ıc ı

DESIGN AND DEMONSTRATION OF A DIRECT MANIPULATION INTERFACE

by

DAVID H. TINGLEY

B.S., Morris Harvey College, 1974
B.S., Florida Institute of Technology, 1982

A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTERS OF SCIENCE

Department of Computing and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Approved by

[illegible]
Major Professor

Table of Contents

Chapter 1 - Introduction.................................. 2

1.1 Purpose.. 2

1.2 Overview of Company Model...................... 2

1.3 Graphical Objects................................. 5

1.4 Organization of Report.......................... 7

1.5 Significance of Demonstration................ 8

Chapter 2 - Literature Review......................... 10

2.1 Introduction.. 10

2.2 General... 10

2.3 Components of the Interface Design........... 12

2.3.1 Linguistic vs Spatial Model................. 13

2.3.2 Time and Rate................................ 13

2.3.3 Messages..................................... 14

2.3.4 Tutorials...................................... 14

2.3.5 Devices.. 15

2.4 Linguistic Models................................. 16

2.5 Spatial Models..................................... 18

2.5.1 Features/Definitions......................... 18

2.5.2 Problems....................................... 21

2.5.3 Icon Design.................................... 22

2.5.4 Design Principles............................ 27

2.6 Related Work....................................... 30

Chapter 3 - Conceptual Model.............................. 33

3.1 Introduction.. 33

3.2 Office Model Definitions............................. 33

3.3 Entities.. 34

3.3.1 Data Entities.................................. 34

3.3.2 Functional Entities........................... 36

3.3.3 Person Entities................................ 38

Chapter 4 - Direct Manipulation Interface............... 39

4.1 Introduction.. 39

4.2 Overview.. 39

4.3 Top Level Model...................................... 39

4.4 File Management...................................... 41

4.5 Routing... 48

Chapter 5 - Implementation............................... 52

5.1 Introduction.. 52

5.2 Hardware/Software Tools............................. 52

5.3 Mouse Interface...................................... 52

5.4 Data Structure....................................... 53

5.5 Software Design...................................... 54

5.6 Statistics.. 57

References..

Appendis A: Source Code..................................

LIST OF FIGURES AND TABLES

Table 1.1 Principles 3

Figure 1.1 Organizational Chart........................ 4

Figure 2.1 Work by [Lod83]........................... 24

Figure 4.1 Top Level Organization Chart.............. 40

Figure 4.2 Manufacturing Organizational Chart........ 43

Figure 4.3 Example of Filing Cabinet and
Display Screens.......................... 44

Figure 4.4 Example of a Data Entity Screen........... 46

Figure 4.5 Example of a Routing Model................ 47

Figure 4.6 Message Screen............................ 51

CHAPTER 1 - INTRODUCTION

1.1 Purpose

The purpose of this report is to demonstrate the concepts of a graphical oriented user interface. These concepts are demonstrated with a model that implements certain office tasks. This interface is designed along the lines of the direct manipulation concepts as discussed in [Shn82a], [Shn82b], [Smi82], and [Sib86]. This interface also adheres closely to the WYSIWYG (What You See Is What You Get) concepts in terms of semantic expressions of the tasks to be rendered. Various forms of iconic interfacing as discussed by [Lod83] are also demonstrated. Finally, the principles addressed by [Smi82] are utilized in the design phase of the interface for this model. These principles are listed in Table 1.1.

1.2 Overview of Company Model

This hypothetical business is a chemical company that has two major subdivisions: manufacturing and research. Figure 1.1 shows the organizational chart for this company. Each subdivision is further broken down into departments as shown. Each department has a workstation and a database associated with it. In addition, each department is connected to the others in a hierarchical

Table 1.1

* Familiar user's conceptual model
* Seeing and pointing versus remembering and typing
* What you see is what you get
* Consistency
* Simplicity
* Modeless interaction
* User tailorability

The contents of this table are copied from an article by [Smi82].

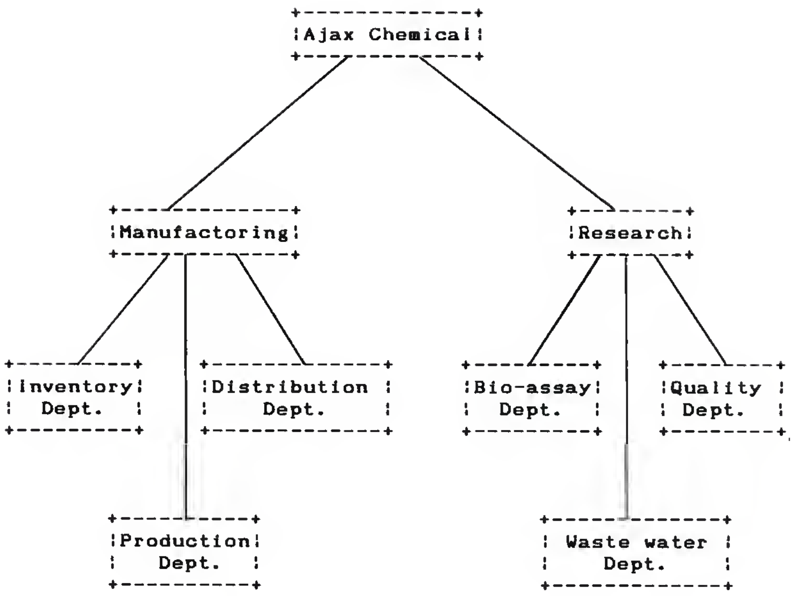


Figure 1.1 Organization Chart

connected network which matches the organizational diagrams.

The office activity being simulated is the sending and receiving of one line electronic messages between the different departments. Each workstation is equipped with a personal computer, keyboard, graphics quality screen, and a mouse.

A means of traversing through the hierarchical structure of the company is provided through the use of the mouse. The user can perform two distinct operations at any of the workstations. The first operation is a means of graphical accessing a database for the purpose of file manipulation and second is a means of routing these files to a variety of people within the business organization.

1.3 Graphical Objects

The user is confronted with three basic graphical objects with which to work with at any given workstation. These objects are data entities, functional entities, and person entities. The following is a discussion of these objects:

Data entities	The data entity is an object that take the form of either a text document or a fill in the blank form. The data entity is the vehicle by which all inter-department communication is achieved. Thus, it is the data entity that will be routed to the appropriate individuals within the organization.
Functional entity	The purpose of the functional entity is to act as a means of support for the data entities. Functional entities are broken down into five classes: 1) Filing cabinets 2) Nodes 3) Printers 4) File servers 5) Ports Filing cabinets act as a depository for data entities. The user opens the filing cabinets to store or retrieve a data entity. Nodes are used to represent individual departments. Printers are used to generate hardcopy

of a data entity image. File servers act as hubs for nodes. Their use achieves a logical clustering of departments around the major subdivisions mentioned earlier. This arrangement has been observed by the author to be similar to actual companies using networked personal computers to address their information needs. The final class of functional entities are ports. Ports act as a doorway between the two subdivisions and their associated departments.

Person entity	Person entities represent people who usually perform some sort of "action" on a data entity. Person entities are grouped along departmental patterns. This means that selection of a recipient for mail is based upon what department they are assigned to.

1.4 Organization of Report

This report is organized into five chapters; Chapter 1 -

The Introduction, Chapter 2 - The Literature Review, Chapter 3 -The Direct Manipulation Interface, Chapter 4 - The Visual Implementation, and Chapter 5 - The Implementation. Chapter i has already been discussed. Chapter 2 is a survey of articles dealing primarily with the human machine interface and the emergence of the concepts used in a direct manipulation interface. Chapter 3 provides a discussion of the model used to demonstrate the direct manipulation interface. In particular, this chapter addresses the details associated with the two major subsystems: File Management and Routing. Chapter 4 offers a discussion of the actual visual interfaces offered to the user. This includes instructions on the use of the direct manipulation tools associated with each instance of a visual screen. Chapter 5 is concerned with the hardware considerations and the software design and data structures associated with the actual implementation.

1.5 Significance of the Demonstration

The significance of the demonstration is best understood with the following observations.

i) The dissemination of information with many companies, regardless of size, is still being done manually. A common procedure is to select or create an item of correspondence, attach a routing list of names, and send this item via traditional (manually) mail. This scenario is usually slow and at times unreliable.

2) The very nature of the manual mail problem suggests that it might be resolved through the use of a natural, visual interface model of the message and routing systems involved. Therefore, an effort is made through the demonstration to show an application of an interface form to a real commercial need.

3) The operator of this interface, usually the person who is currently doing the job manually, will be able not only be able to take advantage of their visual processing skill via these computer tools but also the learning curve associated with these tools is minimal due to the semantic nature of the model. This observation is documented in [Led80], [Car80], and [Shn82a].

CHAPTER 2 - LITERATURE REVIEW

2.1 Introduction

This chapter reviews literature related to the interactive interfaces between the user and the computer. Models of the user interface fall into two broad categories: linguistic models and spatial models [Sib86]. The linguistic models view the interface as a dialog between the user and the computer. The spatial model deals more with the interactive graphic or direct manipulation model. It is the intent of this review to focus on the spatial models though some attention is paid to the linguistic models for the purpose of comparison.

2.2 General

[Ste87a] notes that there are three major concerns that confront the human factor aspect of manufacturing a product for an end user population:

1) The first concern is that more people are using computer products to aid them in their job. This increase in the computer user population will be comprised mainly of novices and non-technical/non-computer types of individuals. In the "old days", programmers designed text editors, operating system

job control languages, programming languages, and application packages for themselves and their peers [Shn82]. Now, with personal computers, office automation , automated spread sheets and other such products, a new type of user is born. This user is more concerned with what can be done on a machine as opposed to how the machine works. It is helpful to look into what was done prior to direct manipulation in terms of the human factors aspect. It has been noted by [Led80] that we have forgotten that the original objective of computer technology was not to develop more powerful systems but to increase the overall effectiveness of the human problem solver. It has to be recognized that in the years to comes these human problem solvers may not and should not have to be "computer experts" in order to solve their problems in their domains.

2) The second major concern is about software interfaces that are interactive. Originally, software was written to implement batch orientated jobs such as payrolls and account functions and was shown to be a cost/benefit improvement. Now, such systems as on-line reservations and inventories require the

interface to be understandable and available at all times.

3) Finally, there is a need to develop interfaces that perform in highly critical functions such as monitoring nuclear reactor site and intensive care units in hospitals. Strategies for interfaces must be developed that immediately impart to the user the state of affairs. In addition, these interfaces must keep the application error rate at a extremely low rate for obvious reasons.

2.3 Components of the Interface Design

Studies by [Shn82] have indicated that the interface design effort can be classed into five groups or areas of concern:

1) General linguistic model vs spatial model
2) Response time and display rate
3) Wording of system messages
4) Online tutorials and messages
5) Hardware devices (as applied to spatial interfaces)

2.3.1 Linguistic vs Spatial Model

It is argued by [Shn82] that spatial models eliminate the need for the user to memorize the syntax for various options of a command. This approach is of particular interest for non-technical users who may not have all the various features of a command committed to memory. [Zlo82] has noted that there are two major problems to a menu type of interface. The first problem deals with the fact that the more complicated a system the more complex the "march" through the menus becomes. In addition, he says that menus lack flexibility in that the user is constrained to those selections available on the menu. An additional option
would cause design changes and its associated overhead. This lack of flexibility applies not only to menu options but also to the actual menu formats themselves (i.e. - vertical and horizontal selection arrangements).

2.3.2 Time and Rate

[Shn82] has presented a set of guidelines that address the response time/display rate issues. He feels that typing and cursor motion should be generated in .1 second , frequent commands should execute in less than a second, response time for similar commands should not deviate more than 20% from the mean, and quicker response is not always good as the incidence of errors increases. It has been observed that novice users actually prefer a slower response time. Guidelines for display rates are the elimination of erratic display rates, display rates for text that must be entirely read does not need to exceed the actual reading rate and faster rates are better for material in which only a small portion is of interest.

2.3.3 Messages

Wording of system messages concerns the implementation of messages generated as the result of an inappropriate user action. These quite frequently occur when a user wanders into areas of unfamiliarity or during the steepest portion of the learning curve. Guidelines for addressing system messages include using a positive tone and attitude when generating system messages, using terminology that is within the grasp of the user, allow the user to be in control of the situation, and use of a neat and consistent format. It is suggested that an acceptance test be administered to the appropriate user groups for validation.

2.3.4 Tutorials

Tutorials, explanations and messages have arisen because of the variety of user's today and offer an excellent source of information for the infrequent or novice user.

