ya1), c
LINE (x, y)-(xa2, ya2), c
LINE (x, y)-(xe, ye), c

END SUB

SUB Calculation
'see Chapter 4
CONST pi = 3.141593

'Finding intersection pt

LET var1 = mf#(1) * lf#(2) - mf#(2) * lf#(1)
LET var2 = mf#(2) * lf#(3) - mf#(3) * lf#(2)
LET var3 = mf#(3) * lf#(1) - mf#(1) * lf#(3)
IF var1 = 0 THEN LET var1 = .000001
IF var2 = 0 THEN LET var2 = .000001
IF var3 = 0 THEN LET var3 = .000001

LET insx(3) = (pf#(1) * lf#(2) - pf#(2) * lf#(1)) / var1
```

```
LET insy(3) = (mf#(2) * insx(3) - pf#(2)) / lf#(2)
LET insx(1) = (pf#(2) * lf#(3) - pf#(3) * lf#(2)) / var2
LET insy(1) = (mf#(3) * insx(1) - pf#(3)) / lf#(3)
LET insx(2) = (pf#(3) * lf#(1) - pf#(1) * lf#(3)) / var3
LET insy(2) = (mf#(1) * insx(2) - pf#(1)) / lf#(1)

'No solution when three forces intersect at a point

LET V1 = insx(1)
LET V2 = insy(1)
LET V3 = insx(2)
LET V4 = insy(2)
LET V5 = insx(3)
LET V6 = insy(3)

IF CheckIntersect(V1, V2, V3, V4, V5, V6) = 1 THEN

     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 21, 11: PRINT "      When three restraint forces
intersect at a point. the"
LOCATE 22, 11: PRINT "object can rotate about the point of
intersection.   Please"
LOCATE 23, 11: PRINT "try another configuration of the
restraint forces."
LOCATE 25, 11: PRINT "Press any key to continue.."

     LET ans = 1
     DO
     LOOP WHILE INKEY$ = ""
     LINE (21, 271)-(619, 439), 0, BF
     WINDOW (rxx1, ryy1)-(rxx2, ryy2)
     GOTO Calculationend

END IF

LET ans = 0

'Find the solution

FOR i = 1 TO 3
    LET torque$(i) = moment$(a(i), pf#(i), insx(i), insy(i))
NEXT i

'In case of the direction is specified
IF mod$ = "1" THEN

     LET a(4) = adjust(a(4))
```

```
   IF (a(4)>135 AND a(4)<225) OR a(4)<45 OR a(4)>315 THEN

      LET below = -1E+11
      LET above = 1E+11
      LET lf#(4) = COS(a(4) * pi / 180)
      LET mf#(4) = SIN(a(4) * pi / 180)

      FOR i = 1 TO 3
           LET py(i) = insy(i)-mf#(4)*insx(i)/lf#(4)
      NEXT i

FOR i = 1 TO 3
  IF (a(4) > 90 AND a(4) < 270) AND torque$(i) = "cw" THEN
     LET below = max(below, py(i))
  END IF
  IF (a(4) > 90 AND a(4) < 270) AND torque$(i) = "ccw" THEN
     LET above = min(above, py(i))
  END IF
  IF (a(4) < 90 OR a(4) > 270) AND torque$(i) = "ccw" THEN
     LET below = max(below, py(i))
  END IF
  IF (a(4) < 90 OR a(4) > 270) AND torque$(i) = "cw" THEN
     LET above = min(above, py(i))
  END IF
NEXT i
     WINDOW (rxx1, ryy1)-(rxx2, ryy2)

     LET xxx = 0
     FOR k = 1 TO c
      LET V1$ = pst$(dtx(k), dty(k), a(4), above * lf#(4))
      IF V1$ = "below" THEN
       LET V2$ = pst$(dtx(k), dty(k), a(4), below * lf#(4))
       IF V2$ = "above" THEN
        IF clsang$(a(4), ang(k)) = "yes" THEN
         CALL arrow(dtx(k), dty(k), a(4), colr(7))
         LET xxx = 1
        END IF
       END IF
      END IF
     NEXT k
     IF xxx = 0 THEN LOCATE 3, 35: PRINT "NO SOLUTION"

  ELSE

     LET Left = -1E+11
     LET right = 1E+11
     LET lf#(4) = COS(a(4) * pi / 180)
     LET mf#(4) = SIN(a(4) * pi / 180)

     FOR i = 1 TO 3
```

```
        LET p(i) = mf#(4) * insx(i) - lf#(4) * insy(i)
        LET px(i) = p(i) / mf#(4)
      NEXT i

FOR i = 1 TO 3
      IF a(4) > 180 AND torque$(i) = "ccw" THEN
        LET Left = max(Left, px(i))
      END IF
      IF a(4) > 180 AND torque$(i) = "cw" THEN
        LET right = min(right, px(i))
      END IF
      IF a(4) < 180 AND torque$(i) = "cw" THEN
        LET Left = max(Left, px(i))
      END IF
      IF a(4) < 180 AND torque$(i) = "ccw" THEN
        LET right = min(right, px(i))
      END IF
NEXT i
WINDOW (rxx1, ryy1)-(rxx2, ryy2)

      LET xxx = 0
      FOR k = 1 TO c
       LET V1$ = pst$(dtx(k), dty(k), a(4), Left * mf#(4))
       IF V1$ = "right" THEN
        LET V2$ = pst$(dtx(k), dty(k), a(4), right * mf#(4))
        IF V2$ = "left" THEN
         IF clsang$(a(4), ang(k)) = "yes" THEN
          CALL arrow(dtx(k), dty(k), a(4), colr(7))
          LET xxx = 1
         END IF
        END IF
       END IF
      NEXT k
      COLOR colr(2)
      IF xxx = 0 THEN LOCATE 3, 35: PRINT "NO SOLUTION"

      END IF

'In case of calculate every point

      ELSEIF mod$ = "2" THEN

 FOR k = 1 TO c
  FOR i = 1 TO 3
   IF moment$(a(i), pf#(i), insx(i), insy(i)) = "ccw" THEN
    LET high(i) = arctan(dtx(k) - insx(i), dty(k) - insy(i))
    LET low(i) = arctan(insx(i) - dtx(k), insy(i) - dty(k))
   ELSE
    LET low(i) = arctan(dtx(k) - insx(i), dty(k) - insy(i))
    LET high(i) = arctan(insx(i) - dtx(k), insy(i) - dty(k))
```

```
    END IF
   NEXT i
   LET maxang(k) = high(1)
   LET minang(k) = low(1)

   FOR j = 2 TO 3
      IF in$(maxang(k), minang(k), high(j)) = "in" THEN
        LET maxang(k) = high(j)
      END IF
      IF in$(maxang(k), minang(k), low(j)) = "in" THEN
        LET minang(k) = low(j)
      END IF
    NEXT j
  NEXT k

   FOR k = 1 TO c
    LET maxang(k) = adjust(maxang(k) - ang(k))
    LET minang(k) = adjust(minang(k) - ang(k))
   NEXT k

   FOR k = 1 TO c
    IF maxang(k)>=180 THEN LET maxang(k)=maxang(k)-360
    IF minang(k)>=180 THEN LET minang(k)=minang(k)- 360
   NEXT k

   WINDOW SCREEN (0, 0)-(639, 479)
   LINE (21, 271)-(619, 439), 0, BF

'Set the output screen
COLOR colr(2)
LOCATE 27, 12: PRINT "The possible EF directions are above
yellow and below blue curves"
LOCATE 22, 5: PRINT "  0-direction"
LOCATE 20, 5: PRINT " 90-direction"
LOCATE 24, 5: PRINT "-90-direction"

      LINE (140, 314)-(600, 285), colr(12), BF
      LINE (140, 376)-(600, 405), colr(12), BF
      LINE (140, 315)-(600, 375), colr(11), BF

      LET var1 = 345 + cone(4) / 3
      LET var2 = 345 - cone(4) / 3
      LINE (140, var1)-(600, var2), colr(10), BF
      LINE (140, 345)-(600, 345), colr(13), B
      LET var1 = 140 + ind * 460 / c
      LINE (var1, 285)-(var1, 405), colr(13)

      FOR k = 1 TO c
         LET var1 = 345 - maxang(k) * 60 / 180
         PSET (140 + k * 460 / c, var1), colr(8)
```

```
        LET var2 = 345 - minang(k) * 60 / 180
        PSET (140 + k * 460 / c, var2), colr(7)
    NEXT k

    WINDOW (rxx1, ryy1)-(rxx2, ryy2)
    LET var1 = dtx(ind) - (rxx2 - rxx1) / 250
    LET var2 = dty(ind) - (rxx2 - rxx1) / 250
    LET var3 = dtx(ind) + (rxx2 - rxx1) / 250
    LET var4 = dty(ind) + (rxx2 - rxx1) / 250
    LINE (var1, var2)-(var3, var4), colr(12), BF
    LET var1 = adjust(maxang(ind) + ang(ind))
    LET var2 = adjust(minang(ind) + ang(ind))
    LET var3 = adjust(ang(ind) + cone(4))
    LET var4 = adjust(ang(ind) - cone(4))
    LET xxx = FitLimit(var1, var2, var3, var4)
    IF xxx = 1 THEN
          CALL arrow(dtx(ind), dty(ind), high(1), colr(8))
          CALL arrow(dtx(ind), dty(ind), low(1), colr(7))
    ELSE
              LOCATE 3, 35: PRINT "NO SOLUTION"
    END IF

    END IF

Calculationend:
END SUB

FUNCTION CheckIntersect (x1, y1, x2, y2, x3, y3)
'This function is to check that whether three lines
'intersect at a point or not.

