



# **Common User Interface Standard**

---

**Draft of April 14, 1993**

*BEST AVAILABLE COPY*

# U.S.A.I.D. Common User Interface Guidelines

## Table of Contents

Preface .....	i
Introduction .....	ii
1.0 Background .....	1
1.1 Principles of CUI .....	3
1.2 User Interface Guidelines .....	4
1.2.1 User Interface Criteria .....	5
1.2.1.1 Familiar User's Conceptual Model (Metaphor) .....	5
1.2.1.2 User-Driven Interface .....	5
1.2.1.3 Consistency .....	6
1.2.1.4 Modeless Operation .....	7
1.2.1.5 Transparent Interface .....	7
1.2.1.6 Simplicity and Forgiveness .....	7
1.2.1.7 Positive Feedback .....	8
1.2.1.8 Universal Commands .....	8
1.2.2 User Interactions .....	8
1.2.2.1 Process Sequence .....	9
1.2.2.2 Object and Action Selection .....	9
1.2.2.3 Selection Techniques .....	10
1.2.2.4 Selection Indicators and Emphasis .....	11
1.3 Benefits of Common User Interface .....	11
1.4 CUI Interface Models .....	13
1.4.1 Common User Interface Styles .....	14
2.0 Graphic Common User Interface .....	15
2.1 Basic Windows Concepts .....	15
2.1.1 Interacting with a Window .....	19
2.1.2 Operating on a Window .....	19
2.1.3 User Interface .....	20
2.1.4 Messages .....	21
2.1.5 Presentation Manager .....	22
2.2 Presentation Elements of Graphic User Interface .....	23
2.2.1 Basic Window Components .....	23

2.2.1.1	Title Bar	23
2.2.1.2	Window Border	23
2.2.1.3	Menu Bar	23
2.2.1.4	Work Area	24
2.2.1.5	Window Elements and Controls	24
2.2.1.6	Scroll Bars	24
2.2.2	Title Bar	24
2.2.2.1	System Menu Button	25
2.2.2.2	Window Title	25
2.2.2.3	Window Control Buttons	25
2.2.3	Window Border	25
2.2.4	Menu Bar and Pull-Downs	26
2.2.4.1	Operation of Menu Bars and Pull-Downs	27
2.2.5	Work Area	32
2.2.6	Window Elements and Controls	32
2.2.6.1	Field and Group Identifiers	34
2.2.6.2	Radio Button	36
2.2.6.3	Check Box	36
2.2.6.4	List Box	37
2.2.6.5	Value Set	37
2.2.6.6	Spin Button	38
2.2.6.7	Combination Box	39
2.2.6.8	Drop-Down Combination Box	39
2.2.6.9	Drop-Down List	39
2.2.6.10	Dialogue Box	40
2.2.6.11	Entry Fields	42
2.2.6.12	Pushbuttons	44
2.2.6.13	Logo Window	46
2.2.7	Scroll Bars	47
2.2.7.1	Interaction	49
2.3	Application Interaction Techniques	50
2.3.1	Combining Elements	50
2.3.2	Informing the Users	53
2.3.2.1	Feedback	53
2.3.2.2	Messages	55
2.3.2.3	Help	59
2.3.2.4	The Help Window	61
3.0	Text Presentation of Common User Interface	63
3.1	Presentation Elements of Text Interface	64
3.1.1	Window or Panel	64

# U.S.A.I.D. Common User Interface Guidelines

---

3.1.1.1	Window Title	65
3.1.1.2	Window Identifier	65
3.1.1.3	Window Area Separators	65
3.1.1.4	Work Area	66
3.1.1.5	Message Line	67
3.1.1.6	Command Area	67
3.1.1.7	Function Key Line	68
3.2	Window Title	68
3.3	Window Identifier	68
3.4	Window Area Separators	68
3.5	Work Area	69
3.5.1	Instructions	69
3.5.2	Headings	69
3.5.2.1	Column Headings	69
3.5.2.2	Group Headings	70
3.5.2.3	Field Prompts	70
3.5.3	Descriptive Text	71
3.5.4	Protected Text	71
3.5.5	Scrolling	71
3.5.6	Scrolling Actions	72
3.5.7	Cursor Independent Scrolling	72
3.5.8	Cursor-Dependent Scrolling	73
3.5.9	Scrolling Indicators	74
3.5.10	Scrolling Arrows	74
3.5.11	Textual Scrolling Information	75
3.5.12	Textual Scrolling Location Information	75
3.6	Pop-Ups	76
3.6.1	Pop-Up Positioning	77
3.6.2	Pop-Up Layout	77
3.6.3	Pop-Up Content	77
3.6.4	Pop-Up Use	77
3.7	Menu bar	78
3.7.1	Menu bar Layout	78
3.7.2	Menu bar Content	78
3.7.3	Selection of Menu bar Options	79
3.7.4	Menu bar Pull-Down Layout	79
3.7.5	Menu bar Pull-Down	80
3.7.6	Menu bar Emphasis	80
3.7.7	Users' Interaction with Menu bar	81
3.8	Message Line	83
3.8.1	Types of Messages	83
3.8.2	Message Layout and Content	83

# U.S.A.I.D. Common User Interface Guidelines

---

3.8.3	Message Pop-up	84
3.8.4	Message Removal	84
3.8.5	Audible Feedback	85
3.8.6	Guidelines for Creating Messages	85
3.9	Command Line	86
3.9.1	Command Line Layout	87
3.9.2	Command Line Interaction	87
3.9.2.1	The Enter Action	87
3.9.2.2	The Command Action	87
3.9.2.3	The Prompt Action	87
3.9.2.4	The Retrieve Action	88
3.9.3	Applications having a Command Line and a Menu bar	88
3.10	Function Key Line	88
3.10.1	Function Key Line Content	89
3.10.2	Function Key Line Action Definitions	90
3.11	U.S.A.I.D. and System Identifiers	92
Microsoft Windows		93
1.	Topic	93
2.	Summary	93
3.	User Considerations	94
3.1	Standardized User Interface	94
3.2	Context Switching	94
3.3	Clipboard (Data Sharing)	95
3.4	Multitasking	95
3.5	Memory Management	95
4.	Developer's Considerations	96
4.1	Standard User Interface	96
4.2	Device Independence	97
4.3	Dynamic Link Libraries	98
4.4	Queued Input	99
4.5	Dynamic Data Exchange	99
4.6	DOS Applications Compatibility	100
5.0	Summary	101
U.S.A.I.D. Function Key Assignments		103
Word Processing Functions		107
Glossary		111

# Preface

As Information Management practices mature in large corporations and government agencies such as U.S.A.I.D., inventories of computer applications grow and, unless care and foresight is applied, each program may require the user to learn a new set of commands and procedures in order to use it.

It is clear that U.S.A.I.D. can greatly decrease the time and cost of its staff learning to use new computer programs if all A.I.D. programs use the same set of commands and procedures (the user interface). Additionally, new applications can be developed faster and more economically by not 'reinventing the user interface' for each new program.

The recent overwhelming market success of Microsoft Windows and applications which use that interface suggest that A.I.D. should move toward a graphical user interface (GUI). A GUI saves the user from memorizing cryptic computer commands or searching for function keys. GUIs allow programmers to provide clearer instructions and help screens to the users.

The following standard recommends a GUI approach for all new applications, while allowing improved text-based approaches for existing applications. The standard emphasizes the values of consistency, simplicity, and user empowerment to make the computing experience more pleasant and productive.

This standard is likely to evolve as technology improves and A.I.D. business needs change, as always we welcome suggestions, questions, and comments at:

Office of the Director  
FA/IRM  
USAID  
Washington, D.C. 20523-1402

# Introduction

Planning for future software and hardware needs is critical to management, particularly in times of rapidly changing technology. Organizations have made large investments in developing applications, choosing appropriate hardware and training staff. A major challenge facing management today is the necessity to incorporate the appropriate technology into the enterprise, while avoiding constant disruption and keeping productivity at acceptable levels.

In order to address these concerns, major computer technology companies are embracing goals for the future that commit to standards. Two areas that affect management today are the standards being set for *interoperability* and *portability*. Interoperability is the goal of having different computer platforms work together to complete a task.

IBM was faced with an especially complicated problem of producing standards, due to the wide range of hardware that has been developed over the years and their commitment to support all current platforms while moving toward the goal of interoperability. Their response was System Application Architecture (SAA). Using the architecture rules of SAA, interoperability will be accomplished through the use of, *common user interface* (CUI), *common programming interface* (CPI), and *common communications interface* (CCI)

The UNIX leaders are still attempting to provide a single standards. The Open Systems Foundation (IBM, DEC, Hewlett-Packard and Santa Cruz Operation) have committed to developing a UNIX operating system based on IBM's AIX. SUN and AT&T joined together in 1987 to develop a UNIX that would include a graphic user interface called Open Look.

The goal of portability addresses the ability to produce and use software that operates on any (or at least most) computer platforms, and appears and acts the same way for the user. This allows management to make hardware decisions without having to rewrite large application systems or retrain all the users. The main objectives of portability are addressed in IBM's SAA:

- Provide a common user interface so that products may be developed or purchased and the user will have expectations of how they will operate.
- Provide a common applications programming interface (API) so that the application developer can write code, confident that it will always perform

## U.S.A.I.D. Common User Interface Guidelines

---

the same way as long as their API and service routines are supported on the hardware.

- Provide a common communications interface to enable the applications developer to place processing (functional, SQL, database access) on the proper platform, and allow system designers to change the location or platform without altering the applications program.

This standards manual addresses the requirements of a common user interface and provides guidelines to developers and programmers of applications for the United States Agency for International Development (U.S.A.I.D.). The purpose of these guidelines is to provide the basis for a common user interface and common user actions that will produce the benefits attributable to these standards.

There are two primary user interface standards available to microcomputer end-users. One is the Apple Corporation standard and is implemented in one form or another on all of its products. The other standard is the portion of IBM's SAA that deals with Common User Access (CUA). This standard is "open", which means that its rules are published and it is available to be used by any and all. Most Independent Software Vendors (ISV) have chosen to implement user interfaces that follow the CUA Standards. Microsoft's Windows, IBM's Presentation Manager and Open System Foundation's (OSF) MOTIF for X Windows have all chosen to be CUA compliant.

With the constant technology upgrade to cooperative processing, SQL servers and multimedia applications, the user interface standards will continue to change and evolve. The purpose of these guidelines is to provide specific guidance to developers in using interface elements and controls, and to provide rationale to develop new elements to take advantage of new technology. As with any standard, developers are free to add extensions and embellish their presentations, as long as they remain within the overall intent and structure of the CUA guidelines. The **U.S.A.I.D. Common User Interface (CUI) Guidelines** are based on IBM's CUA standards and contains extensions and rules appropriate to U.S.A.I.D. and its environment.



## 1.0 Background

In the early years of data processing, the user interface was almost irrelevant. The overriding concern was building computing systems that worked and could be run by competent professionals. Hardware capabilities were constrained, and design strategy called for getting the most out of scarce hardware resources. The users of these systems were engineers, mathematicians, and accountants, who were used to dealing with arcane symbols and abbreviations.

The emergence of user interfaces as a critical component of computer systems has occurred for five reasons. First, advances in hardware devices have provided technology for graphic displays, and manual input devices. These include the mouse, pointer and pen. Early software and display terminals were capable of only character displays. Also hardware costs have decreased in recent years. The computing power that can be put on a desktop now rivals mainframe power of fifteen years ago. Sophisticated displays are not only available, but mandatory for personal computer workstations. As programmable workstations become the norm instead of the exception, graphic presentations become required by more users.

Second, as Personal Computers (PCs) are put in front of more and more people who are neither programmers or engineers, it is important to provide clear, easy to understand interfaces. People have been exposed to easy to use interfaces, from APPLE's graphic user interface of the LISA and MacIntosh, to automatic teller machines, and expect to see them on their computers. They have also been exposed to failures in this area, such as video cassette recorders and microwave ovens. A software developer who cannot demonstrate all of the functionality of his product to potential users without lengthy instructions or classes stands little chance of having the system accepted and used.

Third, the software required to implement the user interface typically accounts for at least forty percent, and often more, of the applications code in a commercial software product (some experts argue that a window based, mouse driven, icon oriented, graphic user interface may need 75 percent of the applications' code to provide the user interface). Toolkits and reusable methodologies become critical productivity aids in software development. The interface is not only important in user acceptability, but also in accounting for development costs.

Fourth, user interfaces are important to systems developers, because standards are being defined for them. These standards come from many sources, but they are

converging on the same objectives. Some of the groups developing standards include:

- IBM's Systems Application Architecture / Common User Access
- APPLE Corporation's Desktop and Finder
- Open Systems Foundation's (OSF) X Windows
- Open Look's X Windows

Fifth, the cost of learning a new software package is substantial. During early use of a new package, users spend considerable time looking for guidance and avoiding mistakes. Even those who take the time to learn a new application may master only a few skills due to this fear. An interface that promotes ease of learning and provides a comfort level consistent with a forgiving interface is important to software use and user productivity.

Microsoft and Zenith Data Systems sponsored a study performed by Temple, Barker and Stone, Inc. This study was completed in the Spring of 1990, and strongly supports the hypothesis that a graphic user interface provides benefits over a character user interface. The results showed significant improvement in the following seven areas by both experienced and novice users:

- Accomplish tasks faster
- Complete tasks more accurately
- Have higher productivity
- Experience lower frustration
- Experience lower fatigue
- Felt more at ease with exploration and self-teaching
- Learned more capabilities of the applications

## 1.1 Principles of CUI

The first serious commercial application using human factors research to produce a user interface design was the Xerox Corporation's 8010 STAR Information System, introduced in April 1981. The Star user interface rigorously adhered to a small set of design principles. These principles made the system seem familiar and friendly, simplified the human-machine interaction, unified nearly two dozen functional areas of Star, and allowed user experience in one area to apply to another. Xerox devoted about 30 work years to design the Star user interface. During this process they learned the importance of formulating the fundamental concepts (the user's conceptual model) *before* software is written, rather than tacking on a user interface *afterward*.