The use of online tutorials eliminates potential disruption caused by having to physically look up something in a manual. A continuation of this is to create a "window" on the screen that allows the user not only to perform the task but also to see the instructions at the same time. It is suggested that separate error messages should be generated for each user group from the novice to the expert [Shn82].

2.3.5 Devices

Finally, the abundance of hardware options has allowed for four logical device types to be a basis for design [Fol&Wal74].

These devices are:

1) Pick - a mechanism for picking from a set of displayed entities.
2) Valuator - a device for setting numeric values.
3) Locater - a way of specifying in two and three dimensional space.
4) Button - a selection device for initiating or terminating action.

In work by [Car78], four pointing devices were tested and compared : a mouse, a joystick, step keys, and text keys. It was discovered that the mouse did the best job in all areas of consideration except the time it takes to actually place your hand on a pointing device. These areas include the effect of distance and target size, effect of approach angle, and overall errors in use.

2.4 Linguistic Models

Linguistic models are often characterized by the use of keyboards to convey textual, line orientated commands to the computer. The user types in a command, usually followed by a return, and waits for the computer to respond. One example of work on the improvement of linguistic models is by [Led80]. He has indicated that the user performance was dramatically increased in text editing by the use of an English language editor as opposed to a notational editor. The hypothesis for this experiment was the following : An interactive system should be based on familiar, descriptive, everyday words and legitimate English phrases. The use of the English language as a standard for worldwide communications is well documented [McR86]. The experiment involved two functional equivalent editors, one with an English-like interface and one with a notational interface. A group of paid volunteers were divided into three groups according to their abilities and asked to perform a variety of editing tasks. The results of the experiment showed that the mean percentage of editing completed and the mean efficiency increased with the use of the English-like editor and opposed to the notational editor. Also, the mean percentage of erroneous commands as less form the English editor. During the course of this experiment an interesting observation was made. It seemed that the users made no distinction between the syntax and the semantics of the editors. The subjects were unable to conceive of editing power as something different from the actual command. The commands so personified the editors to a positive or negative degree that the subjects were surprised to learn that both editors were functionally equivalent. This observation shows the importance of surface syntax or the "looks" of a command in designing an interface. This study shows a path of improvement for the linguistic models by making the interface more English-like and crafting the commands to convey it's functionality.

2.5 Spatial Models

2.5.1 Feature/Definitions

As noted earlier, the spatial models include the interactive graphics and direct manipulation models. [Rae85] has noted that the human mind is strongly visually orientated and that information is more quickly acquired by discovering the graphical relationships between complex pictures than by reading text. He attributes this concept to the following observations:

1) Random vs sequential access - reading text is a sequential action whereas random access can be obtained to any part of a picture.
2) Degree of expression - text is a one dimensional stream of words. These word must then be interpreted to gradually form a picture.
3) Transfer rate - one picture says a thousand words.
4) Concrete vs abstract - using real world objects to demonstrate abstract ideas help make these ideas easier to think about.
5) Picture without names - pictures do not need names and thus a second level of reference is not required.
6) Real world pictures - pictures reflect what is around us whereas text can only point to the real

world.

The above attributes have prompted not only an interest in graphical applications but also what can be done to current and new interfaces.

[Shn82] was able to summarize responses to systems that received positive user reaction. The central ideas according seemed to be the visibility of the object of interest, rapid reversible actions and replacement of complex command language syntax by direct manipulation of the objects of interest - hence the name direct manipulation. He notes that the direct manipulation model can be thought of in three terms:

1) There is a continuous representation of the object of interest.
2) Physical actions or labelled button presses instead of complete syntax.
3) Rapid incremental, reversible operations whose impact on the objects of interest is immediately visible.

The use of the above principles in system design lends a larger degree of user friendliness to an interface. The background for this, as stated earlier, is the syntatic/semantic model. It is easier to remember actions to be performed on an object rather than the syntax to implement the same task. Dealing with tasks as a set of actions on a object is a natural way of doing things. The use of pictures arose long before the use of words. The underlying component of direct manipulation can be described in a phrase coined by Don Hatfield of IBM " What You See Is What You Get " (WYSIWYG).

The previous discussions dealt with the development of direct manipulation. However, [Whi85] in a holistic approach to comparing command, menu and iconic interfaces, made the following observations:

1) There is a large variety of user interfaces available.
2) There seems to be no trade off between how easy the system is to use versus how easy the system is to learn.
3) The type of interface does not relate to preference or performance.

It should be noted that observation three is caveated with a statement that says that careful design of the interface is as important as the interface style itself. The philosophy of this work is that a system must be tested as a whole and not the decomposition into individual facets of the system. Seven different interface styles were tested by [Whi85], and the results indicate that though performance does not seem to be effected by interface style an iconic system had the greatest universal appeal across the user domain.

2.5.2 Problems

The direct manipulation interface is a relatively new but some of it's problems and weak areas have already surfaced.

[Shn83] noted the following shortcomings:

1) There was no noticeable increase in performance as compared to traditional linguistic models.
2) Learning the meaning of the graphical components could be a problem.
3) The same graphical icon could have different meanings to different people.
4) Graphical icons can take up a lot of space on a CRT screen.

In addition, [Whi85] notes that successful input with a mouse on an iconic system requires a rather complete knowledge of the syntax associated with the mouse position, number of button clicks, timing of the clicks, and button choices. Furthermore, he came up with results similar to [Sch83] in terms of user performance. Spatial models (direct manipulation models) also lack straight forward syntatic mechanisms for sequencing events such as those provided by the linguistic (command line) models [Sib82].

This project hopes to overcome some of the above mentioned problems by creating a facility for limited command line dialog.

2.5.3 Icon Design

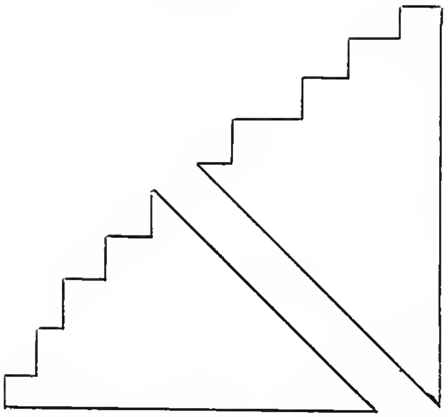
A common theme that runs through the direct manipulation literature is the problem with associating the desired meaning with an appropriate symbol. [Lod83] finds that it is hard to determine the degree to which we can rely on a particular image to carry a specific message is hard to determine. Images that bear a close resemblance to a particular object or task can be generated easily. The situation gets worse with objects or tasks that are not easily rendered into graphical icons. [Lod83] feels that correct interpretation of an image requires the following design considerations:

1) Image code
2) Caption
3) Context

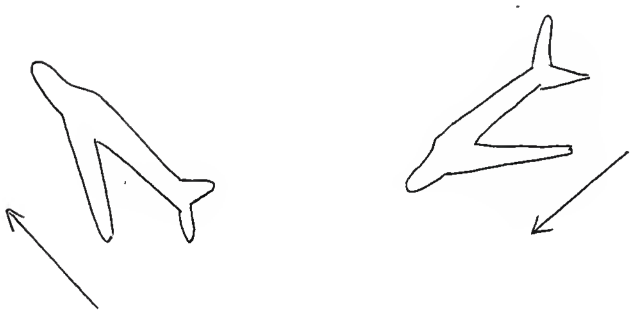
The image code is the image itself. The user must be able identify the concept or object that the image is trying to portray. The caption adds a redundant written assist. The context refers to the frame of reference for interpreting the icon. Figure 2.1 (a) shows an example of an icon that has no meaning without context. This project uses the above considerations in developing the set of icons that are used. It should be noted that icons unlike words must be created as there is no iconic dictionary. Finally, it is important that an icon does not impart some undesirable or unwanted message. Figure 2.1 (b) represents an example of an undesirable meaning. This figure represents icons used at airports in the early 70's.

An aid in eliminating the confusion associated with image meaning is to develop a taxonomy of icons. [Arn69] notes that there are three functions that images support: picture, symbol, and sign. [Lod83] pairs these functions with the image design style listed below.

Design		Function
Representation	–	Picture
Abstract	–	Symbol
Arbitrary	–	Sign



(a)



(b)

These figures are copied from the work by [Lod83].
Copyright 1983 IEEE 0272-1716/83/0300-0011

Figure 2.1

The representational style of icons is usually a fairly concrete image that is closely associated with an object.

Icons along the interstates such as a knife, fork, and spoon and gas pumps easily portray their intended meaning. The abstract icon is used to impart a concept as opposed to an object. An uparrow on a box is used to show which end of the box is supposed to up. The arbitrary icon is one which no attempt is made to convey a meaning. The state driving test requires the applicant to identify a number of road signs that have no lettering on them. Correct identification is made solely through the association of the sign's shape and its intendeo meaning. During the design phase, considerations needs to be paid not only to the artistic content of the icon but also the icon grouping. Tasks or functions that are similar in nature should be represented by similar icons. The three types of iconic designs discussed above are implemented in this project.

In addition to the three types of icon design mentioneo above, there are several factors characterize a good icon design. These include recognition, appropriateness as to designed task or function, and the lack of unpleasant association [Lod83]. In addition, the use of a semantic differential method of measurement discussed by [Car80] can be employed. This effort is characterized by an attempt to statically measure the degree of association between a printed message and it's iconic counterpart. The observation is made that given a final set of similar icons to choose from, the selector will make a choice based on preference. It is shown in [Dew&El177] and [Zaj68] that this is not a good measurement of sign meaning and that outside factors contribute heavily to a preference choice. The elimination of this preference choice is achieved through the use of semantic differentiation. This concept says that an icon and it s meaning can be represented as two objects within a semantic space. The closer the two objects are together in this semantic space the greater the degree of association. This semantic space can be defined by adjective pairs grouped into three classes: Evaluative (good/bad, just/unjust), Potency (big/small, strong/weak) and activity (on/off, quick/slow).

Three experiments were performed by [Car80] using the above model. The first was a means of selecting suitable pairs of adjectives. The next experiment measured the degree of association between the icons and their intenoed messages. The finally experiment attempted to find what contribution each individual part of the icon played in the overall perception. The major contribution of this work to my project is the observations of the last experiment. This experiment attempted to identify significant components of an icon structure. It was found, for example, that a telephone receiver actually conveys the message of a telephone better than a picture of the entire unit.

2.5.4 Design Principles

The design of this project was influenced by the seven principles as enumerated by [Smi82] and related in Chapter 1.

These principles are discussed below:

1) The use of a familiar user conceptual model is an approach in which a physical analogy or metaphor is used to convey a desired function. Apple's MacIntosh uses a trash can in which various icons (such as a file icon) can be deposited. This achieves the result of deleting that icon from the system. This project uses several physical metaphors to help implement the conceptual model. These include buttons, attenuators

and controls. Also, communications between various icons are conveyed via directional line on the screen.

2) The seeing and pointing principle states that witn a picture worth a thousand words, it is best to display as much of the functionality of the system as possible. This acts as a "visual cache" for the limited amount of short-term memory that humans possess. The Star user interface has a property and option list are very similar to the condition and menu selection options of this project in that they both impart attributes to objects on the screen.

3) The WYSIWYG principle, discussed in detail earlier, is utilized in this project. This is shown with the representation of various icons on the screen and their various interconnections and attributes.

4) Consistency deals with the idea that all actions or tasks are achieved the same way regardless of the state of the interface.

5) Simplicity is one of those principles that all designers hope to adhere to but most seldom do. The

concept of having no more than one way of doing something is applied through out my project. This literature, [Smi82], differs from my project in that a three button mouse is used instead of a two button mouse. The author feels that if a need arises then the various mouse button combinations (8 total) can be displayed as a continuous message on the screen as an aid to short term memory.

6) The modeless interaction principle states that there are not multiple functions for the same key depending on which mode the users is in. The UNIX vi editor is an example of a mode dependent text editor. The letter i, for example, when depressed can either generate the character "i" or put the user into insertion mode. This mode shifting can cause confusion and undesirable results. It should be noted however that this project does demonstrate certain mode related aspects. The number and type of actions that a user can engage are limited to the context of the menu or property assignment selections being made. In addition, the various mouse button selections can be viewed as taking on mode characteristics of a finer granularity.

7) User tailorability allows the user to modify the working environment in order to meet the goals of a given task. This principle is reflected in this project by the ability to assign attributes and properties to various iconic object on the screen.