LET Num = 15
LET CheckIntersect = 0
LET var1 = (x1 - x2) ^ 2 + (y1 - y2) ^ 2
LET var2 = ABS(((rxx1 - rxx2) / Num) ^ 2)
LET var3 = (x1 - x3) ^ 2 + (y1 - y3) ^ 2
LET var4 = ABS(((rxx1 - rxx2) / Num) ^ 2)
LET var5 = (x3 - x2) ^ 2 + (y3 - y2) ^ 2
LET var6 = ABS(((rxx1 - rxx2) / Num) ^ 2)

    IF var1 < var2 THEN
       IF var3 < var4 THEN
         IF var5 < var6 THEN
              LET CheckIntersect = 1
         END IF
       END IF
    END IF

    LET limit = var2 * .001
    IF var1 < limit OR var3 < limit OR var5 < limit THEN
```

```
            LET CheckIntersect = 1
        END IF

END FUNCTION

FUNCTION chkpt (ptx, lx1, lx2, pty, ly1, ly2)
'This  function is to find, when there are 2 points
'(lx1, ly1) and (lx2, ly2), whether point (ptx, pty) is
'between those two point or not.

LET chkpt = 1
IF lx1 <> lx2 THEN
    IF ptx <= lx1 AND ptx > lx2 THEN LET chkpt = 0
    IF ptx >= lx1 AND ptx < lx2 THEN LET chkpt = 0
ELSE
    IF pty <= ly1 AND pty > ly2 THEN LET chkpt = 0
    IF pty >= ly1 AND pty < ly2 THEN LET chkpt = 0
END IF

END FUNCTION

FUNCTION clsang$ (a, b)
'This function is to find the difference between the angle
'"a" and the angle "b". The function returns "YES"
'when the difference is less than 90 degree.

LET clsang$ = ""
LET an = min(ABS(a - b), ABS(a - b + 360))
LET an = min(an, ABS(a - b - 360))

IF an < cone(4) THEN
     LET clsang$ = "yes"
ELSE
     LET clsang$ = "no"
END IF

END FUNCTION

SUB drwobj
'This subroutine is to draw the object on screen

LET rx1 = dtx(1)
LET ry1 = dty(1)
LET rx2 = dtx(1)
LET ry2 = dty(1)

FOR k = 2 TO c - 1
     LET rx1 = min(rx1, dtx(k))
     LET rx2 = max(rx2, dtx(k))
     LET ry1 = min(ry1, dty(k))
```

```
      LET ry2 = max(ry2, dty(k))
NEXT k

IF (ry2 - ry1) / 150 > (rx2 - rx1) / 220 THEN
      LET ryy2 = ry2 + (ry2 - ry1) / 150 * 60
      LET ryy1 = ry1 - (ry2 - ry1) / 150 * 270
      LET rxx1 = .5*(rx1 + rx2)-(ry2 - ry1)/150*320
      LET rxx2 = rxx1 + (ry2 - ry1) / 150 * 640
ELSE
      LET rxx1 = rx1 - (rx2 - rx1) / 220 * 210
      LET rxx2 = rx2 + (rx2 - rx1) / 220 * 210
      LET ryy1 = ry1 - (rx2 - rx1) / 220 * 270
      LET ryy2 = ryy1 + (rx2 - rx1) / 220 * 480
END IF

      IF rxx1 = rxx2 THEN LET rxx2 = rxx2 + 10
      IF ryy1 = ryy2 THEN LET ryy2 = ryy2 + 10

      WINDOW (rxx1, ryy1)-(rxx2, ryy2)

FOR k = 1 TO c - 1
  LINE (dtx(k), dty(k))-(dtx(k + 1), dty(k + 1)), colr(15)
NEXT k

END SUB

FUNCTION fcolor (force)
'This routine sets color of force

      IF force = 1 THEN LET fcolor = colr(4)
      IF force = 2 THEN LET fcolor = colr(5)
      IF force = 3 THEN LET fcolor = colr(6)
      IF force = 4 THEN LET fcolor = colr(9)

END FUNCTION

FUNCTION FitLimit (maxa, mina, ul, ll)
'This routine limits direction of the applied force to
'within UL and LL.

      IF adjust(ul - maxa) < 180 THEN
            LET high(1) = maxa
      ELSE
            LET high(1) = ul
      END IF
      IF adjust(mina - ll) < 180 THEN
            LET low(1) = mina
      ELSE
            LET low(1) = ll
      END IF
```

```
      IF adjust(high(1) - low(1)) < 180 THEN
           LET FitLimit = 1
      ELSE
           LET FitLimit = 0
      END IF

END FUNCTION

SUB help
'Help routine

      SHELL "help.exe"
      CALL appearance
      CALL drwobj

END SUB

FUNCTION in$ (high, low, angle)
'This function is to find whether the angle is between the
'"high" and "low" angle or not.

LET in$ = "out"
IF high >= 180 AND angle > low AND angle < high THEN
     LET in$ = "in"
ELSEIF low < high THEN
     IF angle < high AND angle > low THEN LET in$ = "in"
ELSEIF low > high THEN
     IF angle < high OR angle > low THEN LET in$ = "in"
END IF

END FUNCTION

FUNCTION max (a, b)
     IF a >= b THEN LET max = a
     IF a < b THEN LET max = b
END FUNCTION

FUNCTION min (a, b)
     IF a >= b THEN LET min = b
     IF a < b THEN LET min = a
END FUNCTION

FUNCTION moment$ (a, p#, x, y)
'This function finds moment about (x, y) of force passing
'through p# on the X axis in the direction of a.
CONST pi = 3.141593

LET a = adjust(a)
LET l# = COS(a * pi / 180)
LET m# = SIN(a * pi / 180)
```

```
IF (a > 45 AND a < 135) OR (a > 225 AND a < 315) THEN
     LET mom = (x - p# / m#) * m# - y * l#
ELSE
     LET mom = x * m# - (y + p# / l#) * l#
END IF

IF mom > 0 THEN LET moment$ = "cw"
IF mom < 0 THEN LET moment$ = "ccw"
IF mom = 0 THEN LET moment$ = "0"

END FUNCTION

SUB parameters
'This routine sets parameters at the beginning

     LET flip = 0
     LET dyn = 1
     LET mod$ = "1"
     LET a(4) = 0
     LET rxx1 = 0
     LET rxx2 = 10
     LET ryy1 = 0
     LET ryy2 = 10
     LET cone(4) = 90
     LET colr(1) = 7   'text
     LET colr(2) = 14   'highlight text
     LET colr(3) = 11   'input text
     LET colr(4) = 10   'force 1
     LET colr(5) = 11   'force 2
     LET colr(6) = 12   'force 3
     LET colr(7) = 14   'force 4
     LET colr(8) = 1    'force 4
     LET colr(9) = 6    'force 4
     LET colr(10) = 7   'graph possible zone
     LET colr(11) = 8   'graph out zone
     LET colr(12) = 6   'graph in profife
     LET colr(13) = 11  'graph
     LET colr(14) = 0   'background
     LET colr(15) = 7   'profile

END SUB

FUNCTION pick (a1, a2, la)
'This routine determines whether a1 or a2 that is close to
'la

LET a = min(ABS(a1 - la), ABS(a1 - la + 360))
LET a = min(a, ABS(a1 - la - 360))
LET b = min(ABS(a2 - la), ABS(a2 - la + 360))
LET b = min(b, ABS(a2 - la - 360))
```