As a part of this project it was determined that there are some concepts that are inherently difficult for people to deal with, and others that are easier. Experience during the STAR design led to the following classification:

**Easy**

Concrete  
Visible  
Copying  
Choosing  
Recognizing  
Editing  
Interactive

**Hard**

Abstract  
Invisible  
Creating  
Filling in  
Generating  
Programming  
Batch

The characteristics on the left should be incorporated into the user's conceptual model, and the ones on the right should be avoided as much as possible. Microsoft's Windows, IBM's Presentation Manager, OSF Motif, Apple's Lisa and MacIntosh, DEC Windows and others, have adopted this kind of conceptual consistency in a user interface.

The following main goals were pursued in designing the Star user interface:

- Use a familiar conceptual model (metaphor) for the user
- Provide a user driven interface
- Be consistent in using directions, cues and actions

- Use modeless operations whenever possible
- Allow seeing and pointing (transparent interface) rather than forcing a user to remember and type input
- Use simplicity and forgiveness
- Provide positive feedback
- Define as many universal commands as possible

---

## 1.2 User Interface Guidelines

A working definition of user interface is a set of standardized elements and interaction techniques between user and application, and application and user. In a common user interface (CUI) all applications appear the same to the user and follow the same conventions or rules (e.g., pressing F1 should obtain context sensitive help in all applications).

Implied in this definition of interface is a conceptual model of how the users perceive and understand these interactions. In order to be efficient and effective, these interactions must be as clear and as intuitively understandable as possible.

### Application Interaction with User

The application controls the device presentation language, which is the way the application communicates with the user. To the user, *the interface is the system*. The application controls how information is accessed, how computations are performed, and how computed information is delivered in an understandable and usable form to the user.

### User Interaction with Application

Users must be able to understand what information the application is requesting at various steps so they can respond appropriately. This response consists of established actions, such as key strokes, function key selection, or mouse movements. Avoid ambiguous responses or statements that do not make a user response clear (e.g. **Cannot Load *Routinename* - press any key to continue**) may not give the user enough information about what to do next. If they continue, will

their work be lost, or does this just slow processing down or make certain functions unavailable?).

### **Conceptual Model**

Based on their experiences with previous office tasks or computer applications, and even other machines such as calculators or typewriters, users have expectations of what a computer interface is, what it does, and how it works. Some expectations come from the users' experiences with the interface itself. A good interface provides users with the means to develop the expectations they should have for the interface and applies these expectations consistently.

## **1.2.1 User Interface Criteria**

Based on the research done at Xerox's Palo Alto Research Center (PARC), the following main goals are provided to help system designers provide a friendly user interface.

### **1.2.1.1 Familiar User's Conceptual Model (Metaphor)**

A metaphor is a figure of speech used to suggest a resemblance. When a user is confronted with metaphors that are familiar and real-world based, he can transfer previous knowledge of his environment to the application interface. (Be careful when choosing a metaphor to make sure it meets the expectations users have because of their real-world experiences.) Often, an application design is based on a single metaphor that corresponds to the processing of paper in the non-computer world. One of the more common metaphors provided for the user is a desktop. The desktop contains icons that represent items used in an office, such as file folders, paper, trash cans, file and cabinets. Other common metaphors are calculators, telephone directories and card files.

### **1.2.1.2 User-Driven Interface**

If possible, application designers should design the application so that it allows users to apply their real-world knowledge of the paper process to the application interface. Good application design can then support the users' environment and goals. As an important part of the application design, application developers should perform a task analysis to learn what users want to do and how they want to do it. Some industry experts recommend having interface experts on the design team, and producing rapid prototyping of the user interface. Literally within hours after serious technical discussions between users and designers, rough sketches of the proposed user interface should be available for critique.

### 1.2.1.3 Consistency

Making the user interface consistent is one way to develop and reinforce the users' conceptual model of applications. Applications should be consistent with their hardware and software environments and throughout the applications themselves. Consistency throughout an application is supported by:

- Common presentation: what is seen by the user
- Common interaction: how a user interacts with the application screens
- Common process sequence: how a user communicates with the computer
- Common actions: how similar actions are implemented in the same way.

**Common Presentation:** Users become familiar with interface components when the visual appearance of the components is consistent and, whenever possible, when the location of the components is consistent. For example, the window title is a user interface element that is consistent in both appearance and in location. Entry fields, however, are consistent in appearance but not in location; they may appear anywhere in the work area.

**Common Interaction:** After users can recognize interface components, they can interact with them. When the application consistently supports interaction techniques associated with each component, users become familiar and comfortable with these techniques.

**Common Process Sequence: U.S.A.I.D. Common User Interface (CUI) Guidelines** support two process sequences that correspond to the two design approaches used in application programs: *object oriented* and *action oriented*. When the application consistently supports one process sequence, users become familiar with the way to interact with the application. They also learn how the application responds. For example, if the application consistently supports the object oriented sequence, users know that they must first select an object (e.g., file). They also know that a selected object is indicated, but that no action is taken until they request an action. After users learn this way of expecting the computer to respond, their conceptual model would not be supported if they selected an object and the application immediately performed an action.

**Common Actions: U.S.A.I.D. CUI Guidelines** define actions that have specific meanings. Common actions provide a language between users and the computer so users can understand the meaning and result of actions. **U.S.A.I.D. CUI**

**Guidelines** define common actions and terminology to assist you in providing consistency. For example, when users press the F1 key, they are telling the computer that they want to get help for the specific area of the application they are working on.

#### **1.2.1.4 Modeless Operation**

Users are in a mode whenever they must cancel or complete what they are doing before they can do something else or when the same action has different results in different situations. By providing modeless operation, users are free to execute other aspects of an application when they may need to retrieve information. Modes force users to focus on the way an application works, instead of on the task they want to complete.

It is not always possible or desirable to design a modeless application. An example of a mode is a Warning message that needs to be acted on before the user can safely proceed with the application. Whenever users are in a mode, the application should make it obvious by providing good visual cues. The method for ending the mode (e.g., Cancel or data entry) should be easy to identify from the situation.

#### **1.2.1.5 Transparent Interface**

The user interface should be as natural and intuitive as possible so users can anticipate what to do next by applying their previous knowledge of performing tasks without a computer. An application should reflect a real-world model of the user activities and tasks necessary to reach desired objectives. One way to provide an intuitive user interface is to use appropriate metaphors. A spreadsheet uses the metaphor of a lined accounting pad. Those users who are familiar with this metaphor easily accept the notion of putting a column of numbers on the spreadsheet and adding and cross footing them. The software hides all the complexity of the summing and recursive arithmetic inherent in spreadsheets from the user.

#### **1.2.1.6 Simplicity and Forgiveness**

The interface should be highly visual so users can see, through the use of pull-downs, pop-ups, or selection lists, how to proceed. Both the presentation of interface components and the user interaction with the components should be visual. Try to present information to the user, rather than make them recall or guess what the system is requesting.

User actions should be easily reversed. Users should be able to explore without fear of causing an irreversible mistake. They should be able to back up or undo

their previous action. Actions that may cause the unexpected loss or modification to users' data always require a confirmation and a positive response, not just typing the Enter key (*default must be to not change or delete data*). Users feel more comfortable with an application and are able to learn quicker and become more productive when their mistakes do not cause serious or irreversible results.

### 1.2.1.7 Positive Feedback

Provide feedback for users whenever possible. Users should never perform an action without receiving visual feedback, audible feedback, or both. For example, color, emphasis, and other presentation techniques show users which choices they can select, when a choice has been selected, and when a requested action has been completed. Audible feedback is particularly useful for error or warning conditions to make sure the user is aware of the condition and need for action.

### 1.2.1.8 Universal Commands

Consistency of similar actions is important in reinforcing the users' conceptual model. For example, a user may be executing applications that are quite different, but they both involve creating and editing objects such as files or reports. Both applications must provide users with the ability to save and retrieve objects.

**U.S.A.I.D. CUI Guidelines** defines common actions that system designers may apply to different kinds of objects. These actions are provided through the menu bar and the function key area. All common actions, however, may not be applied to all types of objects, so **U.S.A.I.D. CUI Guidelines** provides direction for deciding when to use them. Menu bar common actions are implemented through standard menu bar pull-downs and dialogue pop-ups.

Application actions may be assigned to function keys. These function keys serve as accelerators. By pressing these keys, the associated action is executed immediately, without typing commands or accessing the menu bar and pull-downs.

Typing commands in the command area may be used as a fast path for experienced users. They might be able to issue an application command faster by typing it than by selecting an object and then specifying the action (command name) and its parameters through the menu bar, pull-downs, and associated pop-ups.

## 1.2.2 User Interactions

Interaction is the means through which users interact with the user interface components. Interaction is fundamental to establishing and reinforcing the users' conceptual model.

### 1.2.2.1 Process Sequence

The process sequence for a graphic or text based user interface is *object oriented*. This means that objects are selected and a list of allowable actions is available that will operate on the object. An object is anything that users can manipulate as a single unit (e.g., file, paragraph, record, field, word or character). An action describes any way that users can change, manipulate, create, or delete an object or the properties of an object. When the application consistently supports an object oriented process sequence, it reinforces the users' conceptual model of the user interface.

The object oriented process sequence has some advantages over the action oriented sequence. Some of the benefits of the object oriented process sequence are the following:

It provides a basis for users to explore the application through context-sensitive actions. Users can first select an object and then browse the actions of an application to see which ones apply to the selected object. Users perceive that they are in control because they can see which actions are valid prior to attempting to perform them.

It allows users to perform a series of actions on a selected object rather than to select repeatedly the object for each desired action. For example, in a word processing application, users usually select some text (an object) and then apply to it a number of styles (actions), such as bold, italic, and underlined. Users find an interface inconvenient and repetitive if they have to select an object many times to request all the actions they want to perform on that object. For example, some text editors require the user to select Input mode or Edit mode. If Input mode was selected, then the user cannot edit any of the text. In order to make corrections, the user must exit Input mode and enter Edit mode.

It reduces the complexity of the user interface and simplifies its overall architecture by reducing the levels of action hierarchies needed for the application.

### 1.2.2.2 Object and Action Selection

The distinction between objects and actions affects both the users' interaction with the user interface components as well as the presentation of information. There is a significant difference between the selection of an object and the selection of an action.

**Object Selection** is the act of identifying objects. It is important that users should always see a visual indication that selection occurred.

Selecting an object should never imply an action, affect the object, or commit users to an action. De-selecting an object should be as simple as selecting the object. Users should be able to cancel a wrong selection or change their minds without penalty.

An object may be selected by typing a selection character into the choice entry field immediately preceding it.

**Action Selection** when applied to an object should cause something to occur immediately and should also provide a visual cue. For example, if users want to save a document (perform the Save action on the object, a document), they move the cursor to the menu bar choice *File* and press the Enter key to display the *File* pull-down. Users select the *Save* choice and press the Enter key. The document is saved immediately and a visual indication is given that the action occurred.

If users attempt to select an action before they select an object, the application presents a message that prompts users to select an object and then an action.

The difference between selecting objects and actions is that selecting an object causes nothing to happen except visual confirmation of the selection, whereas selecting an action causes the action to occur immediately and also provides visual confirmation that the action occurred.

Applications may allow an object to remain selected even after an action has been performed on it. If users normally perform another action, or a series of actions, on that object, then it should remain selected (e.g. *Save* operation allows object to remain selected, *Cut* or *Copy* do not). If the object no longer exists in its original form and location as a result of the action, the object is no longer shown as selected.

### 1.2.2.3 Selection Techniques

There are two types of selection: *explicit* and *implicit*.

In *explicit selection*, users select a choice by typing a selection character (e.g., a number of the choice) in the choice entry field for the desired choice.

In *implicit selection*, users simply move the cursor to the desired choice. The choice is automatically selected and when the Enter key is pressed, the application processes the window.

#### 1.2.2.4 Selection Indicators and Emphasis

As users interact with an application, they need to know which choices are currently selected, which choices are not available for selection and the current state of options, and which choices are associated with an error condition.

**U.S.A.I.D. CUI Guidelines** identifies the following visual cues for users:

Selection indicators and selected emphasis

Unavailable emphasis

Error emphasis

When users select a choice, the application should give some visual acknowledgement that the choice is selected. For example this may be reversed video, highlight, or blinking fields.

Unavailable emphasis is a visual cue that shows users that a choice is unavailable because some condition of the application does not allow them to select it. The application must let users know when a choice is unavailable for selection. This is normally done by "graying" the unavailable choice.

Some functions of an application may be available for some users and not authorized for others. Unauthorized selections should not be displayed for users who cannot access them. If they must be displayed, they should be marked as unavailable.

Error emphasis is a visual cue to users that they have incorrectly typed information into an entry field. Error emphasis consists of a change in the appearance of the incorrectly-typed data.

---

### 1.3 Benefits of Common User Interface

Each application may be only one of many run by its users. Users may switch back and forth between U.S.A.I.D. Budget systems to spreadsheets to database applications. To achieve consistency under these conditions, applications must look and act alike. In some cases, the components of the applications should even be identical.

Implementation of the **U.S.A.I.D. CUI Guidelines** elements should be consistent across all applications running on a terminal, but applications should take advantage of the capabilities of the particular type of terminal being used.

Consistency provides a basis for building user expectations and familiarity. It requires the user interface to act in the same way every time. This reinforces the users' conceptual model and makes the application easier to use and quicker to learn. An interface should be consistent *physically, semantically, and syntactically*.

**Physical consistency** refers to the hardware: the keyboard layout, the use of a mouse, or the ability to use sound cues. It would be physically consistent for the function keys to be in the same location on the keyboard, regardless of the system being used. This may not always be possible, when mixing programmable and non-programmable workstations, as well as different vendor models of the PC.

**Semantic consistency** refers to meaning of words and terms used in the interface. The use of "exit", "escape" and "quit" are often confusing. Does "exit", "escape" or "quit" move back one screen, or does it get you out of the entire function or application? The meaning of program specific terms is also important in order to reinforce a users expectations. Does geographic location represent a street address, or longitude/latitude coordinates? In order for users to use information that they retrieve, they must understand what they are going to retrieve.