2.6 Related Work

This section deals with a brief survey of other office automation systems tools that utilize a graphical interface. These include a generator of direct manipulation office systems, tools that aid in the meeting process, and means of graphically querying a database. Work by [Hud86] discusses a system for generating direct manipulation office systems. This system is known as HIGGENS - The Human interface Graphical Generating System. Higgens treats data as an active object not only containing data but also consisting of a description of the semantics of the data and means to the implement the semantics. Their approach has much in common with the general principles of object orientated programming systems and knowledge representation. The system uses the semantics of the application to check boundaries, derive new data, and update the screen. The use of attribute grammar and incremental attribute evaluation is used to implement the data models. It should be noted that this work concentrates primarily on the algorithms and techniques used by the run-time component of the system to support the generated office system.

The next system was prompted by the observation that office workers spend anywhere from 30%-70% of their time in meetings. To this end, an experimental meeting room was developed at Xerox's Palo Alto Research Center. This system, known as Colab, utilized a graphical interface and was networked over a group of workstations. A workstation is assigned to each participant in the meeting and they all have a portion of their screens that are common to each other. Work by [Ste87] developed two major tools for use on the Colab system. These tools were based on the assumption that the meeting process can be broken down into three distinct areas: brainstorming, organizing, and evaluating. The first , Cognoter, is used as an aid in organizing and preparing presentations. This tool allows for a collective effort by the participants to establish some sort of order over a group of individual ideas. Fore example, where in a project report would the discussion of the tools and hardware environment be included. Cognoter allows individuals to enter ideas on a common screen witn supporting text windows. These ideas are then moved around on the screen in to some form of order and are connected by arrows. The second grapnical meeting tool that is used is known as Argnoter. [Ste87] notes that discovering, understanding, and evaluating of proposals aid in making informed decisions. Graphical tools have been developed that aid in the proposal making stage, the argument stage, and the evaluating stage.

Another direct manipulation graphical interface is that of browsing a database. [Lar86] notes that four basic operations occur when browsing a database : structuring, filtering, panning and zooming. These are defined as chosing a finite set of object classes, selecting an occurrence form these object classes, investigating neighboring object occurrences and determine the degree of detail with which an object occurrence is to be studied. Examples of all four of the above operations are demonstrated in this project. Though the above system described by [Lar86] is based on alphanumeric representations, he mentions the need for a graphical interface. Current commercial products such as hypercard aid in the advancement of this approach.

CHAPTER 3 - CONCEPTUAL MODEL

3.1 Introduction

This chapter deals with certain concepts of the office environment in which we wish to apply a direct manipulation interface. The two topics addressed are the familiar user reference model of the office such as messages and file systems and the entities that model the office concept.

3.2 Office Model Definitions

In developing a background for a message distribution system the following items are defined

1) Filing System - a database from which preexisting messages can be either retrieved or created
2) Messages - an instance of an object selected from the filing system
3) People - the ultimate destination of a message
4) Mail - the vehicle by which messages are delivered to targets
5) Responses - certain replies or notification of "no-response" triggered by the sending of mail

3.3 Entities

Entities are the models used to convey the various tasks associated with a message distribution system. There are three types: data, functional, and person.

3.3.1 Data Entities

The data entity takes the form of either a pre-printed document requiring data entry or a textual type of document. The first item can be thought of as a entity that uses a 'fill in the blank' approach generated by prompts or questions on the form itself (e.g. -reporting the results of test or laboratory test forms). A textual document , on the other hand, is a means of communicating ideas, thoughts, decisions, and comments in a narrative manner.

Data entities can be further divided into text data entities and form data entities. The first is comprised of such things as letters, memos, reports and any other non-form related documents. The second is comprised of preformatted question type of forms requiring operator input. This includes such things as test reports, surveys, questionnaires, and inventory records. Both of the above mentioned entities can have any combination of the following properties or attributes.

1) Decisions

2) Comments

3) Administrative action

4) General information

5) Required response time

6) Route

The decision attribute signifies that the data entity contains one or more items that require a yes/no type of decision. An example of this might be a purchase order form requiring management approval. The comment attribute is used when a written response is required to information contained in the data entity. This attribute can be used as a vehicle to gather information from various individuals within an organization. The administrative action attribute is used when some form of clerical function is need. Filling in information on a blank form template is an example of this attribute. Required response time, can also be assigned to the data entity. This attribute is utilized when time dependent/ time critical constraints are required. The route attribute is used to determine where the data entity will eventually be targeted.

3.3.2 Functional Entities

The functional entities are entities that support actions performed on data entities. They are listed as follows.

1) Filing cabinets
2) Nodes
3) Printers
4) File Servers
4) Ports

The filing cabinets are receptacles in which data entities are stored. They are broken down into two drawers, one for the text material and one for form type of material. Filing cabinets supports four of the six phases of paper handling as report by [Cho82]. These phases are:

1) creation of the document
2) its physical or logical presence
3) storage
4) retrieval

the last two phases are handled by the routing mechanism

5) distribution within the organization
6) distribution external to the organization

Data entities, when retrieved, are placed in a temporary location know as a clipboard. It is while the data entity information is resident on the clipboard that operations such as editing, printing and routing can be performed.

Nodes, as used in this project, represent unique organizational units within the firm. In this case, they represent departments. Each node has an associated filing cabinet that acts as a database for that department.

File servers act as a hub for nodes in trafficking data entities through the system and have filing cabinets associated with them. Printers perform their function on the current file that has been retrieved (i.e - it prints the contents of the clipboard). The final entity in this group is know as a port. Ports provide the facility to not only move up and down within the organizational chart but also acts as an outlet to the environment external to the organization.

There are three operations associated with the file management subsystem. The first is the ability of the operator to move through the organizational chart via the ports and the use of the mouse. The second is file manipulation as discussed earlier. The last operation is the ability to print files (data entities).

3.3.3 Person Entities

Person entities can be thought of as a group of individuals that have a predetermined relationship to each other. Messages, obtained from the file system can be routed to selected individuals within these organizational charts. There are two selection operations associated with the routing of a message.

1) Random selection

2) Group Selection

Random selection is used to pick a finite group of people fro receipt of a data entity. This allows not only for individual selection but also for selection of individuals outside the constraints of their organization structures. Group selection is used as a means of broadcasting common interest messages to individuals within an organization. For example, a data entity could be send to all members of the Quality Assurance Department.

CHAPTER 4 - DIRECT MANIPULATION INTERFACE

4.1 Introduction

This chapter presents the visual interface of what the user will see in the performance of direct manipulation actions upon the previously described information distribution model. This chapter does not address any of the implementation details but rather performs a mapping of the representations discussed in Chapter 3 onto the screen displays and controls that the user has access to.

4.2 Overview

A top level screen is utilized that allows the user to gain access to various organizations and departments within the-company. The file management screens represents a modified organizational chart of the chemical company and the routing screens takes the form of a personnel organization chart.

4.3 Top level model

The top level screen is used as a mechanism in which to gain entry into either the file management or routing models. Figure 4.1 show the top level screen layout. The user positions the mouse cursor over either of the organization choices and depresses the left mouse button

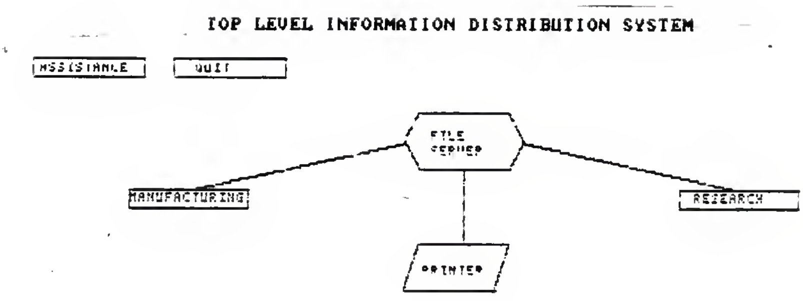


Figure 4.1 Top Level Organizational Chart

to activate the desired screen (henceforth known as clicking the mouse button). There are two additional direct manipulation buttons on this screen. The first, located in the upper left hand corner of the screen, is labeled assistance. Clicking this button causes a window to appear containing guidance and information on the operations available for a given screen. The second button is the quit button and clicking it allows the user to return to the operating system.

4.4 File Management

The file management model is composed of three organizational screens. The functional entities discussed in Chapter 3 are utilized with these screens. The following is a brief graphical description of each:

1) Filing cabinets - an icon representing a filing cabinet
2) Nodes - squares (used to represent departments)
3) Printers - trapezoids
4) File servers - hexagons
5) Ports - buttons that allow the user to traverse through the system

Figures 4.1, and 4.2 show two of the organizational screens available. Traversing through these screens is achieved by placing the mouse cursor over the return button and clicking the left mouse button. The new location within the organizational chart is then presented (i.e. - a new screen is generated).

Filing cabinets act as information storage recepticles and each node and file server has a cabinet associated with it. In order to access the filing cabinets, the user places the mouse cursor over either a node or a file server and clicks the left mouse button. This causes a file cabinet icon to appear. The drawers of the cabinet can then be opened via the left mouse button and the files displayed as a series of file folders. An example of a filing cabinet and a display of files is given in Figures 4.3 (a) and 4.3 (b).

A file folder can be opened by positioning the cursor over the desired file and file and clicking the left mouse button. This allows for one of the two types of data entities to then become available for editing. The data entity screen is composed of three buttons and an area for the actual text or form. The first button, located in the

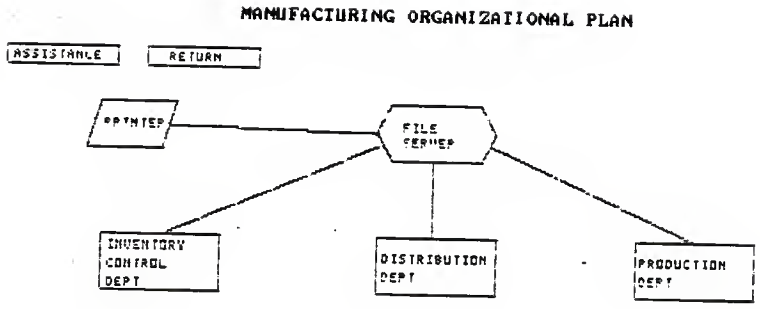
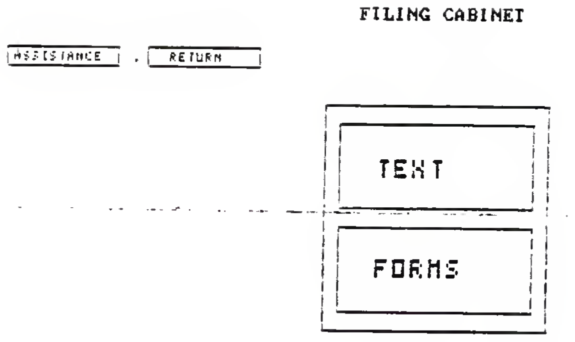
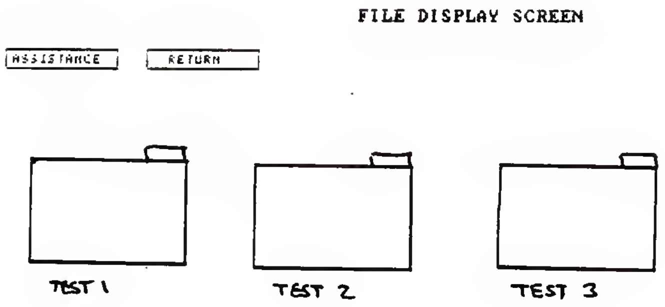


Figure 4.2 Manufacturing Organizational Chart



(a)



(b)

Figure 4.3 Example of Filing Cabinet and Display Screens

upper left hand corner of the screen is the assistance button whose properties have already been discussed. The second button is the return button that allows the user to return to the file selection state. The next button is the attribute assignment button. Clicking , with the left mouse button, brings up an attribute menu selection screen. Figures 4.4 and 4.5 gives an example of these two screens. Selection of the first four attributes in Figure 4.5 is achieved by positioning the mouse over the desired choice and clicking the left mouse button. The response time attributes is assigned by first clicking the left mouse button over the choice and then typing in the desired response time. To exit the attribute selection mode, the return button is triggered. For all of the filing cabinets screens there exists a return button that allows the user to revert back to the previous screen. Also, an assistance button is available for each screen. The final button, return to top, was developed out of the prototype design of this project. The observation was made that once a user had what they wanted in their hands they wanted to distribute it as soon as possible. The button allows one to return immediately to the personnel

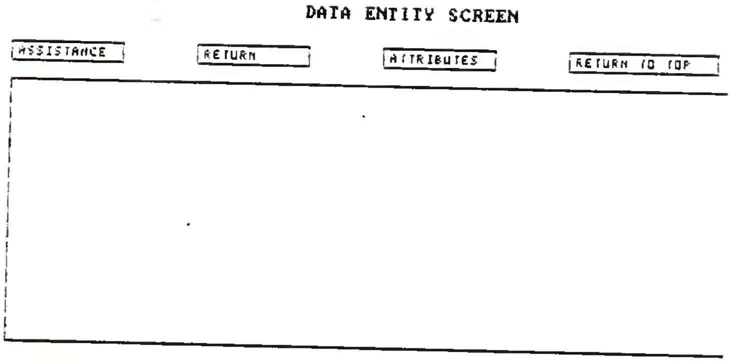


Figure 4.4 Example of the Data Entity Screen

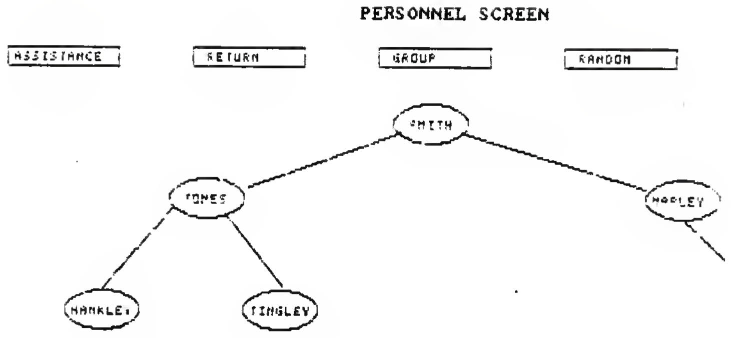


Figure 4.5 Example of a Routing Model

charts in order select personnel for distribution.