```
IF b <= a THEN LET pick = a2
IF a < b THEN LET pick = a1

END FUNCTION

SUB prnfile (j, k, strg$)
'Print data files on screen

IF j < 8 THEN
     LOCATE 20 + j, 14: PRINT USING "\            \"; strg$
ELSEIF j < 15 THEN
     LOCATE 13 + j, 34: PRINT USING "\            \"; strg$
ELSEIF j < 22 THEN
     LOCATE 6 + j, 54: PRINT USING "\            \"; strg$
END IF

END SUB

FUNCTION pst$ (x, y, a, p#)
'Check position of line, passing through p# on the X axis in
'the direction of a, with respect to  point (x, y).
CONST pi = 3.141593

     LET a = adjust(a)
     LET l# = COS(a * pi / 180)
     LET m# = SIN(a * pi / 180)

     IF (a < 45 OR a > 315) OR (a > 135 AND a < 225) THEN
        IF (p# + m# * x) / l# < y THEN LET pst$ = "above"
        IF (p# + m# * x) / l# > y THEN LET pst$ = "below"
     ELSE
        IF (p# + l# * y) / m# > x THEN LET pst$ = "left"
        IF (p# + l# * y) / m# < x THEN LET pst$ = "right"
     END IF

END FUNCTION

FUNCTION ReadNum (Num$)
'This function is to convert string to number

IF Num$ = "" THEN
     LET chkrd$ = "e"
     LET ReadNum = 0
     GOTO empty
END IF
LET chkrd$ = ""
LET ret = 0
LET sign = 1
IF ASC(LEFT$(Num$, 1)) = 45 THEN
     LET sign = -1
```

```
     LET Num$ = RIGHT$(Num$, LEN(Num$) - 1)
END IF

LET length = LEN(Num$)
FOR i = 1 TO length
 IF ASC(LEFT$(Num$, 1))>47 AND ASC(LEFT$(Num$, 1))<58 THEN
  LET ret = 10 * ret + ASC(LEFT$(Num$, 1)) - 48
 ELSEIF ASC(LEFT$(Num$, 1)) = 46 THEN
  LET Num$ = RIGHT$(Num$, LEN(Num$) - 1)
  FOR j = 1 TO LEN(Num$)
   IF ASC(LEFT$(Num$, 1))>47 AND ASC(LEFT$(Num$, 1))<58 THEN
    LET ret = ret + (.1 ^ j) * (ASC(LEFT$(Num$, 1)) - 48)
   ELSE
    LET chkrd$ = "x"
   END IF
   LET Num$ = RIGHT$(Num$, LEN(Num$) - 1)
  NEXT j
     LET i = length
     GOTO readend
     ELSE
     LET chkrd$ = "x"
     GOTO readend
     END IF
     LET Num$ = RIGHT$(Num$, LEN(Num$) - 1)
NEXT i

readend: LET ReadNum = sign * ret
empty:

END FUNCTION

SUB SetAngle
'This routine finds angle perpendicular the the surface
'for every point.
CONST pi = 3.141593

reverse:

     IF flip = 0 THEN
          LET flip = 180
     ELSE
          LET flip = 0
     END IF

  LET a1 = arctan(dtx(1) - dtx(c - 1), dty(1) - dty(c - 1))
  LET a2 = arctan(dtx(1) - dtx(2), dty(1) - dty(2))
  LET ang(1) = adjust((a1 + a2) / 2 + flip)

  FOR k = 2 TO c - 1
   LET a1 = arctan(dtx(k) - dtx(k - 1), dty(k) - dty(k - 1))
```

```
 LET a2 = arctan(dtx(k) - dtx(k + 1), dty(k) - dty(k + 1))
 LET a3 = adjust((a1 + a2) / 2)
 LET a4 = adjust((a1 + a2) / 2 + 180)
 LET ang(k) = pick(a3, a4, ang(k - 1))
NEXT k

LET a1 = arctan(dtx(c) - dtx(c - 1), dty(c) - dty(c - 1))
LET a2 = arctan(dtx(c) - dtx(2), dty(c) - dty(2))
LET a3 = adjust((a1 + a2) / 2)
LET a4 = adjust((a1 + a2) / 2 + 180)
LET ang(c) = pick(a3, a4, ang(c - 1))

   LET MaxX = dtx(1)
   LET Bound = 1
   FOR i = 2 TO c
     IF max(MaxX, dtx(i)) > MaxX THEN
        LET MaxX = dtx(i)
        LET Bound = i
     END IF
   NEXT i

 LET var1 = adjust(ang(Bound))
 LET var2 = adjust(ang(Bound))
 IF (var1 < 90 OR var2 > 270) AND x = 0 THEN GOTO reverse

   LET x = 1
   IF RIGHT$(LCASE$(file$), 3) = "tst" THEN

          LOCATE 29, 1:
          SHELL "del obj.tst"
          LET dfile$ = "obj.tst"
         OPEN dfile$ FOR OUTPUT AS #2
          LET i = 1
          DO WHILE i < c
             WRITE #2, dtx(i), dty(i)
             LET i = i + 1
          LOOP
         CLOSE #2

         OPEN file$ FOR OUTPUT AS #2
          LET i = 1
          DO WHILE i < c
             WRITE #2, dtx(i), dty(i)
             LET i = i + 1
          LOOP
         CLOSE #2

   ELSEIF RIGHT$(LCASE$(file$), 3) = "srf" THEN

          LOCATE 29, 1:
```

```
            SHELL "del obj.tst"
            LET dfile$ = "obj.tst"
           OPEN dfile$ FOR OUTPUT AS #2
            LET i = 1
            DO WHILE i < c
               WRITE #2, dtx(i), dty(i)
               LET i = i + 1
            LOOP
           CLOSE #2

           LET file$ = LEFT$(file$, LEN(file$) - 3) + "tst"
          OPEN file$ FOR OUTPUT AS #2
           LET i = 1
           DO WHILE i < c
               WRITE #2, dtx(i), dty(i)
               LET i = i + 1
           LOOP
          CLOSE #2

     END IF

END SUB

SUB SetEF
'see Chapter 4

     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF
     LET xxx$ = mod$

COLOR colr(1)
LOCATE 19, 8: PRINT "External Force (EF)"
LOCATE 21, 8: PRINT "     External force can be specified in
either of 2 ways. Direction"
LOCATE 22, 8: PRINT "of EF can  be  specified and program
displays possible positions or"
LOCATE 23, 8: PRINT "position can be specified and program
displays possible directions."

COLOR colr(2)
LOCATE 19, 8: PRINT "External Force (EF)"
LOCATE 25, 8: PRINT "Enter EF mode number"
LOCATE 25, 30: PRINT "Mode 1: Specify EF direction"
LOCATE 26, 30: PRINT "Mode 2: Specify EF position"

COLOR colr(3)
LOCATE 25, 35: PRINT "1"
LOCATE 26, 35: PRINT "2"

     DO
```

```
        LET mod$ = UCASE$(INKEY$)
        IF mod$ = "1" THEN EXIT DO
        IF mod$ = "2" THEN EXIT DO
        IF mod$ <> "" THEN
           IF ASC(mod$) = 27 THEN EXIT DO
        END IF
      LOOP

      IF ASC(mod$) = 27 THEN
           LET mod$ = xxx$
           GOTO extend
      END IF
      LET mode$ = mod$

COLOR colr(2)
LOCATE 14, 13: PRINT USING "\   \"; mode$
LOCATE 14, 14: PRINT "      "

      IF mod$ = "1" THEN
di: WINDOW SCREEN (0, 0)-(639, 479)
      LINE (21, 370)-(619, 439), 0, BF

COLOR colr(2)
LOCATE 26, 8: PRINT "Please enter the direction (in degree)"
LOCATE 26, 47: INPUT "of external force: ", aaa$

      LET a(4) = ReadNum(aaa$)
      IF chkrd$ = "x" THEN GOTO di
      LET a(4) = adjust(a(4))

COLOR colr(2)
LOCATE 14, 14: PRINT " "
LOCATE 14, 15: PRINT USING "###"; a(4)
LOCATE 14, 18: PRINT CHR$(248)

      END IF

      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (21, 271)-(619, 439), 0, BF

extend:
END SUB

SUB SetFriction
'see Chapter 4

SetFrictionbegin:
      WINDOW SCREEN (0, 0)-(639, 479)
      LINE (21, 271)-(619, 439), 0, BF
      LINE (171, 21)-(469, 249), colr(14), BF
```