**Syntactic consistency** refers to the sequence and order of appearance of elements on the screen and the sequence of action requests (e.g., pull-down menus that remain down when a mouse button or key is pressed) or to the fact that the title bar is always at the top of the window.

Consistency across systems is a trade-off between the desire to be consistent physically, syntactically, and semantically and the desire to take advantage of a systems' optimum capabilities. A consistent interface benefits users and applications designers and may save time and money.

Users benefit from consistency by taking less time to learn an application and, when doing an application, less time to do their work. A consistent interface reduces user frustrations and makes them feel more comfortable with the application.

A consistent interface benefits applications designers by defining common building blocks through standard interface elements and techniques. These building blocks allow programmers to create and change applications more easily and quickly.

## 1.4 CUI Interface Models

U.S.A.I.D.'s CUI Guidelines define two interface presentation styles, the *graphic interface style* and the *text interface style*. Figure 1.1 shows how the graphic and text user interfaces are related. The graphic presentation style of the graphic user interface is characterized by:

- Object oriented design and code
- Use with programmable terminals
- Graphic elements and controls
- Menu bar
- Supports pointing devices

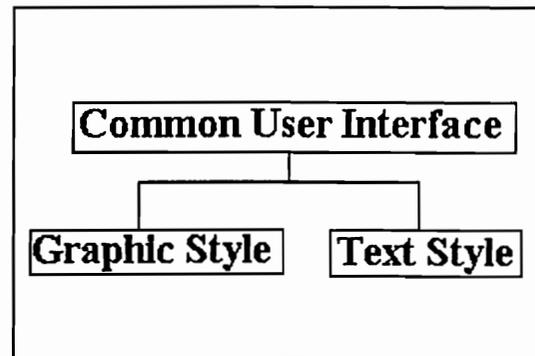


Figure 1.1 User Interface Models

The text presentation style of the graphic user interface is a subset of the graphic user interface, using only text characters and suitable for implementing on nonprogrammable terminals or with processors or languages that do not support graphic interfaces.

- Object oriented design and code
- May be used with programmable or non-programmable terminals
- Uses a subset of graphic elements and controls
- Menu bar
- May support a pointing device on programmable terminal

The graphical presentation style is preferred for all new applications, but will require that pcs be able to support a mouse and a graphical user interface, such as Microsoft Windows. The text presentation style is necessary until the Agency has retired its 'dumb terminals' such as the Wang 4230 terminals.

## 1.4.1 Common User Interface Styles

### Graphic User Interface Style

The graphic user interface makes extensive use of the PC's ability to handle graphic images, such as windows, menu bars, pull-downs, and parallel dialogues. It is used primarily for *object oriented* applications and defines standard graphic cues, such as spin buttons and check boxes.

### Text User Interface Style

The text user interface is a subset of the graphic user interface. It is used on nonprogrammable or programmable terminals. It makes use of as many of the same features of the graphic interface, such as windows, pull-downs, and menu bars, as possible. It is intended for object oriented decision intensive applications.

## 2.0 Graphic Common User Interface

This chapter defines the graphic based common user interface (CUI). The graphic user interface makes extensive use of programmable terminals' abilities to handle graphic images, such as *windows* and *menu bars*. It is used primarily for object oriented applications.

---

### 2.1 Basic Windows Concepts

The fundamental resource in a graphic user interface is a *window*. A window is a rectangular section of screen. Each window has a border, and application programs can combine two or more windows to create a flexible user environment. Windows

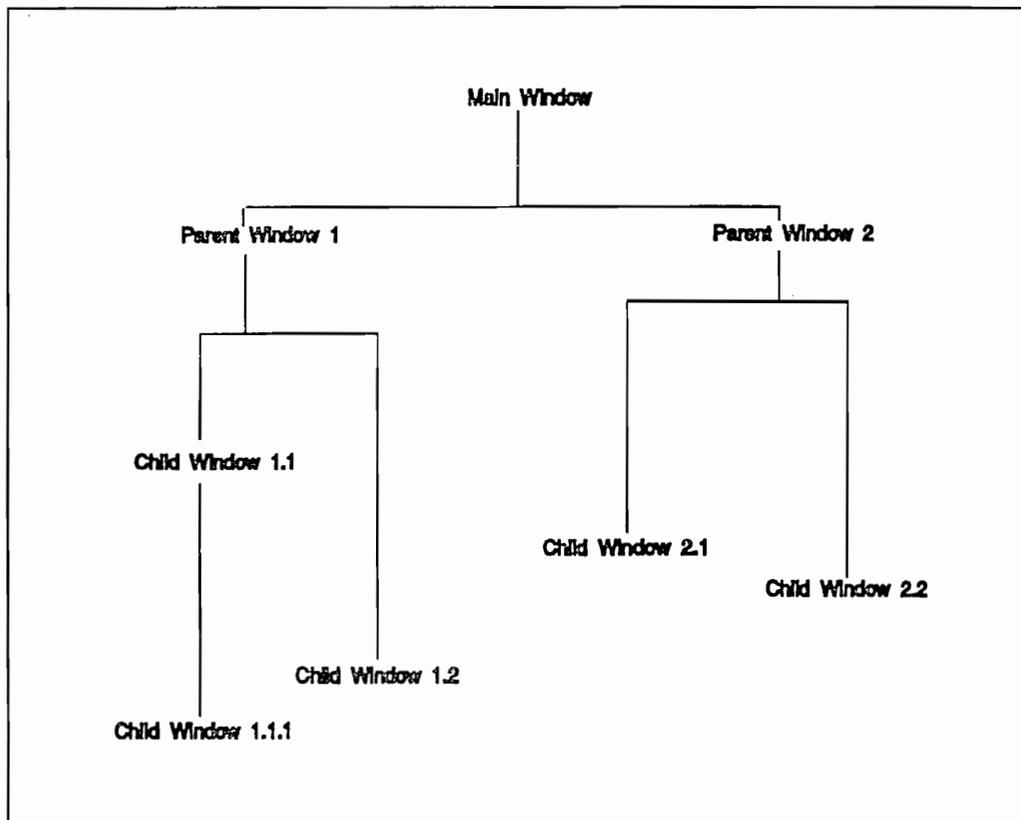


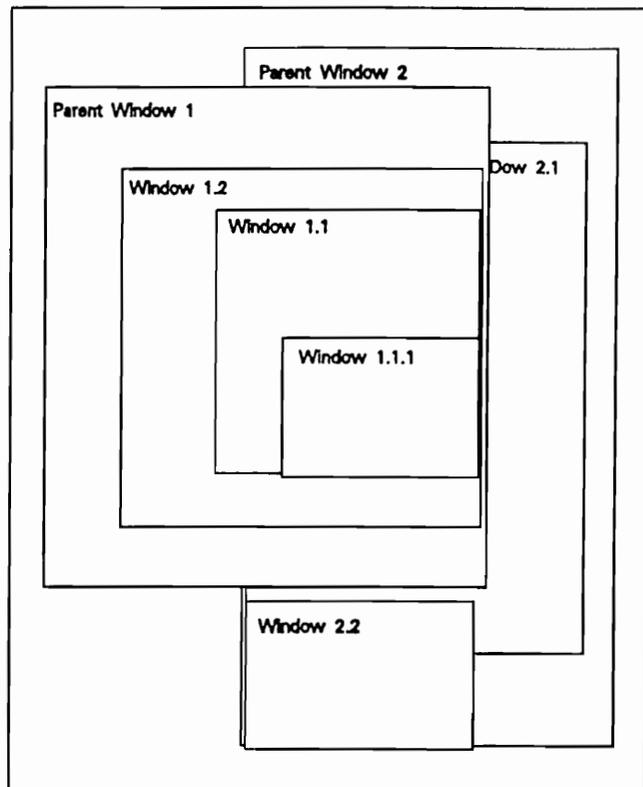
Figure 2.1 Window Tree

are organized within an application in a hierarchy. The top window in an application is the *primary window*, and all others are known as *secondary* or *child* windows. Each window except the primary window has a parent window. Child windows that share the same parent are *siblings*. **Figure 2.1** is an example of a window tree hierarchy.

Child windows have a specified order, known as the *z-order*. The topmost ordered sibling obscures any portion of its siblings and is clipped to its parent. **Figure 2.2** shows the representation of the window hierarchy displayed. Windows 1.1.1 and 2.2 are clipped to their parents and overlay their siblings. The window to which input is directed is the *focus* window. The main window of the hierarchy containing the *focus* window is the active window and is usually presented on top of all the other windows.

The primary visual components of the graphic user environment are the screen background, windows, icons, and a free-moving mouse pointer. These components are identified in **Figure 2.3**.

Within any window are additional standard interface components that provide a consistent way to present information to users. When users are familiar with an application interface, they can identify similar components of this interface when using new applications. Developers may reuse these standard components. This provides the benefits of quicker learning and reduced development time.



**Figure 2.2** Windows Tree Hierarchy

The conceptual model is based on the visual components of the user interface. In windows, applications present objects for users to manipulate. Application icons represent applications that are temporarily set aside for later use. A freemoving pointer allows users to choose where their next interaction will occur.

A window is a visual component through which an application presents objects and allowable actions for users to choose. There are two basic types of windows, primary windows and secondary windows. Each window type is unique and is used by an application for different reasons.

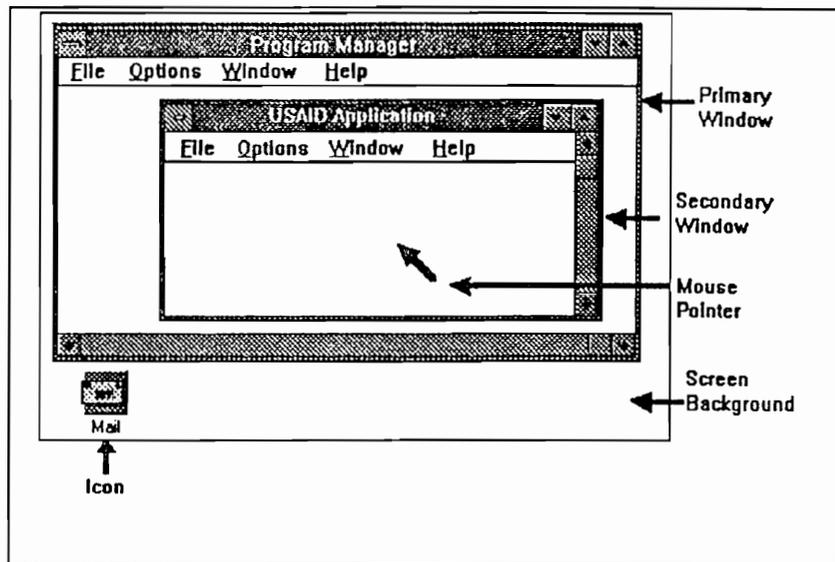
A primary window is a movable, sizable window in which objects and actions of the application are presented to the user. In the primary window, objects are presented in the work area and actions in the menu bar. The primary window is the main focus of the users' work activity. Every application must have a primary window. **Figure 2.4** is an example of a primary window.

The work area is between the top of the screen (menu bar) and bottom of the screen where the users get their work accomplished. Sometimes this area is referred to as the desktop (Apple) or the client area (IBM

SAA/CUA) or

the workbench. For some U.S.A.I.D. systems the work area may be customized, such as customized forms fill-in screens. For some types of applications the work area may be presented completely blank (e.g. word processors).

An application actually supports two types of objects: the form, which is the application object, and the user information, which is the user object. Like a user object, an application object has sub-objects and properties. For example, in a spreadsheet application, the rows and columns are sub-objects of the application object, the spreadsheet. The sub-objects of an application may have properties, such as the height of columns and width of rows in a spreadsheet. Because such characteristics as height and width are properties, the users can modify them just as they can modify the properties of user objects.



**Figure 2.3 Visual Components**

A goal of an application is to provide the most frequently used actions of the application directly in the work area of the primary window. For example, word processor type applications may allow the user to compose and edit a document without using the menu bar.

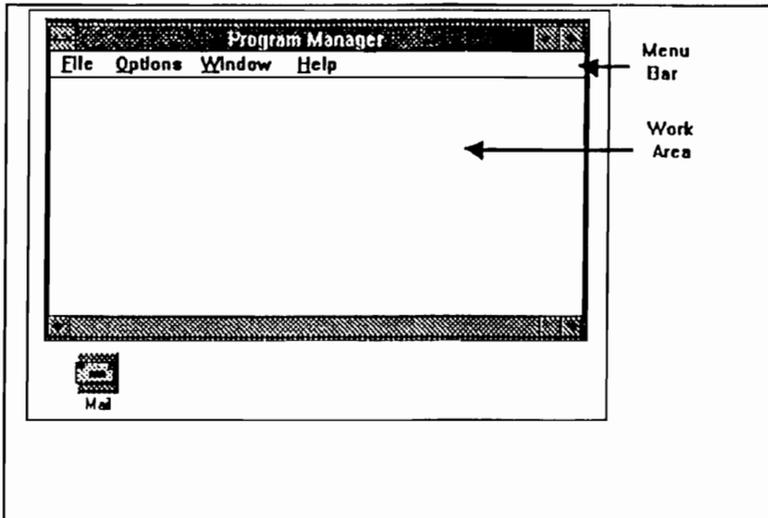


Figure 2.4 Primary Window

All other windows created or used by an application are associated with an applications primary window. Actions that affect the primary window may also affect its secondary windows. For example, when a user closes a primary window, all the supporting windows associated with that primary window are also closed. An information message to this effect should be generated and allow the user to choose this action or cancel it, complying with the forgiving principle of user interaction.

A secondary or child window is a movable, sizable window that is always associated with a primary window. Secondary windows may be used for a modeless, parallel dialogue with users, such as the dialogues for help. Help is an activity that is parallel to the window with which it is associated. The help window carries out a dialogue with users that is independent of the window from which users requested help. **Figure 2.5** shows a secondary window.

Secondary windows, rather than dialogue boxes, should be used when the information to be presented in the window may not fit and scrolling may be required to view additional information. In this situation, users should be allowed to change the size of the window so they can control the amount of the information they view.