4.5 Routing

The routing subsystem is composed of personnel organization charts that are associated with each department in the company. The mouse cursor is placed over the desired department and the right mouse button is clicked. This action brings up the associated personnel chart.

Once the user has called up this screen, there are three general operations that can occur.

1) Return
2) Selection
3) Send

The return button is used to switch the screen back to the company organization chart. The user can then select various individuals from within different departments for a routing list. The return facility is represented by the area labeled return located next to the assistance button in the upper lefthand side of the screen.

The selection operation determines the mode of choice. There are two selection modes available: the group button and the random button. The group choice allows the user to broadcast a data entity over an organizational chart. The selection mode of operation works the following way. The user first positions the mouse cursor over the group button and clicks the left button. Next, the cursor is placed over the desired person entity within the organizational chart and the left mouse button is clicked. This causes the individual selected and all individuals below him to become targets for a data entity. For example, if Jones were selected then Jones, Hankley, and Tingley would all become recipients of the same data entity (see Figure 4.5).

The random selection mode allows the user to select a variety of individuals, outside of the framework of the organizational charts. The user first "pushes" the random button using the left mouse button and proceeds to chose the individuals wanted on the routing list by placing the cursor over each individual and clicking the left mouse button.

The send button is used to cause the transmittal of a data entity to targeted individuals. One of the attributes associated with the data entity is response time. If a response time is specified and an individual exceeds this response time then a ´no response´ message is displayed.

There is one final instance of a screen that needs to be discussed. This is the mail received screen. A very limited simulation is performed in order to demonstrate what can be done with received mail. An area in the upper right hand corner of the screen will become visible and inform the user that a message has been received. Clicking the cursor on this icon will cause the actual message to be displayed. See Figure 4.6. This screen has the usual assistance and return buttons. In addition, there is a save button for filing the message away and discard button that deletes the message.

Figure 4.6 Message Screen

Figure 4.6 Message Screen

CHAPTER 5 - IMPLEMENTATION

5.1 INTRODUCTION

The purpose of this chapter is to present both the hardware and software considerations decided upon for this project. Attention is paid not only to the tools used but also the major data structures, the software design, and the different modules. Finally, a list of implementation assumptions is given in order to define the bounds of the project.

5.2 HARDWARE/SOFTWARE TOOLS

The interface is implemented on a Epson Equity 1 Personal Computer equipped with a Genius Mouse operating under the MS-DOS operating system. The programming tools used are the Borland Turbo Pascal compiler V3.0 and the Turbo Graphics Toolbox V3.0. The word processor is the PFS: Professional Write package.

5.3 MOUSE INTERFACE

Communications wtih the ouse is established through the use of one of the two types of interrupts available on a DOS based personnel computer. This interrupt, known as a BIOS interrupt, allows one to obtain the following information.

1. Check to see if mouse is installed ano oetermine the number of buttons
2. Determine position
3. Set position of cursor
4. Set x and y mouse boundaries
5. Display/erase mouse cursor
6. Set screen movement ratios

Pascal provides a vehicle for generation interrupts therefore
allowing access to the aoove functions.

5.4 DATA STRUCTURES

There are two major data structures associated with this project. An array of records know collectively as fileinfo and a two dimensional array of records know as people. Individual occurrences of fileinfo correspond to the data entities discussed earlier and contain the following information. There are boolean fields that indicated a record is in use, a text or file form, ano indicators of which attributes a data entity might have been selected (i.e.- decision,comment,aoministrative action and general information). An integer field is available for response time . A file name field of 14 characters is used to point to a file on disk that contains the actual text or form message. Finally, an array of characters is used to indicate which person entities that a given fileinfo record is targeted for. Hence, each fileinfo record is compose of pointers to a text or form file located on disk, a list of attributes associated with the file on disk and a list of individuals for which the file on disk is destined.

The second major data structure is the two dimensional array of record know as people and correspond to these person entities as described in Section 1.3. Each occurrence of a people record has the following characteristics. A field indicating the name of the individual record and an array of integers used as pointers to other occurrences of people. This sort of linkage conveys the hierarchical structure of the personnel charts. An additional data structure, know as registers, is composed of 10 integers and is used along with a variety of MS-DOS interrupts in order to communicate with the mouse. This is discussed in detail in the previous section.

5.5 SOFTWARE DESIGN

This project operates in a mode of constantly checking the state of the mouse buttons, the locations of the cursor and the current state of the model. Exiting this sampling mode is achieved by the use of the left or right mouse button over an appropriate area within a given state. The sampling mode is exited only during times of state change, file accesses to the disks, dynamic graphic structures being implemented and certain keyboard dialogues. A state handler is implemented with a Pascal case statement that makes subsequent calls to the appropriate handler. The charter of the handler is to apply on interpretation to the coordinates being passed down from the mouse. This interpretation determines whether new states are created and whether a file related action is to occur.

There are seven state handlers and five file related procedures. These seven state handlers are listed below.

1. Top Level Handler (Fig1loc)
2. Organizational Handler (Fig2loc)
3. Cabinet Handler (Fig4loc)
4. File display Handler (Fig5loc)
5. Data Entity Handler (Fig6loc)
6. Attribute Handler

7. Personnel Handler

The Top Level handler determines whether the manufacturing or research organization has been chosen and changes the sate accordingly. The organizational handler determines which department is being selected and whether tne sate snould be change to either the personnel charts or the filing cabinets. The cabinet handler opens one of the two drawers and call a procedure to dynamically draw the file folders. The file display handler determines which files is to be read in and makes the actual cail to disk in order to retrieve the desired file. The data entity handler loaos the selecteo file onto the clipboard ano is displayed on the screen. The attribute handler allows tne user to assign attributes to the file that is resident on the clipboard. The personnel chart handler determines who was selected for distribution and what department they are associated with.

The four file related handlers are listed below.

1. Drawstuff
2. Getfile
3. Selectperson
4. Grouperson

The drawstuff procedure, called from the cabinet hanoler, dynamically draws all the data entities that are marked as "in-use" within the selected drawer. The Getfile procedure use a pointer (the actual file name) associated with a given data entity in order open and read in a file located on the disk. The Selectpersons procedure transfers data from the person entity to the data entity while in random selection mode. The alternative way to passing this type of information is by a call to Grouperson. This selection mode causes the individual selected to become the root node and is stored in the data entity along with all subsequent nodes descending from the root node. An editor is available for limited one line editing of the textual/form field of the data entity.

5.6 Statistics

The implementation was divided into two Pascal programs : one for intialization of various data stuructures and screens and the second for the actual interface. The total number of lines of code for both programs was 2150.

[Lod83] Loddsing,K.N., "Iconic Interfacing" IEEE CG&A , (Mar/Apr 1983), 11-20

[McC86] McCrum,R.,Cran,W.,and MacNesil,R. "The Story of The English Language " Viking Penguin Inc., New York, New York , 1986

[Mel&Mor85] Melamed, B. and Morris, R.J.T. "Visual Simulation: The Performance Analysis Workstation", Computer , (Aug. 1985), 87-94

[Shn82] Shneiderman,B. "The Future of Interactive Systems and the Emergence of Direct Manipulation ", Behaviour and Information Technology 1,3 (1982) 237-256

[Shn83] Shneiderman,B. "Direct Manipulation: A Step Beyond Programming Languages" , Computer, (Aug 1983), 57-69

[Sib86] Sibert,J.L.,Hurley,W.D. and Bleser,T.W. "An Object-Orientated User Interface Management System " ACM (1986) 259-268

[Smi82] Smith,D.C.,Irby,C.,Kimball,R.,ano Verplank,B. "Designing the Star User Interface" Byte (April 1982) 242-282

[Sta&Mac87] Stanton, R.B. and MacKenzie, H.G. "A Graphics Orientated Deductive Planning System" , The Australian Computer Journal 19,2 (May 1987), 76-83

[Ste87a] Stefik,M., Bobrow,D.G., Kahn,K., Lanning,S., and Suchman, L. "Beyond the Chalkborad: Computer Support for Collaboration ano Problem Solving in Meetings", Communications of the ACM 30,1 (January 1987), 32-47

[Ste87b] Stefik, M., Bobrow, D.G., Foster, G., Lanning, S., and Tatar, D. "WYSIWIS Revised: Early Experiences in Multiuser Interfaces" ACM Trans. Office Informations Systems 5,2 (April 1987), 147-167

[Rae85] Raeder, G. "A scrvey of Current Grapnical Programming Technique" Computer (August 1985), 11-25

[Whi85] Whiteside,J.,Jones,S.,Levy,P.S.,and Wixon,D. "User Performance with Command,Menu,and Iconic Interfaces " CHI 1985 Proceedings (SIGCHI Bulletin) (April 1985) 185-191

[Zie86] Ziegler,J.E.,Hoppe,H.U.,and Fahnrich,K.P. "Learning and Transfer for Text and Graphics with a Direct Manipulation Interface" CHI 1986 Proceedings (SIGCHI Bulletin) (April 1986)72-77

[Zaj68] Zajonc,R.B. "Attitudinal effect of mere exposure" Journal of Personality and Social Psychology Monograph Supplement (1968) , 1-27

[Zlo82] Zloof,M.M. "Office-by-Example: A business language that unifies data and word processing and electronic mail " IBM Systems Journal 21,3 (1982) 272-304