```
COLOR colr(1)
LOCATE 22, 10: PRINT "1) Maximum angle between external
force and surface normal"
LOCATE 23, 10: PRINT "2) The angle between      force and
surface normal"
LOCATE 24, 10: PRINT "3) The angle between      force and
surface normal"
LOCATE 25, 10: PRINT "4) The angle between      force and
surface normal"
LOCATE 26, 10: PRINT "5) Return to main menu"

COLOR colr(2)
LOCATE 20, 7: PRINT "Set Parameters"

COLOR colr(3)
LOCATE 22, 10: PRINT "1)"
LOCATE 23, 10: PRINT "2)"
LOCATE 24, 10: PRINT "3)"
LOCATE 25, 10: PRINT "4)"
LOCATE 26, 10: PRINT "5) R"

     CALL StArrow(265, 120, 0, colr(4))
     CALL StArrow(265, 105, 0, colr(5))
     CALL StArrow(265, 89, 0, colr(6))

     COLOR colr(3)

     DO
          LET sts$ = UCASE$(INKEY$)
          IF sts$ = "1" THEN EXIT DO
          IF sts$ = "2" THEN EXIT DO
          IF sts$ = "3" THEN EXIT DO
          IF sts$ = "4" THEN EXIT DO
          IF sts$ = "5" THEN EXIT DO
          IF sts$ = "R" THEN EXIT DO
          IF sts$ <> "" THEN
          IF ASC(sts$) = 27 THEN EXIT DO
          END IF
     LOOP

     IF sts$ = "1" THEN
set1: WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 21, 10: PRINT "    The maximum angle of the external
force limits possible"
LOCATE 22, 10: PRINT "external  force  positions  (or
directions)  to those  with"
```

```
LOCATE 23, 10: PRINT "absolute value of angle to surface
normal less than limit."

COLOR colr(2)
LOCATE 19, 10: PRINT "1) Maximum angle between external
force and surface normal"
LOCATE 25, 10: INPUT "Input the friction cone half angle in
Deg.  ", con$
LET cone(4) = ReadNum(con$)
LET cone(4) = adjust(cone(4))
IF cone(4) > 90 THEN GOTO set1
IF chkrd$ = "x" THEN GOTO set1
LOCATE 10, 15: PRINT USING "###"; cone(4)
     GOTO SetFrictionbegin
     END IF

     IF sts$ = "2" THEN
set2: WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 21, 10: PRINT "     The angle between this force and
the body surface normal"
LOCATE 22, 10: PRINT "is arctangent of friction force
divided by normal force."

COLOR colr(2)
LOCATE 19, 10: PRINT "2) The angle between      force and
surface normal"
LOCATE 19, 50: PRINT " body surface"
CALL StArrow(265, 185, 0, colr(4))
LOCATE 24, 10: INPUT "Input the angle from the surface
normal in Deg.    ", con$
LET cone(1) = ReadNum(con$)
LET cone(1) = adjust(cone(1))
IF cone(1) > 45 AND cone(1) < 315 THEN GOTO set2
IF cone(1) >= 315 THEN LET cone(1) = cone(1) - 360
IF chkrd$ = "x" THEN GOTO set2
LOCATE 7, 15: PRINT USING "###"; cone(1)

     GOTO SetFrictionbegin
     END IF

     IF sts$ = "3" THEN
set3: WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 21, 10: PRINT "     The angle between this force and
the body surface normal"
```

```
LOCATE 22, 10: PRINT "is arctangent of friction force
divided by normal force."

COLOR colr(2)
LOCATE 19, 10: PRINT "3) The angle between      force and
surface normal"
LOCATE 19, 50: PRINT " body surface"
CALL StArrow(265, 185, 0, colr(5))
LOCATE 24, 10: INPUT "Input the angle from the surface
normal in Deg.     ", con$
LET cone(2) = ReadNum(con$)
LET cone(2) = adjust(cone(2))
IF cone(2) > 45 AND cone(2) < 315 THEN GOTO set3
IF cone(2) >= 315 THEN LET cone(2) = cone(2) - 360
IF chkrd$ = "x" THEN GOTO set3
LOCATE 8, 15: PRINT USING "###"; cone(2)

     GOTO SetFrictionbegin
     END IF

     IF sts$ = "4" THEN
set4: WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 21, 10: PRINT "     The angle between this force and
the body surface normal"
LOCATE 22, 10: PRINT "is arctangent of friction force
divided by normal force."

COLOR colr(2)
LOCATE 19, 10: PRINT "4) The angle between      force and
surface normal"
LOCATE 19, 50: PRINT " body surface"
CALL StArrow(265, 185, 0, colr(6))
LOCATE 24, 10: INPUT "Input the angle from the surface
normal in Deg.     ", con$
LET cone(3) = ReadNum(con$)
LET cone(3) = adjust(cone(3))
IF cone(3) > 45 AND cone(3) < 315 THEN GOTO set4
IF cone(3) >= 315 THEN LET cone(3) = cone(3) - 360
IF chkrd$ = "x" THEN GOTO set4
LOCATE 9, 15: PRINT USING "###"; cone(3)

     GOTO SetFrictionbegin
     END IF

     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 271)-(619, 439), 0, BF
     LINE (171, 21)-(469, 249), colr(14), BF
```

```
      WINDOW (rxx1, ryy1)-(rxx2, ryy2)

      FOR i = 1 TO c
       IF dtx(i) = x(1) AND dty(i) = y(1) THEN
        LET a(1) = adjust(ang(i) + cone(1))
       END IF
       IF dtx(i) = x(2) AND dty(i) = y(2) THEN
        LET a(2) = adjust(ang(i) + cone(2))
       END IF
       IF dtx(i) = x(3) AND dty(i) = y(3) THEN
        LET a(3) = adjust(ang(i) + cone(3))
       END IF
      NEXT i

END SUB

SUB SetObject
'see Chapter 4

DIM ln#(1 TO 1000)
DIM strg$(1 TO 24)

objbegin:
LET Num = 200
LET x = 0

WINDOW SCREEN (0, 0)-(639, 479)
LINE (21, 271)-(619, 439), 0, BF
LINE (171, 21)-(469, 249), colr(14), BF

COLOR colr(1)
LOCATE 21, 8: PRINT "     Object can be defined by entering
corner points (max 200) of"
LOCATE 22, 8: PRINT "the object.  To finish,  enter empty
line (press only ENTER key)."
LOCATE 23, 8: PRINT "Object can also be loaded from a file
that contains these points."

COLOR colr(2)
LOCATE 19, 8: PRINT "Define object"
LOCATE 25, 8: PRINT "Choose one.."
LOCATE 26, 12: PRINT "1) Enter from keyboard"
LOCATE 27, 12: PRINT "2) Read from file"

COLOR colr(3)
LOCATE 26, 12: PRINT "1) E"
LOCATE 27, 12: PRINT "2) R"

LET choice$ = ""
```

```
DO
     LET choice$ = UCASE$(INKEY$)
     IF choice$ = "E" THEN EXIT DO
     IF choice$ = "R" THEN EXIT DO
     IF choice$ <> "" THEN
          IF ASC(choice$) = 27 THEN EXIT DO
     END IF
LOOP WHILE choice$ <> "1" AND choice$ <> "2"

     IF choice$ = "E" THEN LET choice$ = "1"
     IF choice$ = "R" THEN LET choice$ = "2"

IF ASC(choice$) = 27 THEN
     LET file$ = "obj.tst"
     GOTO file
END IF

IF choice$ = "2" THEN 'Read data file

'Print the available files onto the screen
     SHELL "dir *.*>file.dat"
     LET k = 0
     LET pnt = 1
reread: LET file$ = "file.dat"
     OPEN file$ FOR INPUT AS #1
     LET i = 0
     DO WHILE NOT EOF(1)
       IF i > 21 + k THEN GOTO exitread
       INPUT #1, strg$(24)
       IF RIGHT$(LEFT$(strg$(24), 12), 3) = "TST" THEN
          LET strg$(24) = LEFT$(strg$(24), 8)
          LET i = i + 1
          IF i - k > 0 THEN
     FOR h = 1 TO 7
       IF RIGHT$(strg$(24), 1) = " " THEN
        LET strg$(24) = LEFT$(strg$(24), LEN(strg$(24)) - 1)
       END IF
     NEXT h
          LET strg$(i - k) = strg$(24) + ".TST"
          END IF
       ELSEIF RIGHT$(LEFT$(strg$(24), 12), 3) = "SRF" THEN
          LET strg$(24) = LEFT$(strg$(24), 8)
          LET i = i + 1
          IF i - k > 0 THEN
     FOR h = 1 TO 7
       IF RIGHT$(strg$(24), 1) = " " THEN
        LET strg$(24) = LEFT$(strg$(24), LEN(strg$(24)) - 1)
       END IF
     NEXT h
          LET strg$(i - k) = strg$(24) + ".SRF"
```