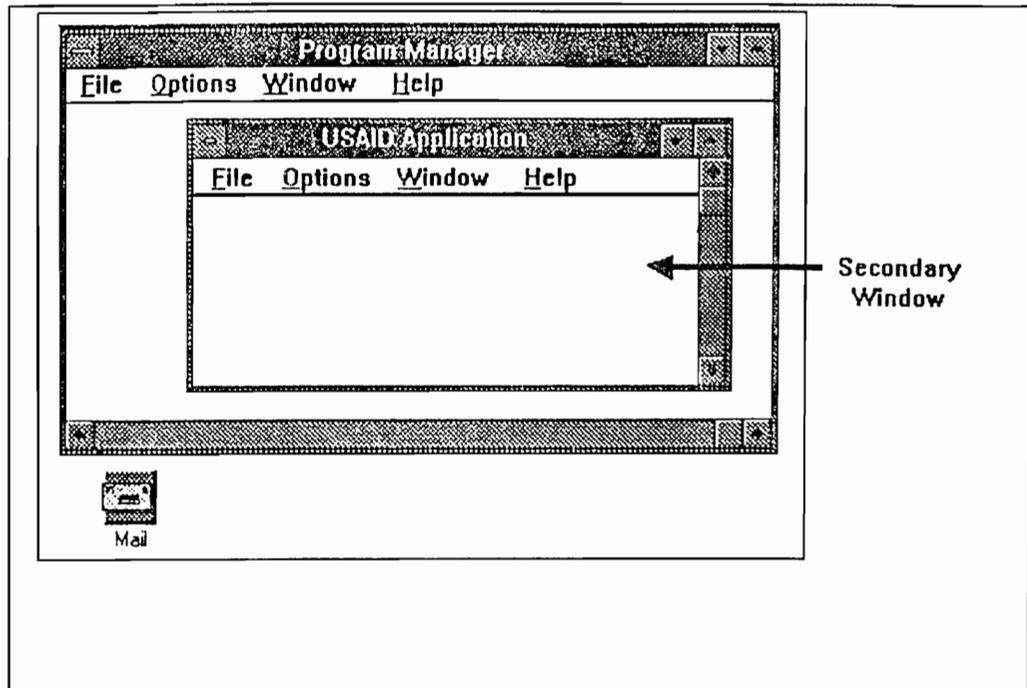


Figure 2.5 Secondary Window

### 2.1.1 Interacting with a Window

A workstation may have input devices consisting of a keyboard and/or a mouse or pointing device. The end user determines what window is to receive input, and the position of the next input character is represented by a cursor, for keyboard, or a pointer, for pointing devices. The selection of which window to direct input is under user control. Input from the pointer or keyboard is given to the application whose window lies underneath the cursor or pointer. The appropriate main window of the application becomes the active window, and the application is responsible for establishing the *focus* window.

### 2.1.2 Operating on a Window

The presentation interface (e.g. Microsoft Windows for DOS, Presentation Manager on OS/2, etc.) provides facilities that allow a user to size, position or arrange windows (e.g. cascade) on the screen.

These facilities are provided independently of the applications program. The application must be aware of the activities to let the presentation interface take appropriate action. If the size of a window changes, the application must be aware

that it may contain more (larger) or less (smaller) data. The user may alter a window by changing its size, or by moving the window. Allowable options are:

Maximize - Causes the window to become as large as the application will allow, normally the entire screen.

Minimize - Causes the window to be made as small as the application allows. Normally this results in the window becoming a small icon representing the application.

Size - Changes the onscreen size of the window, either larger or smaller. The application program can request to be notified when the user changes size of a window. As a window gets smaller, the contents become smaller. The application can reduce the contents of a window in order to allow the user to still read or understand what they are looking at.

Move - Changes the position, but not the size of the window. A window cannot be moved outside its parent. If it is moved too close to the edge, it will become clipped, and the portion outside of the parent border is not visible to the user.

### 2.1.3 User Interface

A presentation manager provides for a set of functions that allow for a dialogue between the end user and the application. These functions are encapsulated in the *controls* and *elements* of the presentation interface. Combinations of windows provides for a wide variety of user interface functions. There are three basic categories of user action that provide input to an application. These are:

Data Entry - allows the user to provide information or direction to the application.

Movement - allows the user to move the pointer.

Selection - allows the user to identify a point (usually the point of the arrow shaped pointer) of interest to the application.

## 2.1.4 Messages

The presentation interface and the applications packages must work together to present the user with a seamless approach to task management. In order to provide a truly interactive interface, the presentation manager must pass each message to the application program. All messages do not have to be acted upon by the application. Some messages may be only needed by the presentation manager (e.g. display color changes)

There are circumstances where the presentation manager is the service requestor, and times when the application is the service requestor. Normally the application requests service from the presentation manager, but there are cases when the presentation manager becomes the requestor. The presentation interface can relocate a window, but does not retain the definition of the other windows that are displayed. It must ask the appropriate applications to refresh their windows (repaint themselves).

A request for service is held in a *message* that is passed to the service provider. Messages contain the information as parameters related to the request for service. A request for synchronous service is *sent* to the provider, and a request for asynchronous service is *posted* to the queue created by the service provider.

This leads to different program logic than traditional sequential process looping programs. In traditional architecture programs, an application progressed until it was ready for input. Then it went into a wait state until the input (keystroke) arrived. The traditional way for this to occur was for the application to relinquish control to the operating system and place a value in one of the registers specifying that the operating system would reactivate the application when the user had entered a keystroke.

In an event driven (modeless) environment, provided as part of the GUI, the user may choose to click on *Exit* instead of entering the next character. If the program had transferred control to the operating system, it wouldn't be ready to receive and act on the mouse click action.

The application and presentation interface must work in partnership to exchange messages, and the application becomes a collection of resources and routines that the operating environment calls upon as required by the users actions.

## 2.1.5 Presentation Manager

A graphic interface requires a great deal of code to be written in order to provide the necessary input and output (windows, etc.) functions to the applications programmer. In the interest of providing both portability and reusability, a graphic *presentation manager* is used by the applications programmers. This presentation manager provides at least some of the following functionality:

- A windowing system, providing elements and controls for use by the programmer
- Support for user interaction by means of the keyboard or pointing device.
- Graphics support for screen display
- Message handling from the presentation interface to the application
- Ability to save and restore images
- Support (drivers) for many different displays and printers
- Support for different fonts

The purpose of a presentation manager is to provide the functionality required by the application program to supply input, output and control functions to the end user. For example, Microsoft Windows with its Windows Developers Kit, and X-Windows through its toolkit, and Style Manual provide all of the functionality of a presentation manager.

In addition to the presentation manager, an application programmer can benefit by using a dialogue manager. A dialogue is the interaction between the user and the computer program through a single primary window and all of its associated windows, pop-ups, dialogue boxes, and elements. A dialogue manager may provide capabilities for verifying input data, provide range and validity checking, translate data or use assignment lists to look up entered data. A dialogue manager may also display messages on error conditions, or other standard program conditions. A dialogue manager can handle Help requests from the user. One example of a dialogue manager is EASEL from EASEL Corporation.

---

## 2.2 Presentation Elements of Graphic User Interface

Both primary and secondary windows are composed of standard components. Several window components are used in all window types while others are specific to a particular window type.

### 2.2.1 Basic Window Components

The fundamental window components are the *title bar*, *window border*, *menu bar*, *work area*, *elements and controls*, and *scroll bars*.

#### 2.2.1.1 Title Bar

The Title Bar identifies the window to the user and provides controls for closing or resizing the window. It contains the *system menu button*, the *window title* and the *window control buttons*.

#### 2.2.1.2 Window Border

Windows may be sizable or non-sizable. Each type of window border is visually distinctive. A sizable window border is a color boundary containing eight segments. The segments provide users with a visual cue that they can change the window size. Non-sizable window borders are not segmented. Non-sizable borders may be used for dialogue boxes and menus.

#### 2.2.1.3 Menu Bar

The menu bar is the area of a window that contains the actions of an application. It is positioned directly below the title bar. An application that supports more than one action must have a menu bar in its primary window. A menu bar may contain multiple lines and use words as well as icons as directions to users.

#### 2.2.1.4 Work Area

The work area of a window is the part of the window inside the border that is below the menu bar. The work area is the focus of the users' attention. The work area of the primary window is where you present to users the object that users want to work on.

#### 2.2.1.5 Window Elements and Controls

A control is an interface component that enables users to select choices and type information. Controls, like other interface components, provide consistency in both presentation and interaction. Each control has a unique appearance and gives users a specific way to interact with it. This includes pushbuttons, radio buttons, check boxes, lists boxes, spin buttons, and field and group identifiers.

#### 2.2.1.6 Scroll Bars

Scroll bars provide users with a visual cue that more information is available and that the unseen information can be manipulated into view using the mouse or cursor arrow keys to scroll information. Scroll bars should be included in all sizable windows.

### 2.2.2 Title Bar

The title bar has two purposes: it identifies an application or function to users through the window title, and it is a visual cue to users that they may move a window. Users may move a window by clicking on the title bar and holding down on the mouse button while they relocate the window.

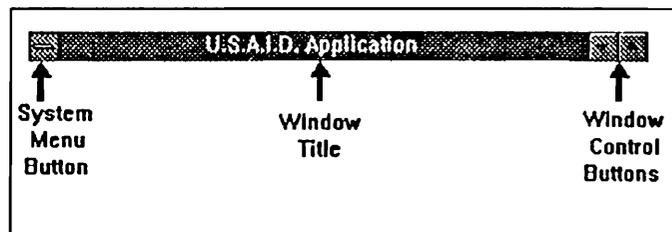


Figure 2.6 Title Bar

The window title bar consists of three parts: the system menu button, the window title, and the window control buttons. Figure 2.6 is an example of a window title bar.

### 2.2.2.1 System Menu Button

Users can use the system menu button to display a pull-down containing the actions that users can perform on a window. The actions may be any of the following: Restore, Move, Size, Minimize, Maximize, Close, and Switch to..., as applicable.

### 2.2.2.2 Window Title

The window title of the primary window contains the application name and the file name, if applicable. The file name represents the stored object users are modifying. If users have not yet named a file, an application can display **Untitled** or may supply a file name consistent with a naming convention that is meaningful to the user.

### 2.2.2.3 Window Control Buttons

Window control buttons provide a fast way to use the mouse to select three of the system menu actions: Minimize, Maximize, and Restore.

## 2.2.3 Window Border

The application provides the initial size of its windows, but window sizes may vary depending on the need for users to work in them. The user should have the ability to resize primary and most secondary windows. This is done by means of the border that surrounds the window. A sizable window border is a color boundary containing eight segments (two horizontal, two vertical, and four corners). This segmentation provides a visual cue that windows are sizable.

A window may have a non-sizable window border. Each type of window border is visually distinctive. Non-sizable window borders may be used for dialogue boxes or messages. Non-sizable window borders are not segmented.

**Figure 2.7** is an example of a sizable window border.

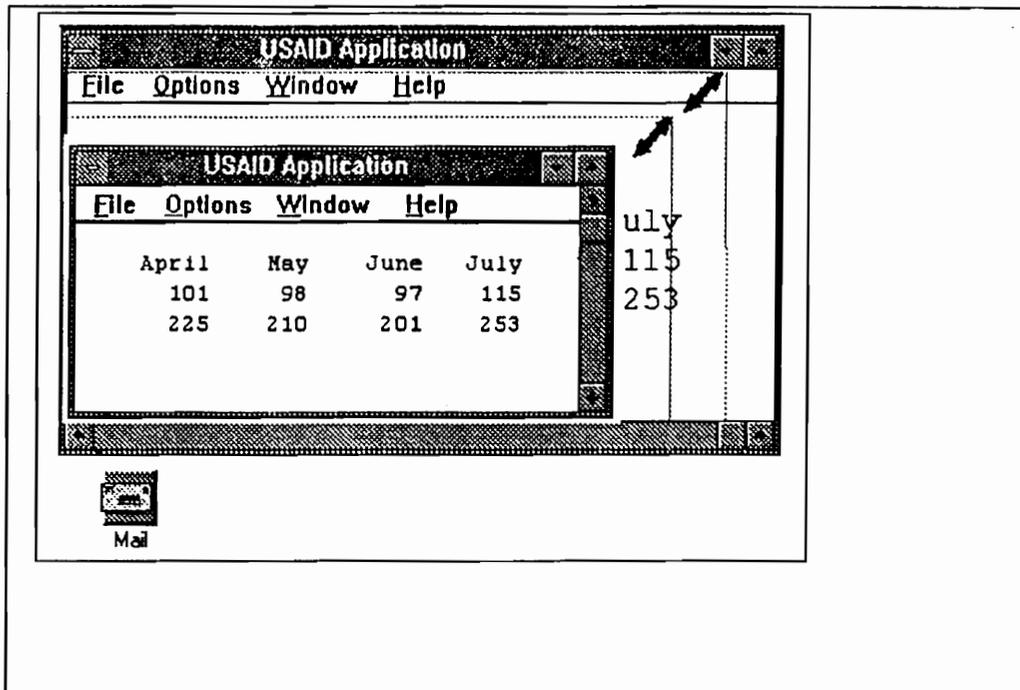


Figure 2.7 Sizable Window Border

## 2.2.4 Menu Bar and Pull-Downs

Menu bars and pull-downs are user interface components that are used to present application actions to users. Menu bar and pull-down choices can be commands, or they can be properties that apply to an object in the work area (e.g., color palettes or style selections).

List single word choices left to right in descending order of use with *Help* being the choice on the farthest right. Place the menu bar choices, if applicable, in the following order:

*File*

*Edit*

Any application-specific choices (e.g., *View*, *Special* or *Options*)

### Help.

Provide each menu bar and pull-down choice with a unique single-character mnemonic that can be used to select the choice. This provides a fast interaction technique for selecting choices from the keyboard. When users type a valid mnemonic, the selection cursor moves to that choice, and it is automatically selected or de-selected as appropriate. A mnemonic may be identified with a special color or the character underscored to provide a visual cue to users that mnemonic selection is available. In each window, a mnemonic must be unique within the menu bar and within its associated pull-down.