APPENDIX A

SOURCE CODE

```
{$C-,U-}
program initchem;

{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}
{$I polygon.hgh}

type
   fnam = string[14];

type
   peopleinfo=record
      pname:string[14];
      links:array [1..5] of integer;
   end;

type
   registers=record
       ax,bx,cx,dx,bp,si,di,ds,es,flags:integer;
   end;

var  aspectloc , rad : real ;
     j,l,x,y,i,jj,worldnum,savx,savy,tmpx,tmpy : integer;
     orgx,orgy,savscale,scale_x,xx,k :integer ;
     prtflag : boolean;
     chars : char;
     regs :   registers;
     aa,bb,cc : plotarray;

     names : array [1..36] of fnam;
     peoplefile : file of peopleinfo;
     people : array[1..6] of array[1..10] of peopleinfo;


{  **  turn Mouse cursor off ** }
procedure turnoff;
begin
   regs.ax:=2;
   intr($33,regs);
end;

{  **  turn Mouse cursor on   **   }
procedure turnon ;
begin
   regs.ax := 1;
   intr($33,regs);
```

```
end;


procedure init;
   begin

      InitGraphic;
      names[1]:='SMITH'  ; names[2]
      :='JONES';names[3]:='MARLY';
      names[4]:='HANKLEY' ; names[5]
      := TINGLEY';names[6]:='BASIL';
      names[7]:='AUTREY' ; names[8]
      :='BRITTON';names[9]:='BROWNER';
      names[10]:='CAHAN' ; names[11]
      :='CHEE';names[12]:='CIPRIANI';
      names[13]:='CRAIG' ; names[14]
      := DAILEY';names[15]:='FREDERICK';
      names[16]:='GRECO' ; names[17]
      :='HARDACRE';names[18]:='HARMON';
      names[19]:='HIRTLER' ; names[20]
      :='HOANG';names[21]:='HORNEY';
      names[22]:='HUCHRO' ; names[23]
      :='INGRAM';names[24]:='INKLEY';
      names[25]:='KLIEN' ; names[26]
      :='MORGAN';names[27]:='NEMR';
      names[28]:='OTTE' ; names[29]
      :='PEELER';names[30]:='RANFT';
      names[31]:='SAAD' ; names[32]
      :='SCHUETZ';names[33]:='SHETTER';
      names[34]:='STEIN' ; names[35]
      :='STRAWN';names[36]:='STREET';


      aa[1,1] := 295; aa[1,2] :=60;
      aa[2,1] := 285; aa[2,2] := 75;
      aa[3,1] := 295; aa[3,2] := 90;
      aa[4,1] := 355; aa[4,2] := 90;
      aa[5,1] := 365; aa[5,2] := 75;
      aa[6,1] := 355; aa[6,2] := 60;
      aa[7,1] := 295; aa[7,2] := 60;

      bb[1,1] := 295; bb[1,2] := 130;
      bb[2,1] := 285; bb[2,2] := 155;
      bb[3,1] := 345; bb[3,2] := 155;
      bb[4,1] := 355; bb[4,2] := 130;
      bb[5,1] := 295; bb[5,2] := 130;

      cc[1,1] := 100; cc[1,2] := 60;
      cc[2,1] := 90 ; cc[2,2] := 85;
```

```
cc[3,1] := 140; cc[3,2] := 85;
cc[4,1] := 150; cc[4,2] := 60;
cc[5,1] := 100; cc[5,2] := 60;

prtflag := false;

drawborder;
gotoxy(20,2);
write('TOP LEVEL INFORMATION DISTRIBUTION SYSTEM');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(130,30,205,40,false);
drawtext(145,34,1,'QUIT');
drawpolygon(aa,1,7,0,1,0);
drawtext(303,70,1,'FILE');
drawtext(303,78,1,'SERVER');
drawpolygon(bb,1,5,0,1,0);
drawtext(298,142,1,'PRINTER');
drawsquare(100,100,180,110,false);
drawtext(101,103,1,'MANUFACTURING');
drawsquare(470,100,550,110,false);
drawtext(480,103,1,'RESEARCH');
drawline(325,90,325,130);
drawline(285,75,140,100);
drawline(365,75,510,100);
savescreen('fig1');
prtflag := true;
if prtflag then hardcopy(false,6);

clearscreen;
drawborder  ;
gotoxy(25,2);
write('MANUFACTURING ORGANIZATIONAL PLAN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(130,30,205,40,false);
drawtext(145,34,1,'RETURN');
drawpolygon(cc,1,5,0,1,0);
drawtext(102,70,1,'PRINTER');
drawpolygon(aa,1,7,0,1,0);
drawtext(303,70,1,'FILE');
drawtext(303,78,1,'SERVER');
drawsquare(100,130,180,160,false);
drawtext(105,135,1,'INVENTORY');
drawtext(105,145,1,'CONTROL');
drawtext(105,155,1,'DEPT.');
drawsquare(286,130,366,160,false);
drawtext(290,140,1,'DISTRIBUTION');
drawtext(290,150,1,'DEPT.');
```

```
drawsquare(460,130,540,160,false);
drawtext(463,140,1,'PRODUCTION');
drawtext(463,150,1,'DEPT.');
drawline(285,75,145,72);
drawline(290,80,140,130);
drawline(323,90,323,130);
drawline(361,80,500,130);
savescreen('fig2');
if prtflag then hardcopy(false,6);
prtflag:=false;

clearscreen;
drawborder  ;
gotoxy(30,2);
write('RESEARCH ORGANIZATIONAL PLAN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(130,30,205,40,false);
drawtext(145,34,1,'RETURN');
drawpolygon(cc,1,5,0,1,0);
drawtext(102,70,1,'PRINTER');
drawpolygon(aa,1,7,0,1,0);
drawtext(303,70,1,'FILE');
drawtext(303,78,1,'SERVER');
drawsquare(100,130,180,160,false);
drawtext(105,135,1,'QUALITY');
drawtext(105,145,1,'ASSURANCE');
drawtext(105,155,1,'DEPT.');
drawsquare(286,130,366,160,false);
drawtext(290,140,1,'WASTE WATER');
drawtext(290,150,1,'DEPT.');
drawsquare(460,130,540,160,false);
drawtext(465,140,1,'BIO-ASSAY');
drawtext(465,150,1,'DEPT.');
drawline(285,75,145,72);
drawline(290,80,140,130);
drawline(323,90,323,130);
drawline(361,80,500,130);
savescreen('fig3');
if prtflag then  hardcopy(false,6);

clearscreen;
drawborder  ;
gotoxy(35,2);
write('FILING CABINET');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(130,30,205,40,false);
drawtext(145,34,1,'RETURN');
```

```
drawsquare(250,60,400,180,false);
drawsquare(260,70,390,115,false);
drawtext(285,92,2,'TEXT');
drawsquare(260,125,390,170,false);
drawtext(285,145,2,'FORMS');
savescreen('fig4');
prtflag:=true;
if prtflag then hardcopy(false,6);

clearscreen;
drawborder  ;
gotoxy(35,2);
write('FILE DISPLAY SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(130,30,205,40,false);
drawtext(145,34,1,'RETURN');
drawsquare(255,30,330,40,false);
drawtext(285,34,1,'NEW');
savescreen('fig5');
if prtflag then hardcopy(false,6);

clearscreen;
drawborder  ;
gotoxy(30,2);
write('DATA ENTITY SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(165,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(290,34,1,'ATTRIBUTES');
drawsquare(410,30,510,40,false);
drawtext(415,34,1,'RETURN TO TOP');
drawsquare(35,50,605,180,false);
drawsquare(560,30,635,40,false);
drawtext(565,34,1,'EDIT');
drawsquare(220,185,295,195,false);
drawtext(230,190,1,'SAVE');
drawsquare(345,185,420,195,false);
drawtext(355,190,1,'DELETE');
savescreen('fig6');
if prtflag then hardcopy(false,6);
prtflag:=false;

clearscreen;
drawborder  ;
gotoxy(30,2);
write('ATTRIBUTES');
```

```
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(165,34,1,'RETURN');
drawsquare(200,50,400,150,false);
drawtext(215,55,1,'DECISION');
drawline(200,65,400,65);
drawtext(215,75,1,'COMMENT');
drawline(200,85,400,85);
drawtext(215,95,1,'ADMINISTRATIVE ACTION');
drawline(200,105,400,105);
drawtext(215,115,1,'GENERAL INFORMATION');
drawline(200,125,400,125);
drawtext(215,135,1,'RESPONSE TIME');
savescreen('fig6a');

clearscreen;
drawborder  ;
gotoxy(35,2);
write('PERSONNEL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(295,34,1,'GROUP');
drawsquare(410,30,485,40,true);
setcolorblack;
drawtext(420,34,1,'RANDOM');
setcolorwhite;
drawsquare(535,30,610,40,false);
drawtext(545,34,1,'SEND');
setaspect(1);
drawcircledirect(320,70,25,false);
drawtext(306,70,1,'SMITH');

drawcircledirect(170,110,25,false);
drawtext(155,110,1,'JONES');

drawcircledirect(490,110,25,false);
drawtext(470,110,1,'MARLEY');

drawcircledirect(100,170,25,false);
drawtext(80,170,1,'HANKLEY');

drawcircledirect(220,170,25,false);
drawtext(200,170,1,'TINGLEY');

drawcircledirect(560,170,25,false);
```

```
drawtext(540,170,1,'BASIL');

drawline(300,75,195,105); {smith to Jones}
drawline(340,75,466,105); {Smith to Marly}
drawline(147,116,99,160); {Jones to Hankley}
drawline(182,119,222,158); {Jones to Tingley}
drawline(490,121,545,162); {Marley to Basil}
prtflag:=true;
if prtflag then
  hardcopy(false,6);
prtflag:=false;
savescreen('fig7a');


clearscreen;
drawborder  ;
gotoxy(35,2);
write('PERSONNEL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(295,34,1,'GROUP');
drawsquare(410,30,485,40,TRUE);
setcolorblack;
drawtext(420,34,1,'RANDOM');
setcolorwhite;
drawsquare(535,30,610,40,false);
drawtext(545,34,1,'SEND');

setaspect(1);
drawcircledirect(320,70,25,false);
drawtext(306,70,1,'AUTREY');

drawcircledirect(170,110,25,false);
drawtext(155,110,1,'BRITTON');

drawcircledirect(490,110,25,false);
drawtext(470,110,1,'BROWNER');

drawcircledirect(100,170,25,false);
drawtext(80,170,1,'CAHAN');

drawcircledirect(220,170,25,false);
drawtext(200,170,1,'CHEE');

drawcircledirect(560,170,25,false);
drawtext(540,170,1,'CIPRIANI');
```

```
drawline(300,75,195,105); {smith to Jones}
drawline(340,75,466,105); {Smith to Marly}
drawline(147,116,99,160); {Jones to Hankley}
drawline(182,119,222,158); {Jones to Tingley}
drawline(490,121,545,162); {Marley to Basil}
if prtflag then  hardcopy(false,6);
savescreen('fig7b');

clearscreen;
drawborder  ;
gotoxy(35,2);
write('PERSONNEL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(295,34,1,'GROUP');
drawsquare(410,30,485,40,true);
setcolorblack;
drawtext(420,34,1,'RANDOM');
setcolorwhite;
drawsquare(535,30,610,40,false);
drawtext(545,34,1,'SEND');
setaspect(1);
drawcircledirect(320,70,25,false);
drawtext(306,70,1,'CRAIG');

drawcircledirect(170,110,25,false);
drawtext(155,110,1,'DAILEY');

drawcircledirect(490,110,25,false);
drawtext(470,110,1,'FREDERICK');

drawcircledirect(100,170,25,false);
drawtext(80,170,1,'GRECO');

drawcircledirect(220,170,25,false);
drawtext(200,170,1,'HARDACRE');

drawcircledirect(560,170,25,false);
drawtext(540,170,1,'HARMON');

drawline(300,75,195,105); {smith to Jones}
drawline(340,75,466,105); {Smith to Marly}
drawline(147,116,99,160); {Jones to Hankley}
drawline(182,119,222,158); {Jones to Tingley}
drawline(490,121,545,162); {Marley to Basil}
```

```
if prtflag then  hardcopy(false,6);
savescreen('fig7c');

clearscreen;
drawborder  ;
gotoxy(35,2);
write('PERSONNEL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(295,34,1,'GROUP');
drawsquare(410,30,485,40,true);
setcolorblack;
drawtext(420,34,1,'RANDOM');
setcolorwhite;
drawsquare(535,30,610,40,false);
drawtext(545,34,1,'SEND');
setaspect(1);
drawcircledirect(320,70,25,false);
drawtext(306,70,1,'HIRTLER');

drawcircledirect(170,110,25,false);
drawtext(155,110,1,'HOANG');

drawcircledirect(490,110,25,false);
drawtext(470,110,1,'HORNEY');

drawcircledirect(100,170,25,false);
drawtext(80,170,1,'HUCHRO');

drawcircledirect(220,170,25,false);
drawtext(200,170,1,'INGRAM');

drawcircledirect(560,170,25,false);
drawtext(540,170,1,'INKLEY');

drawline(300,75,195,105); {smith to Jones}
drawline(340,75,466,105); {Smith to Marly}
drawline(147,116,99,160); {Jones to Hankley}
drawline(182,119,222,158); {Jones to Tingley}
drawline(490,121,545,162); {Marley to Basil}
if prtflag then  hardcopy(false,6);
savescreen('fig7d');

clearscreen;
drawborder  ;
gotoxy(35,2);
```

```
write('PERSONNEL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(295,34,1,'GROUP');
drawsquare(410,30,485,40,true);
setcolorblack;
drawtext(420,34,1,'RANDOM');
setcolorwhite;
drawsquare(535,30,610,40,false);
drawtext(545,34,1,'SEND');
setaspect(1);
drawcircledirect(320,70,25,false);
drawtext(306,70,1,'KLIEN');

drawcircledirect(170,110,25,false);
drawtext(155,110,1,'MORGAN');

drawcircledirect(490,110,25,false);
drawtext(470,110,1,'NEMR');

drawcircledirect(100,170,25,false);
drawtext(80,170,1,'OTTE');

drawcircledirect(220,170,25,false);
drawtext(200,170,1,'PEELER');

drawcircledirect(560,170,25,false);
drawtext(540,170,1,'RANFT');

drawline(300,75,195,105); {smith to Jones}
drawline(340,75,466,105); {Smith to Marly}
drawline(147,116,99,160); {Jones to Hankley}
drawline(182,119,222,158); {Jones to Tingley}
drawline(490,121,545,162); {Marley to Basil}
if prtflag then  hardcopy(false,6);
savescreen('fig7e');

clearscreen;
drawborder  ;
gotoxy(35,2);
write('PERSONNEL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
```

```
drawtext(295,34,1,'GROUP');
drawsquare(410,30,485,40,true);
setcolorblack;
drawtext(420,34,1,'RANDOM');
setcolorwhite;
drawsquare(535,30,610,40,false);
drawtext(545,34,1,'SEND');

setaspect(1);
drawcircledirect(320,70,25,false);
drawtext(306,70,1,'SAAD');

drawcircledirect(170,110,25,false);
drawtext(155,110,1,'SCHUETZ');

drawcircledirect(490,110,25,false);
drawtext(470,110,1,'SHETTER');

drawcircledirect(100,170,25,false);
drawtext(80,170,1,'STEIN');

drawcircledirect(220,170,25,false);
drawtext(200,170,1,'STRAWN');

drawcircledirect(560,170,25,false);
drawtext(540,170,1,'STREET');

drawline(300,75,195,105); {smith to Jones}
drawline(340,75,466,105); {Smith to Marly}
drawline(147,116,99,160); {Jones to Hankley}
drawline(182,119,222,158); {Jones to Tingley}
drawline(490,121,545,162); {Marley to Basil}
if prtflag then  hardcopy(false,6);
savescreen('fig7f');

clearscreen;
drawborder  ;
gotoxy(35,2);
write('INCOMING MAIL SCREEN');
drawsquare(35,30,110,40,false);
drawtext(40,34,1,'ASSISTANCE');