```
          END IF
        END IF
      LOOP
exitread:

      CLOSE #1
      IF i - k < 21 THEN
        IF k <> 0 THEN
      LET k = k - 1
      GOTO reread
        END IF
      END IF
      LINE (21, 271)-(619, 439), 0, BF

COLOR colr(2)
LOCATE 19, 10: PRINT "Please enter the object file      "
LOCATE 20, 10: PRINT "[Use arrow up/down keys to highlight
file and hit enter to open]"
COLOR colr(1)

      FOR j = 1 TO i - k
       CALL prnfile(j, k, strg$(j))
      NEXT j

COLOR colr(3)
      CALL prnfile(pnt - k, k, strg$(pnt - k))

'Ask for input file
fileinput:
LET file$ = ""
DO
  LET getkey$ = ""
  DO
     LET getkey$ = INKEY$
  LOOP WHILE getkey$ = ""
  IF ASC(getkey$) = 13 THEN EXIT DO
  IF ASC(getkey$) = 27 THEN GOTO objbegin
  IF ASC(getkey$) = 0 THEN

IF ASC(RIGHT$(getkey$, 1)) = 80 THEN
      COLOR colr(1)
      CALL prnfile(pnt - k, k, strg$(pnt - k))
      LET pnt = pnt + 1
      IF pnt > i - 1 THEN
         LET pnt = i
         LET k = k + 1
         GOTO reread
      END IF
      COLOR colr(3)
      CALL prnfile(pnt - k, k, strg$(pnt - k))
```

```
      GOTO fileinput
END IF
IF ASC(RIGHT$(getkey$, 1)) = 72 THEN
      COLOR colr(1)
      CALL prnfile(pnt - k, k, strg$(pnt - k))
      LET pnt = pnt - 1
      IF pnt = 0 THEN LET pnt = 1
        IF pnt < k + 1 THEN
           LET k = k - 1
           GOTO reread
        END IF
        COLOR colr(3)
        CALL prnfile(pnt - k, k, strg$(pnt - k))
        GOTO fileinput
      END IF
END IF
IF ASC(getkey$) = 8 THEN
        IF LEN(file$) <> 0 THEN
        LET file$ = LEFT$(file$, LEN(file$) - 1)
        LOCATE 19, 42: PRINT file$ + " "
        END IF
ELSE
        LET file$ = file$ + getkey$
END IF
LOCATE 19, 42: PRINT file$
LOOP

      IF file$ = "" THEN
             LET file$ = strg$(pnt - k)
      END IF

      IF RIGHT$(LCASE$(file$), 4) = ".tst" THEN
       OPEN file$ FOR APPEND AS #1
        LET maxf = LOF(1)
       CLOSE #1
       IF maxf = 0 THEN
           KILL file$
           GOTO reread
       END IF
      ELSE
       OPEN file$ FOR APPEND AS #1
        LET maxf = LOF(1)
       CLOSE #1
       IF maxf = 0 THEN
           KILL file$
           GOTO objbegin
       END IF
      END IF

      COLOR colr(2)
```

```
      LOCATE 3, 5: PRINT USING "\            \"; file$
      LOCATE 25, 10:

 file:
   IF RIGHT$(LCASE$(file$), 3) = "srf" THEN
      OPEN file$ FOR RANDOM ACCESS READ AS #1 LEN = 4
        LET c = INT(LOF(1) / 4)
        LET i = 0
        DO
           LET i = i + 2
           GET #1, i - 1, dtx(i / 2)
           GET #1, i, dty(i / 2)
        LOOP WHILE i < c
        LET c = INT(i / 2)
      CLOSE #1
   ELSEIF RIGHT$(LCASE$(file$), 3) = "tst" THEN
      OPEN file$ FOR INPUT AS #1
        LET i = 0
        DO WHILE NOT EOF(1)
           LET i = i + 1
           INPUT #1, dtx(i), dty(i)
        LOOP
        LET c = i + 1
      CLOSE #1
   ELSE
      LINE (21, 271)-(619, 439), 0, BF

COLOR colr(2)
LOCATE 19, 8: PRINT "Picture file"

COLOR colr(1)
LOCATE 21, 8: PRINT "       Picture file is a gray-scale
image.  The program will find a"
LOCATE 22, 8: PRINT "contour line of the  object and then
transform  it to  a series of"
LOCATE 23, 8: PRINT "points. The series is also saved in the
file named 'obj.srf'."

COLOR colr(2)
lex:
LOCATE 25, 10: PRINT "Enter number of picture columns:
"
LOCATE 25, 43: INPUT " ", xxx$
LET lenx = ReadNum(xxx$)
IF chkrd$ = "x" OR chkrd$ = "e" THEN GOTO lex
ley:
LOCATE 26, 10: PRINT "Enter number of picture rows:
"
LOCATE 26, 43: INPUT " ", yyy$
LET leny = ReadNum(yyy$)
```

```
IF chkrd$ = "x" OR chkrd$ = "e" THEN GOTO ley

     SHELL "del file.dat"
     OPEN "file.dat" FOR OUTPUT AS #1
          WRITE #1, file$
          WRITE #1, lenx
          WRITE #1, leny
     CLOSE #1

'check file
     IF maxf <> lenx * leny THEN GOTO objbegin
     SHELL "grp"
     LET file$ = "obj.srf"
     CALL appearance
     GOTO file
  END IF

ELSE

'Draw an object from keyboard
     LINE (21, 370)-(619, 439), 0, BF
     LET i = 1
     LET c = 2

COLOR colr(2)
r1x: LOCATE 25, 10:
INPUT "Enter the corner point (X-Axis) of object:    ", xxx$
LET dtx(i) = ReadNum(xxx$)
IF chkrd$ = "x" THEN
     LINE (430, 370)-(619, 439), 0, BF
     GOTO r1x
END IF
r1y: LOCATE 26, 10:
INPUT "Enter the corner point (Y-Axis) of object:    ", yyy$
LET dty(i) = ReadNum(yyy$)
IF chkrd$ = "x" THEN
     LINE (430, 400)-(619, 439), 0, BF
     GOTO r1y
END IF

     DO
     LET i = i + 1
     LET c = c + 1
     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 370)-(619, 439), 0, BF

COLOR colr(1)
LOCATE 27, 10: PRINT "Press <ENTER> to finish.."
COLOR colr(2)
```

```
rx: LOCATE 25, 10:
INPUT "Enter the corner point (X-Axis) of object:    ", xxx$
LET dtx(i) = ReadNum(xxx$)
IF chkrd$ = "x" THEN
     LINE (430, 370)-(619, 439), 0, BF
     GOTO rx
END IF
IF chkrd$ = "e" THEN EXIT DO
ry: LOCATE 26, 10:
INPUT "Enter the corner point (Y-Axis) of object:    ", yyy$
LET dty(i) = ReadNum(yyy$)
IF chkrd$ = "x" THEN
     LINE (430, 400)-(619, 439), 0, BF
     GOTO ry
END IF
IF chkrd$ = "e" THEN EXIT DO

     FOR l = 1 TO i - 1
       IF dtx(i) = dtx(l) AND dty(i) = dty(l) THEN
          EXIT DO
       END IF
     NEXT l

     LET dtx(c) = dtx(1)
     LET dty(c) = dty(1)
     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (171, 21)-(469, 249), colr(14), BF
     CALL drwobj
     IF c <> 3 THEN
          LINE (dtx(c - 1), dty(c - 1))-(dtx(1), dty(1)), 0
     END IF
     LOOP
     LET c = c - 1
     LET dtx(c) = dtx(1)
     LET dty(c) = dty(1)
     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (171, 21)-(469, 249), colr(14), BF
     CALL drwobj

     WINDOW SCREEN (0, 0)-(639, 479)
     LINE (21, 370)-(619, 439), 0, BF

LOCATE 26, 10: INPUT "Please enter file name  ", file$


     IF RIGHT$(file$, 3) <> "tst" THEN
          LET file$ = file$ + ".tst"
     END IF

LOCATE 3, 5: PRINT USING "\            \"; file$
```

```
        END IF

        LET dtx(c) = dtx(1)
        LET dty(c) = dty(1)