If possible, assign the first letter of a choice as its mnemonic. Otherwise choose another letter in the choice that seems reasonable for the context of your application (e.g., Enter and Exit). If you must choose a letter or character not contained in the choice, enclose the mnemonic in parentheses immediately following the choice.

If a choice has an associated dialogue box or a secondary window, place an ellipsis (...) or right pointing triangle, immediately following the choice.

For frequently used pull-down choices, the application may provide an accelerator key immediately following the choice. An accelerator assignment may be one key or a combination of keys, such as Ctrl + F5. (NOTE: Use a plus sign to indicate that users must press two or more keys at the same time.)

In a graphics environment bit maps and icons may be used in pull-downs. For example, a drawing application may have a pull-down for selecting graphic fill patterns.

### **2.2.4.1 Operation of Menu Bars and Pull-Downs**

The menu bar is activated by direct selection of an action choice using the mouse pointer or by pressing the *Alt* key from the keyboard.

Accelerator keys assigned for the active window are always active, whether or not the pull-down is displayed (e.g. if F10 is the *File Save* accelerator key, then a file may be saved at any time by pressing F10).

The appearance of pull-downs may be modified to convey information to the user based on conditions in the application:

Choices that are currently unavailable may be displayed in reduced contrast (gray). Selection of an unavailable choice results in a beep and the continued display of the pull-down. Help should be available even if the selection is unavailable (e.g., Edit choices before any data has been entered).

A check mark preceding pull-down choices indicates the current status of the choices.

Users are more productive when they are presented with only a small number of choices. A second level of choices may shown by using *cascading pull-downs*. This is a vertical list of choices associated with a first-level choice.

The first-level choice acts as a group heading in the same way that the menu bar choice acts as a group heading for its pull-down. A right-pointing triangle appears at the right of the pull-down choice text as a visual cue that this choice has an associated, hidden pull-down. This is similar to an ellipsis, which indicates a dialogue box is associated with the selection. Figure 2.8 is an example of a cascading pull-down.

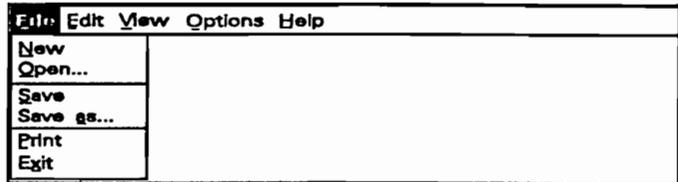


Figure 2.8 Cascading Pull Down

### Standard Menu Bar Pull-Downs

U.S.A.I.D. CUI Guidelines defines common actions and associated pull-downs for File, Edit, and Help. U.S.A.I.D. CUI Guidelines also provides guidance for View and Options. The common actions usually apply to the object in the work area of a window. This section provides guidelines on when to include these actions in the application. If the application supports File and Edit, they should be the first two choices in the menu bar. If application developers include Help in the application, it is the last choice in the menu bar.

U.S.A.I.D. CUI Guidelines defines a standard layout for the File, Edit, and Help pull-downs. This layout includes the text for the pull-down choices. Standard accelerators are also provided for some of the more frequently used functions. Accelerators are function keys that invoke the associated action without accessing the menu bar and a pull-down. Accelerators may also be used for application specific actions that are used frequently.

## U.S.A.I.D. Common User Interface Guidelines

---

Following is a description of three standard pull-downs, File, Edit, and Help, and a general description of the View and Options actions. Each description includes general information and a figure showing the pull-down layout. For File and Edit, each figure is followed by a description of how the application must implement each pull-down action.

The File pull-down enables users to manipulate stored objects, such as files, as a whole. Every application that manipulates stored objects must provide the File pull-down. Some actions in the File pull-down, such as New, Open, and Exit, have the potential to lose unsaved changes. As part of a forgiving user interface, applications must be able to recognize this situation and display a message telling users that their unsaved changes will be lost. The application should also ask users if they want their changes saved. If so, the application should perform a save, as if users had selected the Save action from the File pull-down.

File actions are organized by task: *selecting* actions, *saving* actions, and *output* actions. System developers may add to this pull-down other actions that involve manipulating an object. Any activity that involves manipulation of an entire object is a candidate for inclusion in the File Pull-Down. Some application specific actions that may be included in the File pull-down are Copy, Move, and Delete. If an action has several options, a pull down list may be used, or the option may be given its own heading on the menu bar. This is reasonable if the option is accessed frequently, e.g. *Print*.

Following are descriptions of the File pull-down actions:

*New* allows users to create a new file. The application should ensure that a file with the same name does not already exist.

*Open* reads an existing file.

*Save* writes the existing file to a disk. When an object is untitled, as it might be after requesting the New action, an application must prompt users for the file name or supply a default name consistent with the application conventions (e.g. DEFAULT.FIL).

*Save as* writes the existing file as a new file without changing the original one. The application prompts users for a new file name, using a dialogue pop-up. The application should display a warning message that data is about to be lost if users select an existing file name as the new name.

Print allows a user to schedule and print a file on a user selected device or to store as a disk file.

Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. The Exit action should always appear as the last choice in the first (left most) pull-down.

### **Edit Pull-Down**

The Edit pull-down allows users to perform the types of actions that are similar to those needed to reorganize paper documents in the real world.

In keeping with the user directed attribute, this real-world metaphor for paper documents is transferred to the computer environment for electronic documents through the actions available in the Edit pull-down. For example, users select a paragraph they want to move and then select the Cut or Copy action in the Edit pull-down. If Cut is selected the paragraph disappears from its location in the document. Users select a destination point in the document and select the Paste action in the Edit pull-down. The paragraph is inserted at the new location.

The Edit pull-down contains common editing actions that apply to many types of files. Any application that supports editing must provide these common editing actions in the Edit pull-down.

System designers may add application-specific actions to the Edit pull-down, as appropriate.

Following are descriptions of typical Edit pull-down actions:

Undo reverses the most recently executed user action. Because the Undo action deals with hidden objects, it should be modified dynamically to reflect exactly what is being undone (e.g. **Undo last change**, or show what would be undone by using graphic emphasis).

Cut copies the selection to the clipboard and removes it from the object being edited. Depending on the type of object being edited, the space occupied by the removed portion may be either retained or compressed. For example, text applications usually compress the space, whereas graphics drawing applications usually leave the space blank.

Copy produces a duplicate of the selected portion of an object on the clipboard without removing the selected portion from the object that is being edited.

Paste copies the contents of the clipboard onto the object being edited at the location selected. For text applications that support word wrap, information to the right and below the selected location is shifted and formatted. However, for certain applications, such as graphics drawing applications, it may be appropriate to overlay the information. The last copy or cut should remain on the clipboard to support multiple pasting.

Mark/Unmark (optional) places or removes the emphasis from the currently selected portion of the object. This is sometimes referred to as the block command.

Delete removes the selected portion from the object without copying it to the clipboard.

### **Help Pull-Down**

The Help pull-down provides users with access to various kinds of help information. Help should be context sensitive and provide information about a specific item or field, an application window, or the help facility. All applications that access online help information must provide the following options from the Help pull-down.

Contextual information when help is accessed and the cursor is on a choice or in an entry field in an application window.

Information about the application window, known as Extended help, when the cursor is not on a choice or in an entry field.

Information about the help facility when help (F1 function key) is requested from within a help window.

If a help tutorial facility is provided, then the Help pull-down may provide access to it or at least information about how a user may access it outside of the application.

### **View Pull-Down**

The View pull-down allows users to select different ways to look at an object without affecting the object itself. This pull-down contains actions that control such viewing specifications as how much information is presented, the presentation order, the format, and the scale.

### Options Pull-Down

Provide an *Options* pull-down if users need a way to customize the object itself. Like the View pull-down, the Options pull-down is specific to a particular application, so its contents cannot be standardized. For example, in a document editor, *Options* may allow a document to be displayed in memo, letter or report style.

## 2.2.5 Work Area

The work area is the area in which users perform most application level tasks. The work area is inside the window borders and may contain multiple work areas or secondary windows.

Figure 2.9 is an example of the work area of a window.

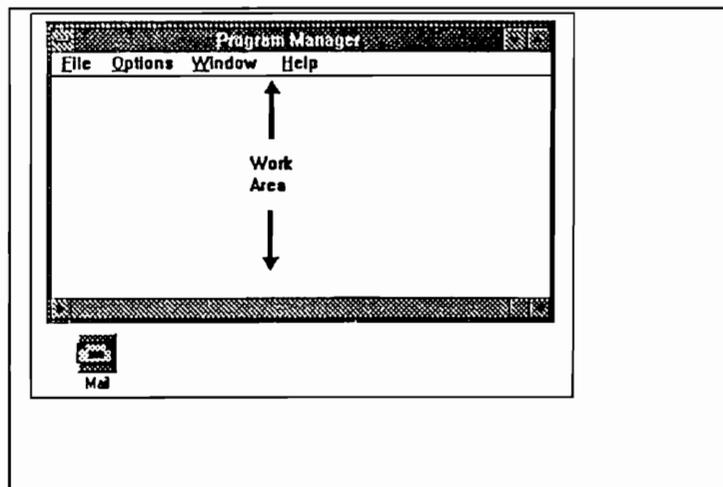


Figure 2.9 Work Area

## 2.2.6 Window Elements and Controls

Standard window elements and controls are the following:

*Field and Group Identifiers* define and identify the purpose of a selection or entry group or field.

*Radio button* allows a user to select one of a set of options that are mutually exclusive. If a user selects a radio button and then chooses another, the first one is automatically de-selected. They are defined as partially filled in circles.

*Check box* allows a user to select one or more of a set of choices. These choices are not mutually exclusive (e.g., text as bold, underscore or italic) and the user may select as many as required.

*List box* contains a list of choices that the user may select from. The list may be scrollable (e.g., a list of files in a directory) and the user will normally select one.

*Value Set* is similar to radio buttons; this allows a user implicitly to select choices.

*Spin Button* allows a user to complete an entry field by scrolling through a limited set of choices organized in a way that makes sense to the user (e.g., a list of state names in alphabetic order or days of the month).

*Combination Box* combines the attributes of an entry field and a list box. This allows the user to select from a list or type the entry in, whichever is more appropriate. An example is a list of files in a directory. The user may type in the correct directory and file name rather than switching directories and scrolling through the list of files.

*Drop-Down Box* is similar to a combination box and may be used if presentation space is limited or when users will more often complete the entry field.

*Drop-Down List* is similar to a drop-down box, but it does not have an entry field. Drop-down lists have a default value filled in. An example of a drop-down list is printer selection, where the printer must be defined to the system prior to selection and the default printer is normally selected.

*Dialogue Box* is used by the application to obtain information from the user before the application can continue to process. Dialogue boxes may be modal or modeless. A Search or Find function is an example of a modeless dialogue box.

*Entry Field* is a single or multiple line entry area provided by the application to allow the user to type in information.

*Pushbuttons* display actions in those windows that do not contain a menu bar: dialogue boxes, message boxes, and some secondary windows. Usually, these window types have a limited number of actions associated with them, making a menu bar inefficient for users. Pushbuttons normally execute their action immediately, instead of presenting other choices.

*Logo Window* displays information pertinent to the application such as name, version number, and U.S.A.I.D. logo, as appropriate.

*Scroll Bars* allow the user to move up/down or right/left to see additional information that is hidden by the current window borders.

### 2.2.6.1 Field and Group Identifiers

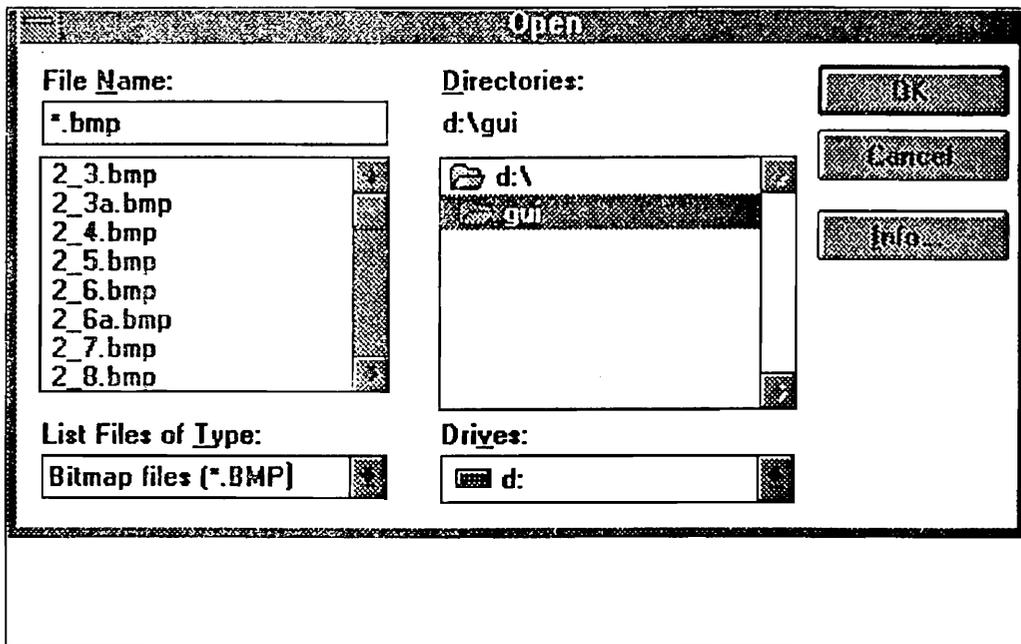


Figure 2.10 Open Dialogue Box

#### Field Identifier

A field identifier is an interface component that describes the purpose of a selection or entry field. It is descriptive, static text for selection and entry fields. It is normally located above or to the left of a field. **Figure 2.10** is an example of an Open dialogue box with field prompts.

A field prompt should also be used for output information.

A field prompt that is located above a field should be left-aligned with the beginning of that field.

A field prompt that is located to the left of a field should be left-aligned with other field prompts.

For entry fields, in addition to a field prompt, you may provide users with other helpful information. For example, if users are to type into an entry field a value that has to be within a certain range of values, you could display the range next to the entry field, as shown in **Figure 2.10**.

### **Column Heading**

A column heading identifies an entry field or selection field that has items in a column and of the same type, such as the items in a list box.