drawsquare(160,30,235,40,false);
drawtext(170,34,1,'RETURN');
drawsquare(285,30,360,40,false);
drawtext(295,34,1,'SAVE');
drawsquare(410,30,485,40,false);
drawtext(420,34,1,'DISCARD');
savescreen('fig8');
```

```
     clearscreen;
     drawborder  ;
     gotoxy(35,2);
     write('NEW MESSAGE SCREEN');
     drawsquare(35,30,110,40,false);
     drawtext(40,34,1,'ASSISTANCE');
     drawsquare(160,30,235,40,false);
     drawtext(170,34,1,'RETURN');
     drawsquare(285,30,360,40,false);
     drawtext(295,34,1,'SAVE');
     hardcopy(false,6);
     savescreen('fig9');


end; { procedure init  }

procedure writelink;
begin
assign(peoplefile,'people.dat');
rewrite(peoplefile);
l := 1;
for k:=1 to 6 do
  begin
    for j:=1 to 6 do
      begin
        with people[j,k] do
          begin
            pname := names[l];
            l:=l+1;
            case j of
            1:
              begin
                links[1]:=-1;
                links[2]:=2;
                links[3]:=3;
                links[4]:=-9;
              end;
            2:
              begin
                links[1]:=1;
                links[2]:=4;
                links[3]:=5;
                links[4]:=-9;
              end;
            3:
              begin
                links[1]:=1;
```

```
                  links[2]:=6;
                  links[3]:=-9;
                  links[4]:=-9;
                end;
              4:
                begin
                  links[1]:=2;
                  links[2]:=-9;
                  links[3]:=-9;
                  links[4]:=-9;
                end;
              5:
                begin
                  links[1]:=2;
                  links[2]:=-9;
                  links[3]:=-9;
                  links[4]:=-9;
                end;
              6:
                begin
                  links[1]:=3;
                  links[2]:=-9;
                  links[3]:=-9;
                  links[4]:=-9;
                end;
            end;  {case}
          end;  {with}
          write(peoplefile,people[j,k]);
       end;  {for j}
    end; { for k}
  close(peoplefile);
end; {writelink}


                { M A I N }

begin   {   main    }
init;
writelink;

{ ** start processing ** }

chars := #32; {clear out chars}
repeat
if keypressed then
  begin
    read(kbd,chars);
    if chars = #112 then hardcopy(false,1);
  end;
```

```
until chars = #113;
leavegraphic;
end.
```

```
{$C-,U-}
program cnem;

{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}


type
   registers=record
        ax,bx,cx,dx,bp,si,di,ds,es,flags:integer;
   end;

type
   holder = string[1];
   line = string[50];
   fnam = string[14];

type
   attributes = record
      inuse,tex_form,decision,comment,admin,geninfo :
boolean;
      response : integer;
      filname : string[14];
      route : array[1..50] of fnam;
    end;

type
    peopleinfo=record
      pname:string[14];
      links:array [1..5] of integer;
    end;


var
v,j,l,offset,oldx,oldy,x,y,i,jj,drawer,savx,savy,pptr,tmpy
: integer;
deptchoice,personchoice,prevfig,fignum,choice,scale_x,xx,k
:integer ;
x1,y1,x2,y2,pct,oldect,ect,mov,ct,topfig,start,stop:intege
r;
     drawfile,pickflag,textflag,formflag,skipflag,quitflag
: boolean;
asstflag,mailflag,control,editdone,perflag,pick,randflag :
boolean;
     printflag,found,mesgdone:boolean;
     chars : char;
     chars2: array [1..2] of char;
     regs :   registers;
     saveval : holder   ;
     fileinfo : array [1..70] of attributes;
     buf : array [1..10] of line;
     filevar: text;

     peoplefile: file of peopleinfo;
     people : array[1..6] of array[1..10] of peopleinfo;
     parray: array[1..10] of integer;

procedure fig5loc; forward;
procedure fig6loc; forward;
procedure fig9loc; forward;

{  **  turn Mouse cursor off ** }
procedure turnoff;
begin
   regs.ax:=2;
   intr($33,regs);
end;

{  **  turn Mouse cursor on   **  }

procedure turnon ;
begin
   regs.ax := 1;
   intr($33,regs);
end;

procedure ast1;
begin
  turnoff;
  copyscreen;
  setcolorblack;
  drawsquare(35,40,250,130,true);
  setcolorwhite;
  drawsquare(35,40,250,130,false);
  drawtext(110,45,1,'ASSISTANCE');
  drawline(35,50,250,50);
  drawline(35,120,250,120);
  drawtext(120,125,1,'CLOSE WINDOW');
  turnon;
  asstflag:=true;
end;

procedure ast2;
begin
  repeat
```

```
     regs.ax := 3;
     intr($33,regs);
     { move variables into generic x and y coordinates }
     x := regs.cx;
     y := regs.dx;

     if ((x>34) and (x<251)) and ((y>120) and (y<131)) then
       begin
         if regs.bx=1 then
           begin
             turnoff;
             swapscreen;
             asstflag:=false;
             x:=0;
             turnon;
           end;
       end;
    until (not asstflag);

end;

procedure noresp(x1,y1,x2,y2:integer);
begin
  drawsquare(x1,y1,x2,y2,true);
  setcolorblack;
  drawtext(x1+6,((y1+y2) div 2),1,'NO RESPONSE');
  setcolorwhite;
end;

procedure checkmark(x,y:integer);
begin
  drawline(x,y,x+3,y+5);
  drawline(x+3,y+5,x+10,y-8);
  pick:=true;
end;


procedure edit;
begin
  turnoff;
  drawsquare(560,30,635,40,true);
  setcolorblack;
  drawtext(565,34,1,'EDIT');
  setcolorwhite;
  turnon;
  chars:='  ';
  ect:=1;
  control := false;
  gotoxy(14+ect,16);
```

```
write('^');
repeat
if keypressed then
   begin
     read(kbd,chars);

     if chars=#27 then
       begin
         read(kbd,chars);
         control := true;

         case chars of

         'K': {left arrow}
              if ect <> 1 then ect:=ect-1;

         'M': {right arrow}
              if ect < length(buf[1]) then ect :=ect+1;

         'S': {delete}
              begin
                delete(buf[1],ect,1);
                gotoxy(15,15);
                write('
                ');
                gotoxy(15,15);
                write(buf[1]);
              end;
           'D':
               chars:=#1;
         end;  {case}

         if (chars='M') or (chars='K') then
           begin
             gotoxy(15,16);
             write('
              );
             gotoxy(14+ect,16);
             write('^');
           end;
       end;


     if (not control) then
       begin
         if ((chars>#64) and (chars<#91)) or
            ((chars>#96) and (chars<#123)) or
            ((chars>#47) and (chars<#58))  or (chars=#32)
           then
```

```
             begin
               if (ect < 50) then
                 begin
                   insert(chars,buf[1],ect);
                   ect:=ect+1;
                   gotoxy(15,15);
                   write(buf[1]);
                   gotoxy(15,16);
                   write('
                     );
                   gotoxy(14+ect,16);
                   write('^');
                   gotoxy(65,15);
                   write('            ');
                   delay(10);
                 end
               else
                 begin
                   chars:=' ';
                   ect:=49;
                 end;
          end;
        end;
    control := false;
    end;

    regs.ax := 3;
    intr($33,regs);
    { move variables into generic x and y coordinates }
    x := regs.cx;
    y := regs.dx;

  if ((y>184) and (y<196)) and ((x>219) and (x<296)) and
    (regs.bx=1) then
    begin
      chars:=#1;
      ect:=1;
    end;
  until chars=#1;
end;

procedure drawstuff;
begin
  if textflag then
    drawer := 0
  else
    drawer := 5;
  for k:= 1 to 5 do
    begin
```

```
        with fileinfo[k+offset+drawer] do
           begin
             if inuse then
               begin
                 turnoff;
                 drawsquare((20+(k-1)*100)+55,65,(100+(k-
                 1)*100),70,false);
                 drawsquare((20+(k-1)*100),70,(100+(k-
                 1)*100),100,false);
                 drawtext((20+(k-1)*100)+20,115,1,filname);
                 turnon;
               end;
        end; {with}
     end; {for}

end; {drawstuff}

procedure savemesg;
begin
turnoff;
copyscreen;
setcolorblack;
drawsquare(200,30,400,150,true);
setcolorwhite;
drawsquare(200,30,400,150,false);
drawtext(210,40,1,'ENTER FILE NAME AND');
drawtext(210,50,1,'SAVE BY DEPRESSING F10');
gotoxy(30,16);
write('^');
delete(buf[6],1,80);
ect:=1;

repeat
  if keypressed then
    begin
      read(kbd,chars);

      if chars=#27 then
        begin
          read(kbd,chars);
          control := true;

          case chars of

          'K': {left arrow}
              if ect <> 1 then ect:=ect-1;

          'M': {right arrow}
              if ect < length(buf[6]) then ect :=ect+1;
          'S': {delete}
              begin
                delete(buf[6],ect,1);
                gotoxy(30,15);
                write('
                 ');
                gotoxy(30,15);
                write(buf[6]);
              end;
            'D':
              chars:=#1;
          end;  {case}

          if (chars='M') or (chars='K') then
            begin
              gotoxy(30,16);
              write('
               );
              gotoxy(29+ect,16);
              write('^');
            end;
        end;


      if (not control) then
        begin
          if ((chars>#64) and (chars<#91)) or
             ((chars>#96) and (chars<#123)) or
             ((chars>#47) and (chars<#58))  or (chars=#32)
             then
             begin
               if (chars <> #81) and (ect < 10) then
                 begin
                   insert(chars,buf[6],ect);
                   ect:=ect+1;
                   gotoxy(30,15);
                   write(buf[6]);
                   gotoxy(30,16);
                   write('                    ');
                   gotoxy(29+ect,16);
                   write('^');
                 end;
             end;
       end;
       control:=false;
 end

until chars = #1;
```

```
if fignum=9 then
  assign(filevar, mail ')
else
  assign(filevar,buf[6]);
rewrite(filevar);
if fignum=9 then
  begin

    write(filevar,buf[3])
  end
else
  write(filevar,buf[5]);
close(filevar);
found:=false;
if formflag then
  k:=6
else
  k:=1;

repeat
  if (not fileinfo[k].inuse) then
   begin
    with fileinfo[k] do
      begin
        inuse := true;
        if k<6 then
          tex_form := true
        else
          tex_form := false;
        decision := false;
        comment := false;
        admin := false;
        geninfo := false;
        response := -1;
        found := true;
        filname:=buf[6];
     end;
   end
   else
     begin
       k:=k+1;
       if (k > 10) or ((k>5) and textflag) then
         begin
           gotoxy(15,20);
           write('FILE CABINET IS FULL');
           delay(300);
         end;
     end;
until found;
```

```
swapscreen;
eno;

procedure getfile;
begin
  with fileinfo[choice] do
    begin
      assign(filevar,filname);
      reset(filevar);
      readln(filevar,buf[1]);
      gotoxy(15,15);
      write(buf[1]);
      close(filevar);
    end;  { with }
end;   {  getfile  }

procedure readlink;
begin
assign(peoplefile,'people.dat');
reset(peoplefile);
l := 1;
for k:=1 to 6 do
  begin
    for j:=1 to 6 do
        read(peoplefile,people[j,k]);
end; { for k}
close(peoplefile);

end;

procedure selectperson;
begin
  fileinfo[choice].route[pptr] :=
people[personchoice,deptchoice].pname;
  pptr:=pptr+1;
eno;

procedure grouperson;
begin
  start:=personchoice;
  stop := personchoice;

  fileinfo[choice].route[pptr]:=people[start,deptchoice].p
  name;
  pptr:=pptr+1;

  case personchoice of
    1:
      begin
```

```
          start:=2;
          stop:=6;
        end;
     2:
        begin
          start:=4;
          stop:= 5;
        end;
     3:
        begin
          start:=6;
          stop:=6;
        end;
   end;  {case}

  for k:=start to stop do
    begin
 fileinfo[choice].route[pptr]:=people[k,deptchoice].pname;
        pptr:=pptr+1;

        parray[pct]:=k;
        pct:=pct+1;

        case k of
         1:
            checkmark(275,70);
         2:
            checkmark(125,110);
         3:
            checkmark(445,110);
         4:
            checkmark(55,170);
         5:
            checkmark(175,170);
         6:
            checkmark(515,170);
        end; {case}

      end; {for}
end; {grouperson}

procedure fig1loc;
begin
   if mailflag then
     begin
        turnoff;
        drawsquare(440,60,515,80,true);
        setcolorblack;
        drawtext(450,70,1,'NEW MAIL');
    setcolorwhite;
    turnon;
  end;

if (y > 29) and (y < 40)
  then
    begin
      if (x > 34) and (x < 111)        { assistance }
        then
          begin
             ast1;
             drawtext(45,55,1,'PLACING THE CURSOR ON
             EITHER OF');