        WINDOW SCREEN (0, 0)-(639, 479)
        LINE (21, 271)-(619, 439), 0, BF
        LINE (171, 21)-(469, 249), colr(14), BF

'Make number of points to over 0.8*Num
        IF c < .8 * Num THEN
        FOR i = 1 TO c - 1
             LET var1 = (dtx(i) - dtx(i + 1)) ^ 2
             LET var2 = (dty(i) - dty(i + 1)) ^ 2
             LET ln#(i) = (var1 + var2) ^ .5
        NEXT i

'Find the total length of the surface
        LET length = 0
        FOR i = 1 TO c - 1
             LET length = length + ln#(i)
        NEXT i
        LET stp = length / (.8 * Num)
        LET i = 1
        LET j = 0

'Generate points
        DO
          LET i = i - 1
          LET j = j + 1
          LET k = 0
          DO
             LET i = i + 1
             LET k = k + 1
             LET var1 = (k - 1) * stp * (dtx(j + 1) - dtx(j))
             LET var1 = dtx(j) + var1 / ln#(j)
             LET dtx(i + INT(1.25 * Num)) = var1
             LET var2 = (k - 1) * stp * (dty(j + 1) - dty(j))
             LET var2 = dty(j) + var2 / ln#(j)
             LET dty(i + INT(1.25 * Num)) = var2
             LET V1 = dtx(i + INT(1.25 * Num))
             LET V2 = dtx(j)
             LET V3 = dtx(j + 1)
             LET V4 = dty(i + INT(1.25 * Num))
             LET V5 = dty(j)
             LET V6 = dty(j + 1)
             IF chkpt(V1, V2, V3, V4, V5, V6) <> 0 THEN
             EXIT DO
             END IF
```

```
        LOOP
      LOOP WHILE j < c - 1

      LET c = i
      FOR i = 1 TO c - 1
      LET dtx(i) = dtx(INT(1.25 * Num) + i)
      LET dty(i) = dty(INT(1.25 * Num) + i)
      NEXT i
      LET dtx(c) = dtx(1)
      LET dty(c) = dty(1)

END IF

'Reduce number of points to Num
      IF c > Num THEN
        FOR i = 1 TO Num
           LET var1 = dtx(INT((i - 1) * c / Num) + 1)
           LET dtx(INT(1.25 * Num) + i) = var1
           LET var2 = dty(INT((i - 1) * c / Num) + 1)
           LET dty(INT(1.25 * Num) + i) = var2
        NEXT i
        FOR i = 1 TO Num
           LET dtx(i) = dtx(INT(1.25 * Num) + i)
           LET dty(i) = dty(INT(1.25 * Num) + i)
        NEXT i
        LET dtx(Num) = dtx(1)
        LET dty(Num) = dty(1)
        LET c = Num
      END IF

      CALL drwobj
      CALL SetAngle
      LET p = 1
      LET dyn = 1

      LET x(1) = dtx(1)
      LET x(2) = dtx(INT(c / 3))
      LET x(3) = dtx(INT(2 * c / 3))

      LET y(1) = dty(1)
      LET y(2) = dty(INT(c / 3))
      LET y(3) = dty(INT(2 * c / 3))

      LET a(1) = ang(1) + cone(1)
      LET a(2) = ang(INT(c / 3)) + cone(2)
      LET a(3) = ang(INT(2 * c / 3)) + cone(3)

      LET lf#(1) = COS(a(1) * pi / 180)
      LET mf#(1) = SIN(a(1) * pi / 180)
      LET pf#(1) = mf#(1) * x(1) - lf#(1) * y(1)
```

```
      LET lf#(2) = COS(a(2) * pi / 180)
      LET mf#(2) = SIN(a(2) * pi / 180)
      LET pf#(2) = mf#(2) * x(2) - lf#(2) * y(2)
      LET lf#(3) = COS(a(3) * pi / 180)
      LET mf#(3) = SIN(a(3) * pi / 180)
      LET pf#(3) = mf#(3) * x(3) - lf#(3) * y(3)

objend:
END SUB

SUB StArrow (x, y, a, c)
'This subroutine is to draw the arrow of force outside the
object
'window
CONST pi = 3.141593

      WINDOW (0, 0)-(639, 479)
      LET xe = x - (640) / 30 * COS(a * pi / 180)
      LET ye = y - (640) / 30 * SIN(a * pi / 180)
      LET xa1 = x + (640) / 80 * COS(a * pi / 180 - 3)
      LET xa2 = x + (640) / 80 * COS(a * pi / 180 + 3)
      LET ya1 = y + (640) / 80 * SIN(a * pi / 180 - 3)
      LET ya2 = y + (640) / 80 * SIN(a * pi / 180 + 3)
      LINE (xa1, ya1)-(xa2, ya2), c
      LINE (x, y)-(xa1, ya1), c
      LINE (x, y)-(xa2, ya2), c
      LINE (x, y)-(xe, ye), c

END SUB
```

SOURCE CODE OF GRP

```
'*************************************************************
'* Title:       GRP.EXE                                      *
'* Programmer: Ratchatin Chancharoen                         *
'*             Graduate Student in Mechanical Eng. Dpt.      *
'*             Oregon State University                       *
'* Date:       30 March 1994                                 *
'* Purpose:    This program is to convert a gray             *
'*             scale image to a format that can be read      *
'*             from PRDA.The result will also saved in       *
'*             "OBJ.SRF".                                    *
'*************************************************************
DECLARE FUNCTION max! (a!, b!)
DECLARE FUNCTION min! (a!, b!)
DECLARE SUB ImageProcess ()
DECLARE SUB Appearance ()
DECLARE SUB Conversion3 ()
DECLARE SUB Conversion2 ()
DECLARE SUB Conversion1 ()
DECLARE SUB drwobj ()
DECLARE SUB picture (sizx!, sizy!, xy%(), x!, y!)
DECLARE SUB bitS (test%)

'array variables
DIM bit%(1 TO 16)
DIM dtx(1 TO 1000), dty(1 TO 1000)

'shared variables
COMMON SHARED pt%(), bit%(), lenx, leny
COMMON SHARED dtx(), dty()
COMMON SHARED rxx1, rxx2, ryy1, ryy2
COMMON SHARED c, file$

'main
    SCREEN 12
    CLS

'begin
    CALL Appearance 'program appearance
    CALL ImageProcess 'image process

DO
LOOP WHILE INKEY$ = ""

SUB Appearance
'Set the program appearance

SCREEN 12
```

```
CLS

    LINE (0, 0)-(639, 460), 6, BF
    LINE (170, 20)-(470, 250), 2, BF
    LINE (171, 21)-(469, 249), 0, BF
    LINE (20, 20)-(150, 180), 2, BF
    LINE (20, 200)-(150, 250), 2, BF
    LINE (20, 270)-(620, 440), 2, BF

    FOR i = 0 TO 216 STEP 49
        LINE (490, 20 + i)-STEP(60, 34), 2, BF
        LINE (560, 20 + i)-STEP(60, 34), 2, BF
    NEXT i

END SUB

SUB bitS (test%)
'Look at each bit of a number

    FOR i = 1 TO 16
        LET bit%(i) = 0
    NEXT i
    IF test% < 0 THEN
        LET bit%(1) = 1
        LET test% = test% + 32768
    END IF
    FOR i = 2 TO 16
        IF test% >= 2 ^ (16 - i) THEN
            LET bit%(i) = 1
            LET test% = test% - 2 ^ (16 - i)
        END IF
    NEXT i

END SUB

SUB Conversion1
'adjust color

    LET clr = pt%(1, 1)
    FOR j = 1 TO leny
        FOR i = 1 TO lenx
            IF pt%(i, j) = clr THEN
                LET pt%(i, j) = 0
            ELSE
                LET pt%(i, j) = 255
            END IF
        NEXT i
    NEXT j

END SUB
```

```
SUB Conversion2
'Find object pixels and contour pixels
DIM DOT%(1 TO lenx, 1 TO leny)

FOR y = 1 TO leny
    FOR x = 1 TO lenx
        LET DOT%(x, y) = 0
    NEXT x
NEXT y

FOR j = 2 TO leny - 1
  FOR i = 2 TO lenx - 1
     IF pt%(i, j) >= 8 THEN
        IF pt%(i + 1, j + 1) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i + 1, j) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i + 1, j - 1) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i, j + 1) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i, j - 1) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i - 1, j + 1) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i - 1, j) < 8 THEN LET DOT%(i, j) = 255
        IF pt%(i - 1, j - 1) < 8 THEN LET DOT%(i, j) = 255
      END IF
  NEXT i
NEXT j