A column heading is located above the field it identifies. If information can be scrolled vertically, place the column heading so that it does not scroll and the heading remains visible. If information scrolls horizontally, the column heading scrolls to remain above the appropriate area.

### **Group Heading**

A group heading identifies a related set of entry fields or selection fields or both. You may use them alone or with field prompts. Use a group heading to identify the related fields and field prompts to further identify the individual fields.

If group headings are located in windows that scroll, the headings scroll with the items they identify.

### **Group Box**

A group box is a single-line rectangular frame with a title that groups related choices.

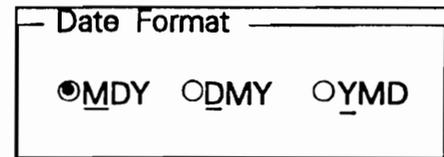
A group box provides a visual cue to users by isolating and naming groups of controls that work together. A group box may contain a single set or many sets of choices. The title appears at the top left side of a group box.

### 2.2.6.2 Radio Button

A radio button is a two-part control consisting of a circle and choice text. Radio buttons are combined to show users a specific set of choices that are mutually exclusive. Users can select only one radio button in each selection field. If users select a different one, the previous button is no longer selected.

A radio button field contains at least two choices, one of which is normally selected. When more than one object is selected in the work area of the primary window, a situation can occur in which the current state of the selected objects cannot be reflected by one radio button. For some of the objects selected, one radio button applies. For others, another radio button applies. In that situation, all radio buttons in the group should be off when the group is initially displayed.

When users select a radio button, its circle is partially filled in to show users that the radio button is selected. **Figure 2.11** is an example of a Radio Button selection.

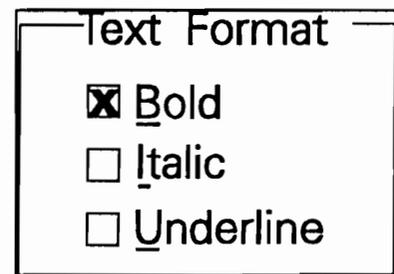


**Figure 2.11 Radio Buttons**

### 2.2.6.3 Check Box

A check box is a two-part control consisting of a square box and text describing the choice. Check boxes can be used alone or grouped in related sets. Check boxes are not mutually exclusive; when users select more than one check box, all other check boxes remain unchanged. When users select a check box, its square box is filled in with an X.

Similar to radio buttons, when more than one object is selected in the work area of a primary window, a situation can occur in which the current state of a check box is neither completely on or off. For some objects selected, the selection is true; for others it is not. When this happens, the square box is filled with gray. For example, users can select a block of text and have some, but not all, of the text in bold. A set of check boxes

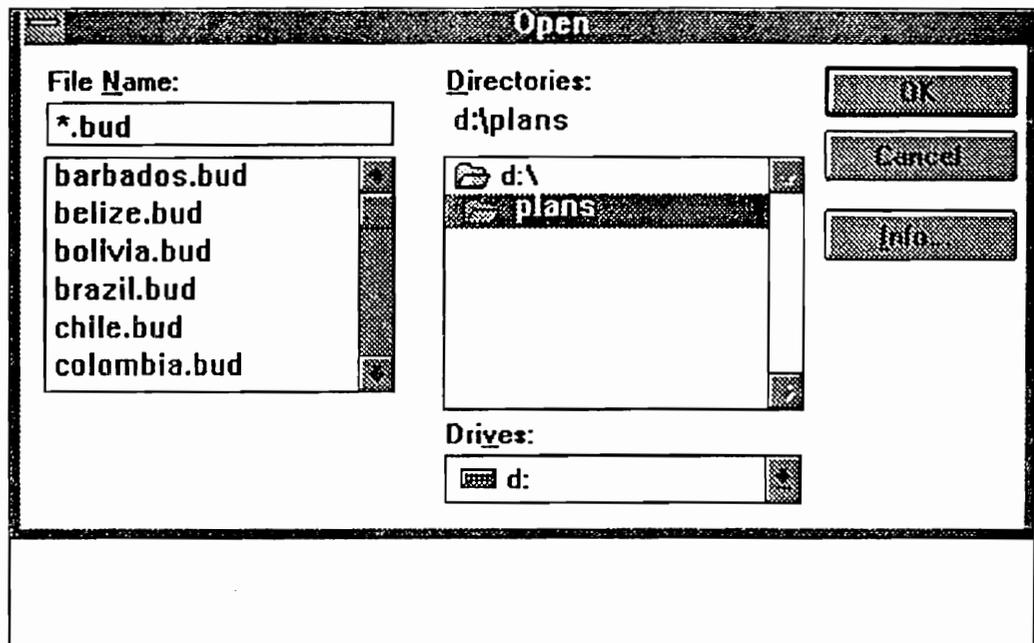


**Figure 2.12 Check Box**

displaying the current state of the properties of that text would contain a Bold choice whose square box is filled with gray. This gray color shows users that some of the selected text is bold but the rest is not. **Figure 2.12** is an example of a check box.

#### 2.2.6.4 List Box

A list box is a rectangular box with scroll bars that contains a scrollable list of choices from which users can select one choice. A list box may be used to show a



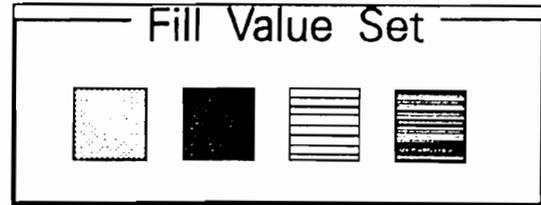
**Figure 2.13** Dialog Box with List Boxes

variable length list (e.g., files) or a long fixed length list that does not normally fit in the display area. **Figure 2.13** shows a dialog box for Open. It contains two list boxes, Files and Directories.

#### 2.2.6.5 Value Set

A value set is a special type of single selection field that is provided by an application and allows users implicitly to select choices. As with radio buttons, value sets support mnemonic selection when the value set contains text. **Figure 2.14** is an example of a value set.

Choices may be either text or graphics.



**Figure 2.14 Value Set**

Allow users to select a value set choice by mouse or keyboard--with a mouse by clicking on a choice or with the keyboard by moving the cursor to a choice.

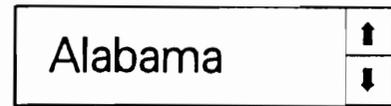
The choice selected is identified by a dotted outline box.

Selected emphasis may also be supplemented by using color.

Choices that are currently unavailable are identified by reduced contrast (gray).

### 2.2.6.6 Spin Button

A spin button is a special type of entry field that is provided by the application to allow users to complete an entry field by scrolling through a finite set of choices. For example, you may use a spin button to allow users to select a valid state name from a list of states in a region, as shown in **Figure 2.15**.



**figure 2.15 Spin Button**

A spin button should be used only when the valid list of choices is in recognizable order.

Spin Buttons must contain a default choice.

They allow users to type directly into the entry field or to scroll a list of choices with either the mouse or keyboard. The users scroll through the list using the mouse to click on either of the arrows or with the keyboard, using the appropriate arrow keys.

### 2.2.6.7 Combination Box

A combination box is a control that combines the capabilities of both an entry field and a list box. Use a combination box when users can type information but when your application can provide a list of possible choices to complete the entry field.

The list box appears beneath and to the right of an entry field. **Figure 2.16** shows a combination box.

The application should have space for only six to eight choices in the list box.

The choices in the list box should be in an order that is obvious to users, considering the context of the entry field.

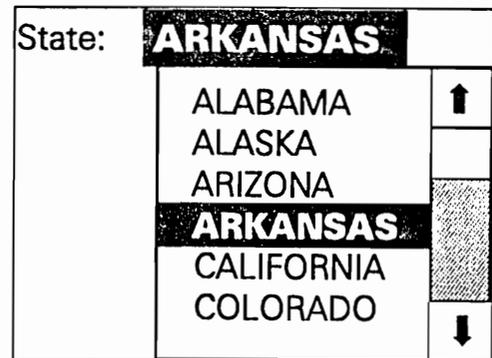


Figure 2.16 Combination Box

### 2.2.6.8 Drop-Down Combination Box

A drop-down combination box is a variation of the combination box in which the list box is hidden until users request it. Users are given a visual cue of a downward-pointing arrow at the right side of an entry field to indicate a list box is available.

Use drop-down combination boxes for either of two situations. Use one when presentation space is limited, making a standard combination box undesirable; or use one when users will complete an entry field more often by typing text than by choosing the entry field text from a list. **Figure 2.17** shows an example of a drop-down combination box.

### 2.2.6.9 Drop-Down List

A drop-down list is similar to a drop-down combination box, but it does not have an entry field for typing text. Instead of an entry field, users see a single selection field with one choice displayed as the default value. As for drop-down combination boxes, users have the same visual cue that a list is hidden; they see a prompt box

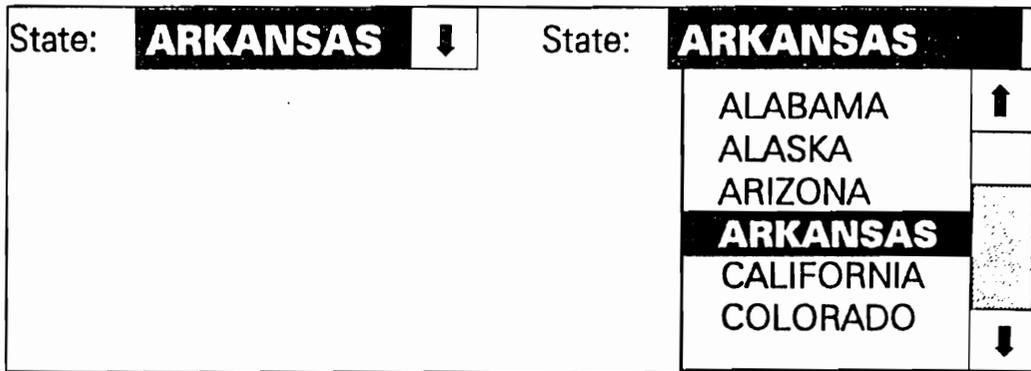


Figure 2.17 Drop-Down Combination Box

containing a downward-pointing arrow at the right side of the selection field.

A drop-down list may be used in place of a list box when a choice is not changed frequently and space is limited. For example, a drop-down list can be used when users want to select a different printer. Assuming users do not change printers often, they do not need to see a list of available printers every time the printer selection field is displayed. Figure 2.18 is an example of a drop-down list.

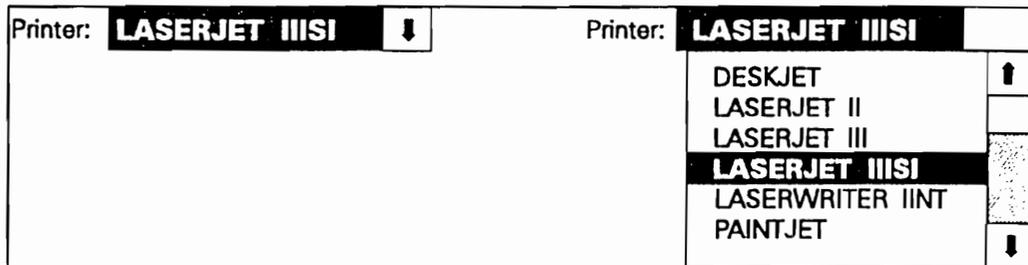
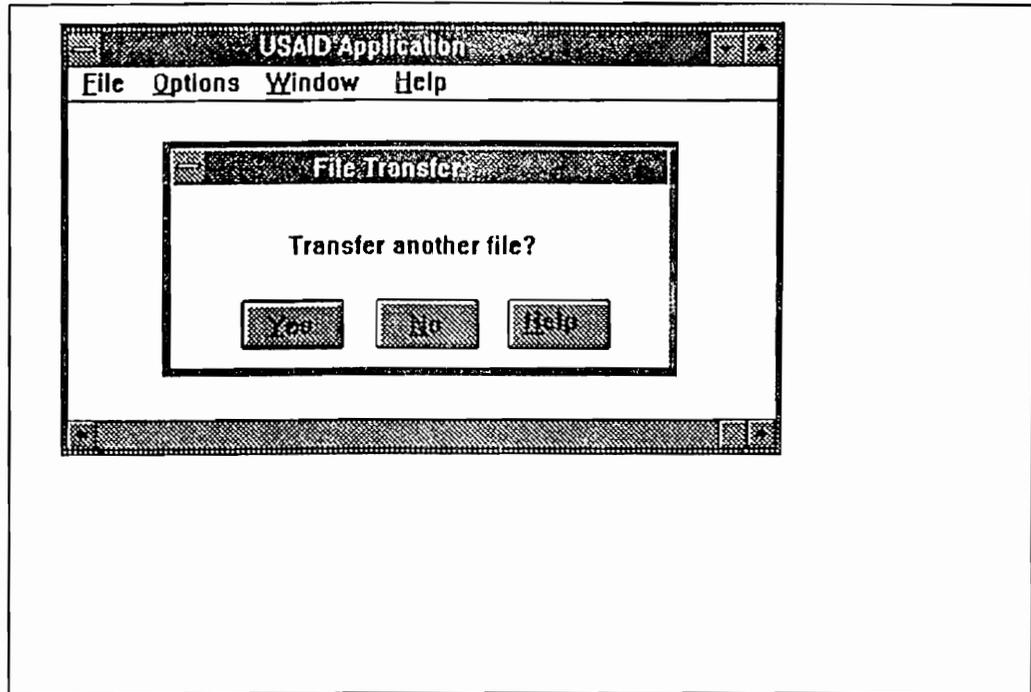


Figure 2.18 Drop Down List

### 2.2.6.10 Dialogue Box

A dialogue box is a movable window, fixed in size, in which the application can request information required to complete an action. A dialogue box is associated with a specific window. Figure 2.19 is an example of a file transfer dialogue box within a window.

The size of the dialogue box is determined by the application and cannot be changed by users. The size should be as small as possible to contain all



**Figure 2.19 File Transfer Dialogue Box**

information without the need for scrolling. This should make it less likely that a dialogue box will cover pertinent information in the underlying window. After the dialogue box is displayed, users can move it around the screen, if necessary, to see any information in the underlying window that has been covered by the display of the dialogue box.