             drawtext(45,65,1,'ORGANIZATIONS
             (MANUFACTORING OR');
             drawtext(45,75,1,'RESEARCH) AND CLICKING
             THE LEFT');
             drawtext(45,85,1,'MOUSE BUTTON WILL ALLOW
             ACCESS TO');
             drawtext(45,95,1,'THE INDIVIDUAL
             DEPARTMENTS');
             ast2;
          end;

      if (x > 129) and (x < 206)       { quit }
        then
          begin
            quitflag := true;
          end;
   end;

if ((y>59) and (y<81)) and ((x>439) and (x<516)) and
  mailflag then
  begin
    fignum:=9;
    clearscreen;
    turnoff;
    loadscreen('fig8');
    turnon;
    assign(filevar,fileinfo[70].filname);
    reset(filevar);
    readln(filevar,buf[3]);
    gotoxy(15,15);
    write(buf[3]);
    close(filevar);
  end;

if (y > 99) and (y < 111)
  then
```

```
      begin
        if (x > 99) and (x < 181)
           then
             begin
               fignum := 2;
               topfig:=fignum;
               turnoff;
               clearscreen;
               loadscreen('fig2');
               turnon;
             end;
        if (x > 469) and (x <  551)
          then
            begin
              fignum := 3;
              topfig:=fignum;
              turnoff;
              clearscreen;
              loadscreen('fig3');
              turnon;
            end;
     end;
end;

procedure fig2loc;
begin
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (x < 111)       { assistance }
          then
            begin
               ast1;
               drawtext(45,55,1,'PLACING THE CURSOR ON ANY
               OF THE ');
               drawtext(45,65,1,'DEPARTMENTS AND CLICKING
               THE LEFT');
               drawtext(45,75,1,'MOUSE BUTTON WILL CAUSE
               THE ');
               drawtext(45,85,1,'ASSOCIATED DATABASE TO
               THE SCREEN.');
               drawtext(45,95,1,'CLICKING THE RIGHT MOUSE
               BUTTON');
               drawtext(45,105,1,'CAUSES THE PERSONNEL
               CHART TO');
               drawtext(45,115,1,'APPEAR. ');
               ast2;

            end;
      if (x > 129) and (x < 206)      { return }
        then
          begin
            fignum := 1;
            turnoff;
            clearscreen;
            loadscreen('fig1');
            turnon;
          end;
    end;
if (y > 129) and ( y < 161)
  then
    begin
      if ((x > 99) and (x < 181))  or
         ((x > 285) and (x < 367)) or
         ((x > 459) and (x < 541))
         then
           begin

             if ((x > 99) and (x < 181) ) then
               begin
                 if fignum =2 then
                   offset := 0
                 else
                   offset := 30;
               end;
             if ((x > 285) and (x < 367) ) then
               begin
                 if fignum = 2 then
                   offset := 10
                 else
                   offset := 40;
               end;
             if ((x > 459) and (x < 541 ) ) then
               begin
                 if fignum = 2 then
                   offset := 20
                 else
                   offset := 50;
               end;
               prevfig := fignum;
               turnoff;
               clearscreen;
               if (not perflag) then
                 begin
                   fignum := 4;
                   loadscreen('fig4');
                 end
               else
        begin
          case offset of
            0:
              loadscreen('fig7a');
            10:
              loadscreen('fig7b');
            20:
              loadscreen('fig7c');
            30:
              loadscreen('fig7d');
            40:
              loadscreen('fig7e');
            50:
              loadscreen('fig7f');
          end; {case}
if randflag then
  begin
    turnoff;
    setcolorblack;
    drawsquare(285,30,360,40,true);
    setcolorwhite;
    drawsquare(285,30,360,40,false);
    drawtext(295,34,1,'GROUP');
    drawsquare(410,30,485,40,true);
    setcolorblack;
    drawtext(420,34,1,'RANDOM');
    setcolorwhite;
    turnon;
  end
else
  begin
    turnoff;
    drawsquare(285,30,360,40,true);
    setcolorblack;
    drawtext(295,34,1,'GROUP');
    drawsquare(410,30,485,40,true);
    setcolorwhite;
    drawsquare(410,30,485,40,false);
    drawtext(420,34,1,'RANDOM');
    turnon;
 end;

          pct:=1;
          for jj:=1 to 6 do
            parray[jj]:=-99;
          fignum := 7;
          perflag := false;
        end;
    turnon;
```

```
                end;
          end;
end;



procedure fig3loc;
begin
end;

procedure fig4loc;
begin
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (x < 111)          { assistance }
          then
            begin
               ast1;
               drawtext(45,55,1,'PLACING THE CURSOR ON
               EITHER ');
               drawtext(45,65,1,'OF FILE DRAWERS AND
               CLICKING');
               drawtext(45,75,1,'THE MOUSE BUTTON WILL
               CAUSE  ');
               drawtext(45,85,1, DRAWER TO OPEN );
               ast2;

            end;
        if (x > 129) and (x < 206)        { return }
          then
            begin
              fignum := prevfig;
              turnoff;
              clearscreen;
              str(fignum,saveval);
              loadscreen('fig'+saveval);
              turnon;
            end;
     end;

   if (x > 259) and (x < 391)
     then
       begin
         if (y > 69) and (y < 116)
           then
             begin
               fignum := 5;
               textflag := true;
              formflag := false;
              drawfile :=true;
            end;
        if (y > 124) and (y < 171)
          then
            begin
              fignum := 5;
              textflag := false;
              formflag := true;
              drawfile := true;
           end;
        if drawfile then
          begin
              turnoff;
              clearscreen;
              loadscreen('fig5');
              drawstuff;
              turnon;
          end;
      end;
end;


procedure fig5loc;

begin
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (x < 111)       { assistance }
          then
            begin
              ast1;
              drawtext(45,55,1,'PLACING THE CURSOR ON THE
              NEW');
              drawtext(45,65,1,'BUTTON AND CLICKING THE
              LEFT');
              drawtext(45,75,1,'MOUSE BUTTON WILL ALLOW
              THE');
              drawtext(45,85,1,'USER TO CREATE A NEW
              FILE');
              ast2;

            end;
        if (x > 129) and (x < 206)      { return }
          then
            begin
             fignum := 4;
             turnoff;
```

```
              clearscreen;
              loadscreen('fig4');
              turnon;
            end;

        if (x > 254) and (x < 331)      { new }
          then
            begin
              fignum := 10;
              turnoff;
              clearscreen;
              loadscreen('fig9');
              fig9loc;
            end;

     end;


   if (y > 69) and (y<101) then
     begin
       for k:=1 to 5 do
         begin
           if ((x >(19+(k-1)*100)) and (x < (101+(k-
1)*100)) ) then
             begin
               choice := k+offset+drawer;
               turnoff;
               clearscreen;
               loadscreen('fig6');
               pptr := 1;
               getfile;
               turnon;
               fignum := 6;
             end;
         end; {for}
     end;

end;


procedure fig6loc;

begin
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (x < 111)       { assistance }
          then
            begin
          ast1;
          ast2;
        end;
    if (x > 159) and (x < 236)       { return }
      then
        begin
          turnoff;
          clearscreen;
          loadscreen('fig5');
          drawstuff;
          fignum := 5;
          turnon ;
        end;
    if (x > 284) and (x < 361)         { attributes }
      then
        begin
          turnoff;
          clearscreen;
          loadscreen('fig6a');
          fignum:=8;
          turnon;
        end;
    if (x > 409) and (x < 511) then    {return to top }
      begin
        fignum:=topfig;
        turnoff;
        clearscreen;
        str(fignum,saveval);
        loadscreen('fig'+saveval);
        turnon;
      end;
    if (x > 559) and (x < 631) then    {edit}
      begin
       getfile;
       edit;
     end;
 end;


if ((y>184) and (y<196)) and ((x>219) and (x<296)) then
 begin
   with fileinfo[choice] do
     begin
        assign(filevar,filname);
        rewrite(filevar);
        write(filevar,buf[1]);
        close(filevar);
     end;
end;
```

```
if ((y>184) and (y<196)) and ((x>344) and (x<421)) then
  begin
    with fileinfo[choice] do
      begin
        inuse:=false;
        decision := false;
        comment := false;
        admin := false;
        geninfo := false;
        response := -1;
        assign(filevar,filname);
        erase(filevar);
        close(filevar);
        delete(filname,1,14);
      end;
    gotoxy(15,15);
    write('
    ');
 end;


end;

procedure fig6aloc
;
begin
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (x < 111)       { assistance }
          then
            begin
              ast1;
              ast2;
            end;
        if (x > 159) and (x < 236)      { return }
          then
            begin
              turnoff;
              clearscreen;
              loadscreen('fig6');
              pptr := 1;
              getfile;
              turnon;
              fignum := 6;
            end;
      end;
```

```
if (x > 199) and (x < 400)
  then
    begin
      if (y > 49) and (y < 65)  { decision }
        then
          begin
            turnoff;
            drawsquare(200,50,400,65,true);
            setcolorblack;
            drawtext(215,55,1,'DECISION');
            setcolorwhite;
            fileinfo[choice].decision := true;
            turnon;
          end;
      if ( y > 64) and (y < 85)  { comment }
          then
            begin
            turnoff;
            drawsquare(200,65,400,85,true);
            setcolorblack;
            drawtext(215,75,1,'COMMENT');
            setcolorwhite;
            fileinfo[choice].comment := true;
            turnon;
            end;
      if (y > 84) and (y < 105)    { admin. action}
        then
          begin
            turnoff;
            drawsquare(200,85,400,105,true);
            setcolorblack;
            drawtext(215,95,1,'ADMINISTRATIVE ACTION');
            setcolorwhite;
            fileinfo[choice].admin := true;
            turnon;
          end;
      if ( y > 104) and (y < 125)   { general info }
          then
            begin
            turnoff;
            drawsquare(200,105,400,125,true);
            setcolorblack;
            drawtext(215,115,1,'GENERAL INFORMATION');
            setcolorwhite;
            fileinfo[choice].geninfo := true;
            turnon;
          end;
      if ( y > 124) and (y < 149)   { response time }
          then
```

```
              begin
                mov:=315;
                ct:=1;
                repeat
                  if keypressed then
                    begin
                      read(kbd,chars);
                      if chars<>#113 then
                      drawtext(mov,135,1,chars);
                      ct:=ct+1;
                      mov:=mov+5;
                      if ct = 5 then chars := #113
                    end;
                until chars = #113
              end;
          end;
end;{fig6aloc}


procedure fig7loc;
begin
  delay(250);
  gotoxy(15,15);
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (x < 111)       { assistance }
          then
            begin
              ast1;
              ast2;
            end;

        if (x > 159) and (x < 236)      { return }
          then
            begin
              turnoff;
              clearscreen;
              str(prevfig,saveval);
              loadscreen('fig'+saveval);
              turnon ;
              fignum := prevfig;
            end;

        if (x > 284) and (x < 361) then  { group }
          randflag:=false;

        if (x > 409) and (x < 486) then   {random}
          randflag:=true;
        if ((x > 284) and (x < 361)) or ((x>409) and
          (x<486)) then
          begin
            if randflag then
              begin
                turnoff;
                setcolorblack;
                drawsquare(285,30,360,40,true);
                setcolorwhite;
                drawsquare(285,30,360,40,false);