FOR j = 1 TO leny
    FOR i = 1 TO lenx
        LET pt%(i, j) = DOT%(i, j)
    NEXT i
NEXT j

END SUB

SUB Conversion3
'Find a series of points

DIM nox(1 TO 200), noy(1 TO 200)

'find pt1
    LET c = 1
    LET i = 1
    LET j = INT((1 + leny) / 2)

    DO
        IF pt%(i, j) > 8 THEN EXIT DO
        LET i = i + 1
        IF i >= lenx THEN
            LET i = 1
            LET j = j + 1
```

```
        END IF
        IF j >= leny THEN LET j = 1
    LOOP

    LET dtx(1) = i
    LET dty(1) = j

'find pt2
    IF pt%(i + 1, j) > 8 THEN
        LET i = i + 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF

    IF pt%(i, j - 1) > 8 THEN
        LET j = j - 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF

    IF pt%(i - 1, j) > 8 THEN
        LET i = i - 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF

    IF pt%(i, j + 1) > 8 THEN
        LET j = j + 1
        LET dtx(2) = i
        LET dty(2) = j
    END IF

    IF pt%(i + 1, j - 1) > 8 THEN
        LET i = i + 1
        LET j = j - 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF

    IF pt%(i - 1, j - 1) > 8 THEN
        LET i = i - 1
        LET j = j - 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF
```

```
    IF pt%(i + 1, j + 1) > 8 THEN
        LET i = i + 1
        LET j = j + 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF

    IF pt%(i - 1, j + 1) > 8 THEN
        LET i = i - 1
        LET j = j + 1
        LET dtx(2) = i
        LET dty(2) = j
        GOTO aagain
    END IF

aagain:
'find pt3 to last pt
    LET c = 2

    DO
        IF dtx(c - 1) = i - 1 AND dty(c - 1) = j THEN

            IF pt%(i + 1, j) > 8 THEN
                LET i = i + 1
                LET dtx(c + 1) = i
                LET dty(c + 1) = j
                GOTO again
            END IF

            IF pt%(i, j - 1) > 8 THEN
                LET j = j - 1
                LET dtx(c + 1) = i
                LET dty(c + 1) = j
                GOTO again
            END IF

            IF pt%(i, j + 1) > 8 THEN
                LET j = j + 1
                LET dtx(c + 1) = i
                LET dty(c + 1) = j
                GOTO again
            END IF

            IF pt%(i + 1, j - 1) > 8 THEN
                LET i = i + 1
                LET j = j - 1
                LET dtx(c + 1) = i
                LET dty(c + 1) = j
```

```
            GOTO again
        END IF

        IF pt%(i + 1, j + 1) > 8 THEN
            LET i = i + 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

    END IF
    IF dtx(c - 1) = i + 1 AND dty(c - 1) = j THEN

        IF pt%(i - 1, j) > 8 THEN
            LET i = i - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i, j + 1) > 8 THEN
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i, j - 1) > 8 THEN
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF
```

```
    IF pt%(i - 1, j + 1) > 8 THEN
        LET i = i - 1
        LET j = j + 1
        LET dtx(c + 1) = i
        LET dty(c + 1) = j
        GOTO again
    END IF

    IF pt%(i - 1, j - 1) > 8 THEN
        LET i = i - 1
        LET j = j - 1
        LET dtx(c + 1) = i
        LET dty(c + 1) = j
        GOTO again
    END IF

    IF pt%(i + 1, j + 1) > 8 THEN
        LET i = i + 1
        LET j = j + 1
        LET dtx(c + 1) = i
        LET dty(c + 1) = j
        GOTO again
    END IF

    IF pt%(i + 1, j - 1) > 8 THEN
        LET i = i + 1
        LET j = j - 1
        LET dtx(c + 1) = i
        LET dty(c + 1) = j
        GOTO again
    END IF

END IF
IF dtx(c - 1) = i AND dty(c - 1) = j + 1 THEN

    IF pt%(i, j - 1) > 8 THEN
        LET j = j - 1
        LET dtx(c + 1) = i
        LET dty(c + 1) = j
        GOTO again
    END IF

    IF pt%(i - 1, j) > 8 THEN
        LET i = i - 1
        LET dtx(c + 1) = i
        LET dty(c + 1) = j
        GOTO again
    END IF
```

```
        IF pt%(i + 1, j) > 8 THEN
            LET i = i + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j - 1) > 8 THEN
            LET i = i + 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

    END IF
    IF dtx(c - 1) = i AND dty(c - 1) = j - 1 THEN
        IF pt%(i, j + 1) > 8 THEN
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j) > 8 THEN
            LET i = i - 1
```

```
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j) > 8 THEN
            LET i = i + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j + 1) > 8 THEN
            LET i = i + 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j - 1) > 8 THEN
            LET i = i + 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

    END IF
    IF dtx(c - 1) = i - 1 AND dty(c - 1) = j - 1 THEN

        IF pt%(i + 1, j) > 8 THEN
            LET i = i + 1
            LET dtx(c + 1) = i
```

```
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i, j + 1) > 8 THEN
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j + 1) > 8 THEN
            LET i = i + 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j - 1) > 8 THEN
            LET i = i + 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

    END IF
    IF dtx(c - 1) = i - 1 AND dty(c - 1) = j + 1 THEN

        IF pt%(i + 1, j) > 8 THEN
            LET i = i + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i, j - 1) > 8 THEN
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
```

```
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j + 1) > 8 THEN
            LET i = i + 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j - 1) > 8 THEN
            LET i = i + 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

    END IF
    IF dtx(c - 1) = i + 1 AND dty(c - 1) = j - 1 THEN

        IF pt%(i - 1, j) > 8 THEN
            LET i = i - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i, j + 1) > 8 THEN
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF
```

```
        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF
    END IF
    IF dtx(c - 1) = i + 1 AND dty(c - 1) = j + 1 THEN

        IF pt%(i - 1, j) > 8 THEN
            LET i = i - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i, j - 1) > 8 THEN
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j - 1) > 8 THEN
            LET i = i - 1
            LET j = j - 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i - 1, j + 1) > 8 THEN
            LET i = i - 1
            LET j = j + 1
            LET dtx(c + 1) = i
            LET dty(c + 1) = j
            GOTO again
        END IF

        IF pt%(i + 1, j - 1) > 8 THEN
```

```
                LET i = i + 1
                LET j = j - 1
                LET dtx(c + 1) = i
                LET dty(c + 1) = j
                GOTO again
            END IF

        END IF

again:
        LET c = c + 1
        IF dtx(c) = dtx(1) AND dty(c) = dty(1) THEN EXIT DO
    LOOP WHILE c < 1000

    FOR k = 1 TO 10
        IF c > 200 THEN
          FOR i = 1 TO 200
              LET nox(i) = dtx(INT((i - 1) * c / 200) + 1)
              LET noy(i) = dty(INT((i - 1) * c / 200) + 1)
          NEXT i
          FOR i = 1 TO 200
              LET dtx(i) = nox(i)
              LET dty(i) = noy(i)
          NEXT i
          LET c = 200
        ELSE
            LET nox(1) = dtx(1)
            LET nox(1) = nox(1) + dtx(2)
            LET nox(1) = nox(1) + dtx(c)
            LET nox(1) = nox(1) / 3

            LET noy(1) = dty(1)
            LET noy(1) = noy(1) + dty(2)
            LET noy(1) = noy(1) + dty(c)
            LET noy(1) = noy(1) / 3

            FOR i = 2 TO c - 1
                LET nox(i) = dtx(i)
                LET nox(i) = nox(i) + dtx(i + 1)
                LET nox(i) = nox(i) + dtx(i - 1)
                LET nox(i) = nox(i) / 3

                LET noy(i) = dty(i)
                LET noy(i) = noy(i) + dty(i + 1)
                LET noy(i) = noy(i) + dty(i - 1)
                LET noy(i) = noy(i) / 3
            NEXT i

            LET nox(c) = dtx(c)
            LET nox(c) = nox(c) + dtx(c - 1)
```

```
            LET nox(c) = nox(c) + dtx(1)
            LET nox(c) = nox(c) / 3

            LET noy(c) = dty(c)
            LET noy(c) = noy(c) + dty(c - 1)
            LET noy(c) = noy(c) + dty(1)
            LET noy(c) = noy(c) / 3