A dialogue box may be modeless or modal. A modal dialogue box provides a fill-in-the-blanks which users must complete before they can interact with any other window associated with that application. For example, if users try to save a file that has not been named, a dialogue box is displayed because the application must have a file name before it can complete the action. A modeless dialogue box allows parallel dialogue from which users may do something else before they complete the dialogue. Users may switch between a modeless dialogue box and its associated window.

Modeless dialogue boxes, rather than secondary windows, should be used when all the information to be presented fits within the box and scrolling is not required. List boxes and entry fields can scroll individually without forcing the entire window to scroll.

A Search request is an example of a modeless dialogue. Usually, when an application searches for something, more than one match is possible and it is necessary to check out several matches before the user finds the correct information. A modeless dialogue allows users to find an object through a dialogue box, switch to the primary window to modify the object that has been found, and then return to the dialogue box to continue searching for the same or another object. A modal dialogue would require the user to initiate the find dialogue each time it was necessary to search for another match. A modeless dialogue allows users to repeatedly perform an action without having to reinitiate the dialogue.

A modeless dialogue box remains displayed until users either request the *Cancel* action in the dialogue box or select *Close* from the system menu pull-down.

A message box is a type of modal dialogue box that is used exclusively for displaying messages to users. A user must *Cancel* or *OK* a message box before continuing to process.

Preselected choices may be used when applicable to help users complete dialogues. If used, these choices should reflect the current state of options. Preselected choices allow users to proceed immediately to the next field when this choice is also the users' most requested choice.

### 2.2.6.11 Entry Fields

An Entry field may be a single-line or a multiple-line field. A single-line entry field is a two-part control consisting of descriptive text and a single line box into which users type information. An application may decide to initially display an entry field blank or with default text.

**Figure 2.20** is an example of a single line entry field.

The length of an entry field should be equal to the average amount of text that will be entered into the field. The application can provide scrolling so the user can see text that exceeds the normal length.

A multiple-line entry field consists of a rectangular box into which users type information that consists of more than one line. For example, users may need more than one line to input text for a comment or memorandum field to be appended to a record.

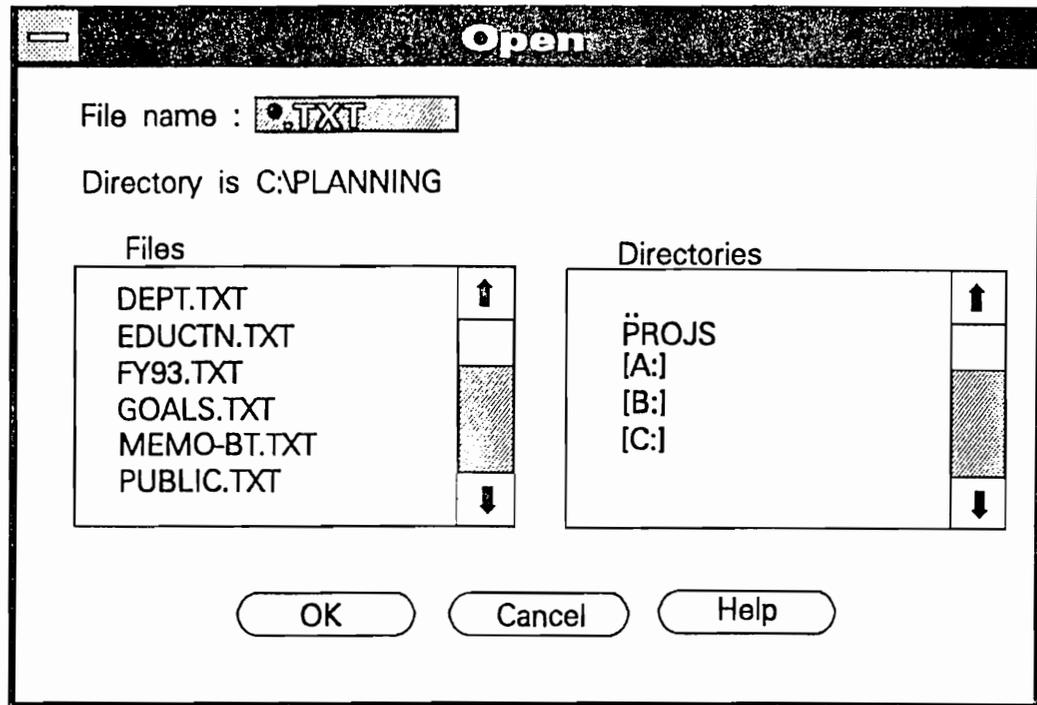


Figure 2.20 Single Line Entry Field

Scroll bars are provided so users can see the text that exceeds the size of the entry field box. Applications may select the color of the window background and the font and color of the text.

Normally the Enter or Tab key should be used to accept input in an entry field. In some cases an Auto-return, which causes the cursor to move to the next field after the user has typed in the last character of a fixed-length field, may be used if required. Auto-return can be useful for high-volume data-entry fields that are filled completely with text. In order to provide consistency, avoid using Auto-return when you have a mixture of entry fields, some that must be completely filled and some that can be partially filled.

In dialogue boxes, the Enter key is used to perform the default action; this action usually submits the dialogue to the application. However, the Enter key may also be needed to move the cursor to the next line. The application may have to support both of these actions by enabling the Enter key to work contextually according to the location of the cursor.

If the cursor is in a multiple-line entry field, pressing the Enter key moves the cursor to the beginning of the next line. Any text that

was to the right of the cursor prior to pressing the Enter key is repositioned on the next line.

If the cursor is not in a multiple-line entry field, the Enter key performs the action defined in the window. For example, in a dialogue box, pressing the Enter key will perform the default action of the dialogue box.

Both single-line and multiple-line entry fields should support insert-and-replace modes. Users toggle between the two modes using the Insert (Ins) key. The mode should be obvious to the user by a change in the shape of the cursor (e.g., replace mode is indicated by an underscore cursor and insert mode by a blinking block cursor).

Both single-line and multiple-line entry fields provide users with the capability to automatically delete selected text and insert new text at the starting point of the selected text.

When users type their first character, the Replace function automatically deletes the previously selected text. The typed character and any additional typed characters are inserted at the starting point of the selected text.

### 2.2.6.12 Pushbuttons

A pushbutton is a rounded-corner rectangle with text inside that shows users the actions available in dialogue boxes, secondary windows, and message boxes. Pushbuttons and menu bars provide users with access to actions. A window should usually contain a menu bar or pushbuttons.

**Figure 2.21** is an example of a pushbutton.

To maintain layout consistency for pushbuttons, place the pushbuttons horizontally in the lower part of a window or dialogue box. If this position does not adequately support your window layout, the pushbuttons may be placed vertically at the right side of the dialogue box.

Pushbuttons that are arranged horizontally should be the same height, and pushbuttons arranged vertically should be the same width.

When you display several pushbuttons in the same dialogue box, identify one of them as the default action by using a bold border. When users move

the selection cursor between pushbuttons, the bold border moves with the cursor.

The bold border usually appears on the action that lets users respond positively to the dialogue, for example, on the OK action. However, when the dialogue box supports an action that would destroy or modify data, such as Delete, the default should be Cancel. Users, therefore, have to explicitly change the selection from Cancel in order to perform a destructive action.

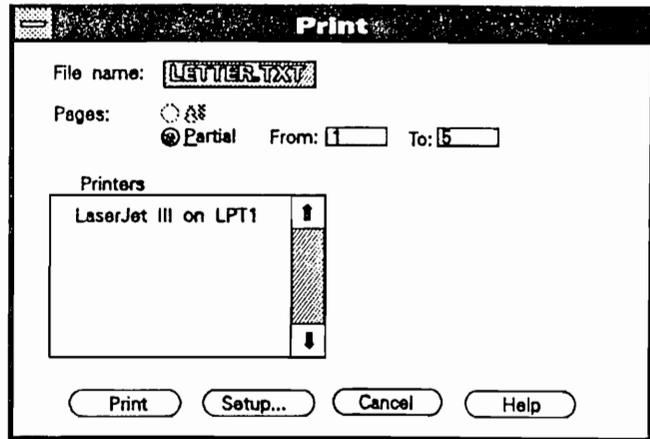


Figure 2.21 Pushbuttons

You should use an ellipsis (...) in pushbuttons when the action associated with the pushbutton results in another dialogue box.

Do not leave any space between the pushbutton text and an ellipsis.

### Standard Pushbuttons

To promote consistency across applications, the following common actions are identified for pushbuttons.

**Action** -- A pushbutton that immediately performs the action implied by its text. An application should define the pushbutton text so it will be meaningful to users (e.g., Exit).

**OK** -- A generic pushbutton that causes the application to accept any changed information in the window or dialogue box. When users select this pushbutton, the window or dialogue box is closed.

**Apply** -- A pushbutton that causes the application to accept any user changes in dialogues that set properties. After users select this pushbutton, the dialogue box should remain open and display the accepted changes for users. This allows users, if they want, to change the properties again, without going through the selection process that displays that dialogue box.

***Reset*** -- A pushbutton that cancels any user changes that have not been submitted to the application. It also resets any changed fields to their initial values and displays them for users. This allows users to make changes again without going through the selection process that displays that dialogue box.

***Cancel*** -- A pushbutton that closes the dialogue box without performing user changes not committed. The Cancel action does not affect previously committed changes.

***Help*** -- A pushbutton that displays, if available, contextual help for the item the cursor is on. If contextual help is not available, it displays help for the entire dialogue box, or at least a help index.

It is recommended that an application first position any application-specific pushbuttons, including *OK*, *Apply*, and *Reset*, and follow them with *Cancel* and *Help*.

### 2.2.6.13 Logo Window

U.S.A.I.D. applications should display a logo window the first time users start the application. The logo window is also accessible by selecting the About... choice in the Help Menu pull-down. The logo window is a modal dialogue box that is displayed within the primary window of the application.

The logo window should be large enough to contain the required information and any optional information, but it should be smaller than the size of the work area of the primary window.

**Figure 2.22** is an example of a logo window for an U.S.A.I.D. application.

For all U.S.A.I.D. applications, the logo window must contain the following information centered in the window:

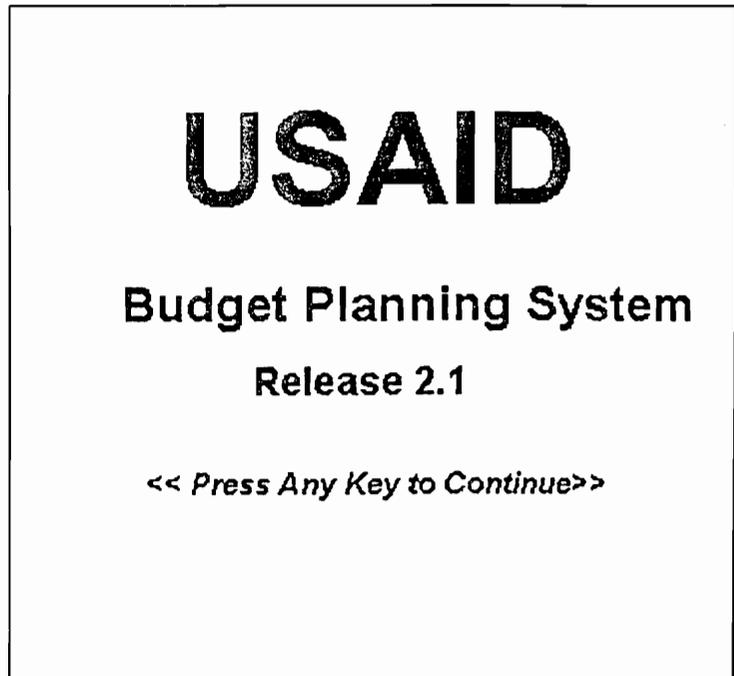
U.S.A.I.  
D.  
Organiz  
ation  
name

Purpose  
of this  
system

Applicat  
ion  
name

Applicat  
ion  
version  
number

Applicat  
ion logo,  
if appropriate



**Figure 2.22 Logo Window**

Copyright statement, if appropriate.

As an option, your application may display relevant information about the application, such as the required memory or hardware configuration.

The application may choose to display the logo for an indefinite time until users select the OK pushbutton, or the logo may be removed automatically after an application specified time period (e.g., 5 seconds).

The About... choice in the Help menu pull-down of the primary window of an application allows users to view the logo window of the application.

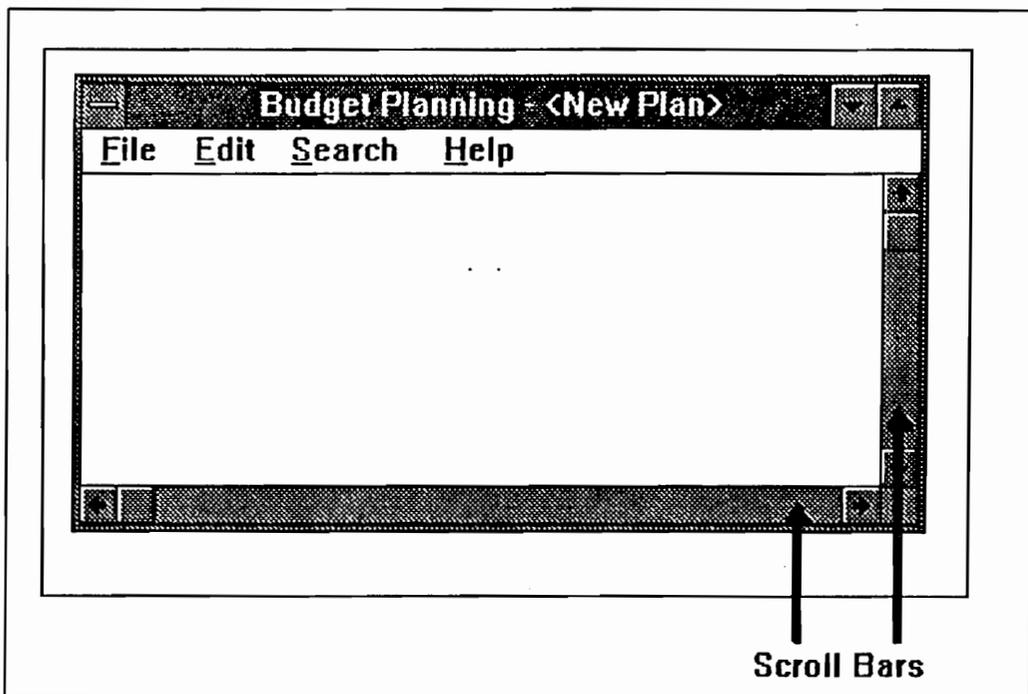
Users select the OK pushbutton in the dialogue box to remove the About ... choice logo window.