                drawtext(295,34,1,'GROUP');
                drawsquare(410,30,485,40,true);
                setcolorblack;
                drawtext(420,34,1,'RANDOM');
                setcolorwhite;
                turnon;
              end
            else
              begin
                turnoff;
                drawsquare(285,30,360,40,true);
                setcolorblack;
                drawtext(295,34,1,'GROUP');
                drawsquare(410,30,485,40,true);
                setcolorwhite;
                drawsquare(410,30,485,40,false);
                drawtext(420,34,1,'RANDOM');
                turnon;
             end;
         end;

        if (x > 534) and (x < 611)     { send }
          then
            begin
            if pick then
            begin
              x:=0;
              y:=0;
              pick:=false;
              fileinfo[choice].route[pptr] :='zzz';
              if printflag then
                begin
                  writeln(lst,'file name =
',fileinfo[choice].filname);
                  k:=1;
                  while (fileinfo[choice].route[k] <>
                  'zzz') do
                    begin
```

```
                      writeln(lst,'target =
',fileinfo[choice].route[k]);
                      k:=k+1;
                    end;
                  with fileinfo[choice] do
                    begin
                      writeln(lst,'decision =  ',decision);
                      writeln(lst,'comment  =  ',comment);
                      writeln(lst,'admin    =  ',admin);
                      writeln(lst,'gen info =  ',geninfo);
                     writeln(lst);
                  end;
                  writeln(lst);writeln(lst);
                end;

{             writeln(lst,'the people chosen ');}
{             for k:=1 to 6 do             }
{                 writeln(lst,parray[k]);}
 {}


            pptr:=1;
            turnoff;
            clearscreen;
            str(prevfig,saveval);
            case offset of
              0:
                  loadscreen('fig7a');
             10:
                  loadscreen('fig7b');
             20:
                  loadscreen('fig7c');
             30:
                  loadscreen('fig7d');
             40:
                  loadscreen('fig7e');
             50:
                  loadscreen('fig7f');
            end; {case}


            if randflag then
              begin
                turnoff;
                setcolorblack;
                drawsquare(265,30,360,40,true);
                setcolorwhite;
                drawsquare(265,30,360,40,false);
```

```
      drawtext(295,34,1,'GROUP');
      drawsquare(410,30,485,40,true);
      setcolorblack;
      drawtext(420,34,1,'RANDOM');
      setcolorwhite;
      turnon;
    end
  else
    begin
      turnoff;
      drawsquare(285,30,360,40,true);
      setcolorblack;
      drawtext(295,34,1,'GROUP');
      drawsquare(410,30,485,40,true);
      setcolorwhite;
      drawsquare(410,30,485,40,false);
      drawtext(420,34,1,'RANDOM');
      turnon;
   end;

  jj:=1;
  while ((parray[jj] <> -99) and (jj<7)) do
    jj:=jj+1;
  jj:=random(jj);
  jj:=parray[jj];

      turnon;
      delay(1600);
      case jj of
        1:
           noresp
           (210,65,285,75);
        2:
           noresp(60,105,135,115);
        3:
           noresp(525,105,600,115);
        4:
           noresp(15,141,90,151);
        5:
           noresp(255,165,330,175);
        6:
           noresp(450,165,525,175);
      end;  {case}

with fileinfo[choice] do
    begin
      route[1]:='nogo';
      decision :=false;
      comment := false;
```

```
                  admin := false;
                  geninfo := false;
                  response:=-1;
                end;


                       pct:=1;
                       for jj:=1 to 6 do
                         parray[jj]:=-99;



              fignum := 7;
              end
          else
             begin
               gotoxy(28,16);
               write('PLEASE  SELECT A ROUTE FIRST ');
               turnon;
               fignum:=7;
             end;
  end;{send}
end;     {top line choices}


pick:=false;

  if ( ((y > 44) and (y < 96)) and ((x > 294)  and (x  <
    346)) )
    then
      begin
        personchoice :=1;
        pick:=true;
      end;

  if (y > 84 ) and ( y < 136) then
    begin
      if (x> 144) and ( x < 196) then
        begin
          personchoice := 2;
          pick:=true;
        end;
      if ( x > 464) and ( x < 516) then
        begin
          personchoice := 3;
          pick:=true;
        end;
    end;
```

```
if (y > 144) and ( y < 196) then
  begin
    if (x > 74) and (x < 126) then
      begin
        personchoice := 4;
        pick:=true;
      end;
    if (x > 194) and (x < 246) then
      begin
        personchoice := 5;
        pick:=true;
      end;
    if (x > 534) and (x < 585) then
      begin
        personchoice := 6;
        pick:=true;
      end;
  end;

  if pick then
    begin

     case offset of
       0:
            deptchoice:=1;
      10:
            deptchoice:=2;
      20:
            deptchoice := 3;
      30:
            deptchoice := 4;
      40:
            deptchoice := 5;
      50:
            deptchoice := 6;
    end; { case }

    parray[pct]:=personchoice;
    pct:=pct+1;

    case personchoice of
     1:
        checkmark(275,70);
     2:
        checkmark(125,110);
     3:
        checkmark(445,110);
     4:
        checkmark(55,170);
      5:
         checkmark(175,170);
      6:
         checkmark(515,170);
     end; {case}


    if randflag then
      selectperson
    else
      grouperson;

end;

end;   { fig7loc }


procedure fig8loc;
begin
  if (y > 29) and (y < 40)
    then
      begin
        if (x > 34) and (y < 111)        { assistance }
          then
            begin
            end;

        if (x > 159) and (x < 236)       { return }
          then
            begin
              turnoff;
              clearscreen;
              loadscreen('fig1');
              turnon ;
              fignum := 1;
            end;

        if (x > 284) and (x < 361) then   { save }
          begin
            savemesg;
            turnon;
          end;
        if (x > 409) and (x < 486) then    {discard}
          begin
            gotoxy(15,15);
            write('
            ');
          end;
  end;
  mailflag:=false;
end;

procedure fig9loc;
begin
turnon;
mesgdone:=false;
ect:=1;
gotoxy(15,16);
write('^');
delete(buf[5],1,80);
repeat
    regs.ax := 3;
    intr($33,regs);
    { move variables into generic x and y coordinates }
    x := regs.cx;
    y := regs.dx;

  if (y > 29) and (y < 40)  and (regs.bx=1)
    then
      begin
        if (x > 34) and (x < 111)       { assistance }
          then
            begin
              ast1;
              ast2;
            end;

        if (x > 159) and (x < 236)      { return }
          then
            begin
              turnoff;
              clearscreen;
              loadscreen('fig5');
              drawstuff;
              turnon ;
              fignum := 5;
              mesgdone:=true;
            end;

       if (x>284) and (x<361) then       { save }
         begin
           savemesg;
           turnon;
         end;
     end;

  if keypressed then
```

```
begin
  read(kbd,chars);

  if chars=#27 then
    begin
      read(kbd,chars);
      control := true;

      case chars of

      'K': {left arrow}
          if ect <> 1 then ect:=ect-1;

      'M': {right arrow}
          if ect < length(buf[5]) then ect :=ect+1;

      'S': {delete}
          begin
            delete(buf[5],ect,1);
            gotoxy(15,15);
            write('
            ');
            gotoxy(15,15);
            write(buf[5]);
          end;
      end;  {case}

      if (chars='M') or (chars='K') then
        begin
          gotoxy(15,16);
          write('
          ');
          gotoxy(14+ect,16);
          write('^');
        end;
    end;


  if (not control) then
    begin
      if ((chars>#64) and (chars<#91)) or
         ((chars>#96) and (chars<#123)) or
         ((chars>#47) and (chars<#58))  or (chars=#32)
        then
        begin
          if (ect < 50) then
            begin
              insert(chars,buf[5],ect);
              ect:=ect+1;
```

```
                      gotoxy(15,15);
                      write(buf[5]);
                      gotoxy(15,16);
                      write('
                      ');
                      gotoxy(14+ect,16);
                      write('^');
                    end;
               end;
           end;
       control := false;
   end

  until mesgdone;
  end;

  procedure checkloc;
  begin
    case fignum of
      1:
            fig1loc;
      2,3:
            fig2loc;
      4:
            fig4loc;
      5:
            fig5loc;
      6:
            fig6loc;
      7:
            fig7loc;
      8:
            fig6aloc;
      9:
            fig8loc;
     10:
           fig9loc;
    end; {case}
  end;



  procedure init;
     begin
       Initgraphic;
       defineworld(1,0,0,640,200);
       selectworld(1);
       readlink;
       fignum := 1;
      perflag := false;
      quitflag := false;
      editdone := false;
      control := false;
      mailflag := false;
      printflag:=false;

      for jj:=1 to 6 do
         parray[jj]:=-99;

      fileinfo[70].filname := 'mail';
      fileinfo[70].inuse :=true;

      for k :=1 to 8 do
       begin
         with fileinfo[k] do
           begin
             inuse := true;
             if k<6 then
               tex_form := true
             else
               tex_form := false;
             decision := false;
             comment := false;
             admin := false;
             geninfo := false;
             response := -1;
             route[1]:='nogo';
             if k < 6 then
               begin
                 str(k,saveval);
                 filname := 'TEST   + saveval;
               end
             else
               begin
                 str(k-5,saveval);
                 filname := 'FORM' + saveval;
               end;
         end ; {with}
      end;  {do}
    end; { procedure init  }


                 { M A I N }

begin   {   main    }
init;
```

```
loadscreen('fig1');

{  put a mouse cursor on the screen  }

regs.ax := 1;
intr($33,regs);
regs.ax:=4;

{ ** set Mouse to center of screen ** }

regs.cx := 350;
regs.dx := 100;
intr($33,regs);

{ ** start processing ** }

k:=0;
skipflag := false;
pptr :=1;
randflag:=true;
v:=0;
repeat


  { **  determine where the mouse is  and get button
status  ** }

      regs.ax := 3;
      intr($33,regs);
      { move variables into generic x and y coordinates }
      x := regs.cx;
      y := regs.dx;

{       gotoxy(5,5);          }
{       write('x =       ');   }
{       gotoxy(5,6);          }
{       write('y =       ');}
{       gotoxy(5,5);     }
{       write('x=',x);   }
{       gotoxy(5,6);    }
{       write('y=',y);}


{  **** The LEFT mouse button was chosen   ****      }

  if regs.bx=1 then
     begin
       checkloc ;
     end;        { LEFT   }
  {   MIDDLE Mouse button selected }

  if regs.bx = 4  then
       mailflag:=true;



  { **   RIGHT  Mouse button was chosen ** }

  if regs.bx=2 then
     begin
       perflag := true;
       checkloc;
     end;
  if regs.bx=3 then
    begin
      if printflag then
        printflag:=false
      else
        printflag:=true
    end;

  if (fignum = 1) and mailflag then
     begin
       drawsquare(440,60,515,80,true);
       setcolorblack;
       drawtext(450,70,1,'NEW MAIL');
       setcolorwhite;
       delay(100);
     end;


until quitflag;
leavegraphic;
end.
```

DESIGN AND DEMONSTRATION OF A DIRECT MANIPULATION INTERFACE

by

DAVID H. TINGLEY

B.S., Morris Harvey College, 1974
B.S., Florida Institute of Technology, 1982

AN ABSTRACT OF A REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTERS OF SCIENCE

Department of Computing and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

ABSTRACT

The purpose for this project is to demonstrate an interface to an office automation system that takes advantage of innate human visual processing abilities. This interface is cast in the mold of a direct manipulation file management system linked directly to a routing system. This system allows users to create and route various office correspondence for subsequent action. A simple model of a hypothetical chemical company, simulating a small portion of an office environment, is used to allow for the demonstration of the interface.