            FOR i = 1 TO c
                LET dtx(i) = nox(i)
                LET dty(i) = noy(i)
            NEXT i
        END IF
    NEXT k

    FOR i = 1 TO c
        LET dty(i) = -dty(i)
    NEXT i

END SUB

SUB drwobj
'Draw object

    LET rx1 = dtx(1)
    LET ry1 = dty(1)
    LET rx2 = dtx(1)
    LET ry2 = dty(1)

    FOR k = 2 TO c - 1
        LET rx1 = min(rx1, dtx(k))
        LET rx2 = max(rx2, dtx(k))
        LET ry1 = min(ry1, dty(k))
        LET ry2 = max(ry2, dty(k))
    NEXT k

    IF (ry2 - ry1) / 150 > (rx2 - rx1) / 220 THEN
        LET ryy2 = ry2 + (ry2 - ry1) / 150 * 60
        LET ryy1 = ry1 - (ry2 - ry1) / 150 * 270
        LET rxx1 = .5 * (rx1 + rx2) - (ry2 - ry1) / 150 *
320
        LET rxx2 = rxx1 + (ry2 - ry1) / 150 * 640
    END IF

    IF (ry2 - ry1) / 150 <= (rx2 - rx1) / 220 THEN
        LET rxx1 = rx1 - (rx2 - rx1) / 220 * 210
        LET rxx2 = rx2 + (rx2 - rx1) / 220 * 210
        LET ryy1 = ry1 - (rx2 - rx1) / 220 * 270
        LET ryy2 = ryy1 + (rx2 - rx1) / 220 * 480
    END IF
```

```
    WINDOW (rxx1, ryy1)-(rxx2, ryy2)

    FOR k = 1 TO c - 1
        LINE (dtx(k), dty(k))-(dtx(k + 1), dty(k + 1)), 3
    NEXT k

END SUB

SUB ImageProcess
'Conversion process

    OPEN "file.dat" FOR INPUT AS #1
      INPUT #1, file$
      INPUT #1, lenx
      INPUT #1, leny
    CLOSE #1

        DIM pt%(1 TO lenx, 1 TO leny)
        LET x = 1
        LET y = 1
        OPEN file$ FOR RANDOM ACCESS READ AS #1 LEN = 2
        LET maxf = LOF(1) / 2
        IF maxf = 0 THEN GOTO ProcessEnd
            COLOR 2
            LOCATE 4, 34: PRINT "FILE :"
            LOCATE 5, 34: PRINT "Running.."
            LOCATE 5, 47: PRINT "%"
            LOCATE 6, 34: PRINT "Reading.."
            LOCATE 7, 34: PRINT "Column :"
            LOCATE 8, 34: PRINT "Row    :"

            COLOR 14
            LOCATE 4, 41: PRINT USING "\        \"; file$

        FOR i = 1 TO maxf
            LOCATE 5, 43: PRINT INT(i / maxf * 100)
            LOCATE 7, 43: PRINT USING "#####"; x
            LOCATE 8, 43: PRINT USING "#####"; y

            GET #1, i, info%
            CALL bitS(info%)
            LET pt%(x, y) = 0

            FOR c = 1 TO 8
                IF bit%(c + 8) = 1 THEN
                     LET pt%(x, y) = pt%(x, y) + 2 ^ (8 - c)
                END IF
            NEXT c
```

```
            LET x = x + 1

            IF x > lenx THEN
                LET y = y + 1
                LET x = 1
            END IF

            LET pt%(x, y) = 0

            FOR c = 1 TO 8
                IF bit%(c) = 1 THEN
                    LET pt%(x, y) = pt%(x, y) + 2 ^ (8 - c)
                END IF
            NEXT c

            LET x = x + 1

            IF x > lenx THEN
                LET y = y + 1
                LET x = 1
            END IF
        NEXT i

        CLOSE #1

    CALL Conversion1
    LINE (171, 21)-(469, 249), 0, BF
    CALL picture(lenx, leny, pt%(), 240, 90)
    CALL Conversion2
    LINE (171, 21)-(469, 249), 0, BF
    CALL picture(lenx, leny, pt%(), 240, 90)
    CALL Conversion3
    LINE (171, 21)-(469, 249), 0, BF
    LET c = c + 1
    LET dtx(c) = dtx(1)
    LET dty(c) = dty(1)

'save the object
COLOR 2
LOCATE 9, 32: PRINT "Writing to.."
COLOR 14
LOCATE 9, 45: PRINT "Obj.srf"
LOCATE 10, 39: PRINT "Press <Enter>"

    SHELL "del obj.srf"
    OPEN "obj.srf" FOR RANDOM ACCESS WRITE AS #1 LEN = 4
    FOR i = 2 TO 2 * c STEP 2
        PUT #1, i - 1, dtx(i / 2)
        PUT #1, i, dty(i / 2)
    NEXT i
```

```
    CLOSE #1

    CALL drwobj
ProcessEnd:

END SUB

FUNCTION max (a, b)
    IF a >= b THEN LET max = a
    IF a < b THEN LET max = b
END FUNCTION

FUNCTION min (a, b)
    IF a >= b THEN LET min = b
    IF a < b THEN LET min = a
END FUNCTION

SUB picture (sizx, sizy, xy%(), x, y)
'Draw object

    FOR j = 1 TO sizy - 1
        FOR i = 1 TO sizx
            IF xy%(i, j) > 8 THEN PSET (i + x, j + y), 1
        NEXT i
    NEXT j

END SUB
```

SOURCE CODE OF HELP

```
'*************************************************************
'* Title:     HELP.EXE, a program to help beginner using FA*
'* Programmer: Ratchatin Chancharoen                        *
'*             Graduate Student in Mechanical Eng. Dpt.     *
'*             Oregon State University                      *
'* Date:       30 March 1994                                *
'* Purpose:    This program is to view the document         *
'*             named "readme.txt". This document contains   *
'*             the detail of FA force analysis software.    *
'*************************************************************
DIM line$(1 TO 1000)
SCREEN 9
'Draw the screen appearance

     LINE (0, 0)-(639, 349), 6, BF
     LINE (0, 0)-(639, 349), 14, B
     LINE (50, 70)-(579, 339), 14, B
     LINE (50, 20)-(579, 60), 7, BF
     LINE (50, 20)-(579, 60), 14, B

     COLOR 12, 7
     LOCATE 3, 10: PRINT "Command keys:"
     LOCATE 3, 27: PRINT "Page Up"
     LOCATE 4, 27: PRINT "Page Down"
     LOCATE 3, 41: PRINT "Arrow Up"
     LOCATE 4, 41: PRINT "Arrow Down"
     LOCATE 3, 55: PRINT "P to Print"
     LOCATE 4, 55: PRINT "Escape to eXit"

'Read the document named readme.txt

     LET file$ = "a:\readme.txt"
     OPEN file$ FOR INPUT AS #1
          LET i = 1
          DO WHILE NOT EOF(1)
               INPUT #1, line$(i)
               LET i = i + 1
          LOOP
          LET N = i
     CLOSE #1

'Display text in the document

     LET x = 0
     DO
          LINE (51, 71)-(578, 338), 7, BF
          FOR i = 1 TO 17
```

```
            LOCATE i + 6, 13: PRINT line$(x + i)
        NEXT i
        LET key$ = ""
        DO
            LET key$ = INKEY$
        LOOP UNTIL key$ <> ""
          LET front = ASC(key$)
          LET hind = ASC(RIGHT$(key$, 1))

        IF front = 0 AND hind = 80 THEN
            IF x < N THEN
            LET x = x + 1
            END IF
        ELSEIF front = 0 AND hind = 72 THEN
            IF x > 1 THEN
            LET x = x - 1
            END IF
        ELSEIF front = 13 AND hind = 13 THEN
            IF x < N THEN
            LET x = x + 1
            END IF
        ELSEIF front = 0 AND hind = 81 THEN
            IF x < N - 30 THEN
            LET x = x + 16
            ELSE
            LET x = N - 16
            END IF
        ELSEIF front = 0 AND hind = 73 THEN
            IF x > 17 THEN
            LET x = x - 16
            ELSE
            LET x = 0
            END IF
        ELSEIF front = 27 AND hind = 27 THEN
            GOTO exitloop
        ELSEIF front = 25 AND hind = 25 THEN
            SHELL "copy prnt.txt prn"
            SHELL "copy prnt.txt lpt1"
        END IF
    LOOP WHILE front <> 120 AND hind <> 120

exitloop:
'Clear the screen and exit
CLS
COLOR 7, 0
```