## 2.2.7 Scroll Bars

Scroll bars allow users to see information that is not visible on the screen. If an application often has information for users that is too large to fit on the screen (e.g., multiple page reports), applications need to provide users with the ability to scroll information. In a windowing environment, scrolling capabilities are even more important because several windows can appear on the screen simultaneously, limiting the space available for the information in each window. Also, in a windowing environment, users have the ability to change the size of windows, limiting the available presentation space even more.

A window can have either a vertical or horizontal scroll bar or both. The vertical scroll bar appears on the right side of a window. A vertical scroll bar should be the height of the scrollable portion of the work area. The horizontal scroll bar appears at the bottom of a window. A horizontal scroll bar should be at least half the width of the scrollable portion of the work area.

The unused space adjacent to a horizontal scroll bar may be used to present navigational information, icons, or other relevant information. For example, in a report display application, some of the unused space may be used to show the page



**Figure 2.23** Vertical and Horizontal Scrolling Bars

number of a document. **Figure 2.23** is an example of vertical and horizontal scroll bars.

A scroll bar consists of a scroll area with a slider box inside and an arrow in a square box at each end.

The slider box represents the position and size of the visible information in relation to all the information that is available. For example, if the slider box is one-third of the way down from the top of the scroll bar, the portion of information that users see is one-third of the way down from the beginning of the displayed information.

A slider box may also vary in size according to the amount of information visible to users. For example, if an application starts out with an empty work area, the slider box may fill the length of the scroll bar. If the total information available for viewing is greater than what is visible in the work area, the slider box may reduce in size in proportion to the total information that is visible.

The scroll bar arrows show users the direction in which scrolling is available. An arrow is used to scroll a fixed amount of information, as defined by an application.

If users cannot scroll in a particular direction, the application reduces the contrast (grays out) of the scroll bar arrow that represents the unavailable scrolling direction. This provides a visual cue to users that there is no more information to view in that direction. The slider bar should also be against that margin.

An application should use scroll bars in all sizable windows when the object contained in the window will not be completely displayed as the window size changes. Even if no information can be currently scrolled, inactive (grayed) scroll bars should be displayed. If a new object is being created, the scroll bars serve as a visual cue indicating that the object is not limited to the size of the window.

An application should present scrollable information from top-to-bottom (vertical scrolling) rather than left-to-right (horizontal scrolling) if there is a choice. Some applications, such as wide reports require both horizontal and vertical scrolling.

### **2.2.7.1 Interaction**

Users can scroll information using either the mouse or the keyboard.

#### **Mouse**

There are three types of scrolling available with a mouse: using the arrow, page, and direct positioning of the slider bar.

Users scroll information incrementally by clicking on the scroll bar arrow at either end of a scroll bar. The scrolling increment is set by the application. For text, the scrolling increment is usually one line vertically or the average width of one horizontal character.

A click above or below the slider box in the scroll area scrolls information a logical page at a time in the appropriate direction. For vertical scroll bars, the logical page is usually the height of the scrollable portion of the work area minus one unit of information, such as one line of text. Similarly, for horizontal scrolling, the logical page is usually the width of the scrollable portion of the work area minus one unit of information, such as one column on a report. An application should leave at least one unit of previous information visible as a reference point for users.

Direct positioning allows users to scroll to a specific location in the information. Users drag the slider box to the point in the scroll bar that is directly proportional to where the desired information is in the list.

### **Keyboard**

Users can scroll incrementally with the keyboard by using *the arrow keys* while the cursor is at a scrolling boundary. The scrolling increment for the keyboard should be the same as for the mouse. Users can perform vertical page scrolling by using the *PgUp* and *PgDn* keys and horizontal scrolling by using the *Ctrl + PgUp* and *Ctrl + PgDn* keys.

---

## **2.3 Application Interaction Techniques**

The application developer may wish to combine some of the individual elements and controls described in the preceding sections to provide the easiest communication with the user. In addition to combining elements, positive user feedback should be provided. User feedback may take the form of highlighted items, messages, a contextual help system, and a tutorial system, if necessary.

### **2.3.1 Combining Elements**

Controls may be used with each other in a way that is contextually appropriate and allows an application to provide an interface that is easier for the user to understand. Combining controls can help users to become more proficient when

they make selections or enter information. Controls should react in a way that provides feedback to users and puts them in control.

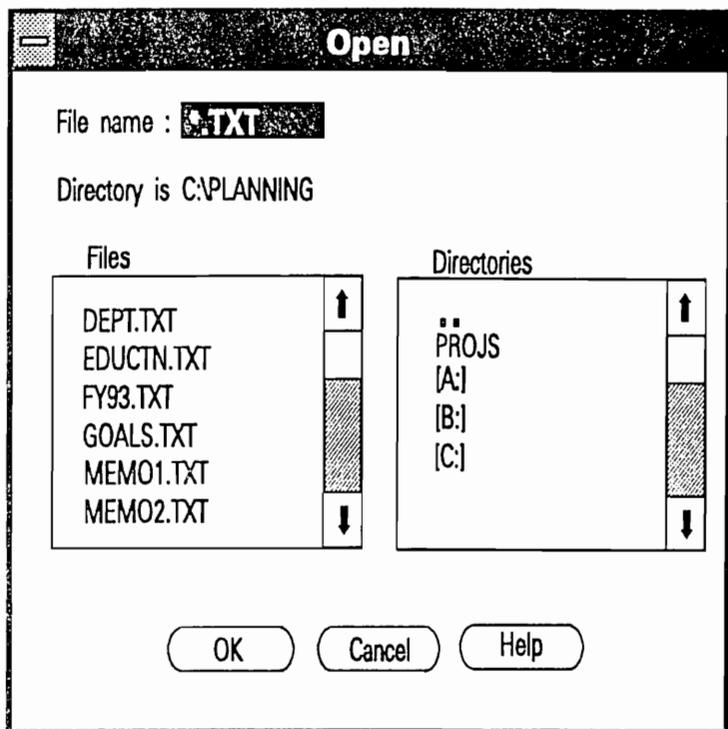
### Combining Entry Fields and List Boxes

It can help users if an application provides more than one way to complete a dialogue. **Figure 2.24** shows the Open dialogue box that contains one entry field and two list boxes. Users can complete the File name entry field by typing a file name into the entry field or by selecting a file name from the list box.

### Controls May Affect Each Other

Controls in dialogue boxes may affect each other to provide feedback to users and provide better user understanding.

In **Figure 2.24**, the second line in the work area indicates that the selected directory is C:\PLANNING. The contents of that directory are displayed in the scrollable list box under the heading Files. If users change to another directory by double-clicking on a choice from the list box under the heading Directories, the chosen directory appears in the field. The multiple controls, affecting each other, have provided this updated information to users.



**Figure 2.24** Open Dialogue Box

Also in **Figure 2.24**, the File name entry field displays a search string for the files listed in the Files list box. If users want to open a file in the listed directory and path and they know the name of the file, they may replace the search string with a valid file name; they will immediately see the file they want. If users do not know what file they want to open, they can further qualify the search string or even

change the displayed directory. In that case, all affected entry fields and list boxes will display the updated information if users double-click or select the OK push-button. Therefore, the Files list box is sensitive to the entry field search string, as well as the Directories list box.

Displaying information based on the current context provides users with immediate feedback. It also offers users several ways to complete a task, such as locating a file, and puts users in control of the dialogue.

Another way to provide context-sensitive feedback is to gray unavailable choices within a dialogue based on the current context of the choices. An application can help users decide which fields must be completed in a dialogue box by graying fields until their completion is required. For example, Figure 2.25 shows a Print dialogue box with a set of two radio buttons. The All radio button is selected; the Partial is not. Partial has two entry fields associated with it, From and To. These two entry fields are grayed to show they are unavailable.

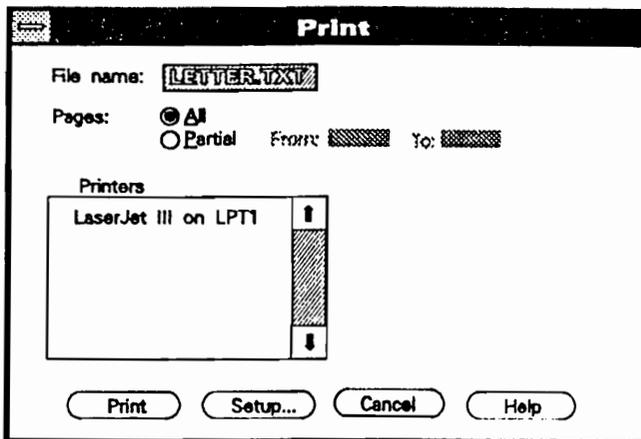


Figure 2.25 Print Dialogue Box

If a user selected the Partial radio button in Figure 2.25, he will see Figure 2.26. The two previously grayed entry fields, From and To, are no longer grayed; the user can now request the starting and ending range for printing.

## 2.3.2 Informing the Users

### 2.3.2.1 Feedback

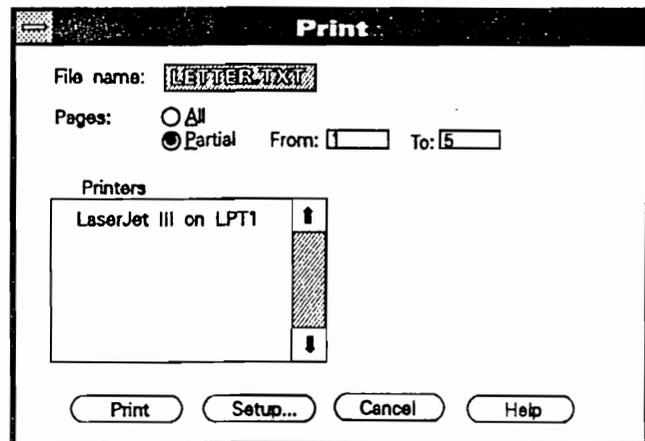
This section describes the *graphic* and *audible* feedback mechanisms and the messages that are used to support communication between users and the application.

Graphic feedback is used when users request that the application perform an operation, but the request cannot be satisfied immediately. When this occurs, the application may be unavailable for user input and should inform the user that this situation exists.

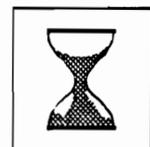
Two standard methods tell users that an application is unavailable: changing the mouse pointer to an hourglass, and using a progress indicator. An application designer may create other visual methods of feedback to fit best the specific situation.

#### Hourglass Pointer

The application may change the shape of the mouse pointer to an hourglass when the computer is performing simple operations, such as opening or saving a file. This shows users that the function they requested is in process. It also reminds users that user requests cannot be accepted until the hourglass pointer changes back to its previous shape. **Figure 2.27** is an example of the hourglass pointer.



**Figure 2.26** Printer Dialogue Box



**Figure 2.27**  
Hourglass Pointer

### Progress Indicators

A progress indicator may be displayed to keep users informed of the current status of complex user requests. Actions such as downloading files or copying several files may take a long time for the computer to complete.

Figure 2.28 is an example of a progress indicator.

A progress indicator is a modal dialogue box. At the top of the dialogue box, information tells users about any exceptional conditions the computer detects during the execution of the tasks or any other relevant progress information. A rectangle below the text shows the progress of the task toward completion. A sliding scale with color fill moves along the rectangle to indicate the progress of the task. If possible, a scale marked in application-defined increments to signify relative completion may be used below the rectangle. If it is not possible for the application to assess the progress toward completion, the rectangle and scale may be omitted from the dialogue box. An active timer or slow blinking cursor below the rectangle may be used to indicate elapsed time since the start of the task.

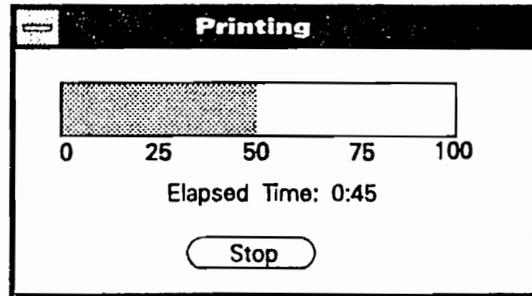


Figure 2.28 Progress Indicator

A *Stop* or *Cancel* pushbutton at the bottom of the window allows users to stop the task. When the application stops a task in response to the *Stop* pushbutton, it must ensure that the integrity of the system is maintained. The application must decide if it is appropriate to complete the requested processing for the current task before stopping or to restore the current object to its prior state. If the nature of the task can cause objects to be destroyed or become unusable if the process is stopped before completion, the *Stop* pushbutton should be shown in reduced contrast (grayed) as a visual cue to users.

**Audible feedback** is a sound from the computer that either warns users of an error condition or draws their attention to a certain situation or specific information. Because users may elect to turn off the sound, the application should not generate the sound unless the setting for it is turned on.

Audible feedback may be used for the following situations, as well as for others that may be appropriate for an application:

When an application displays warning and action messages.

When users type a character that is not a valid mnemonic in selection fields.

When users attempt to select an unavailable (grayed) choice.

When users type invalid data into an entry field (e.g., non-numeric data into a numeric field).

### 2.3.2.2 Messages

Messages are feedback that tell users that something has happened because of a request they made. The application can use a message box or a dialogue box to provide a message to users. A dialogue box can be used to provide more capability than a message box. Messages should be modal with respect to the application. Message type may fall into one of three categories: *Information*, *Warning*, *Action* or *Question*.

A message box consists of an icon, text that concisely describes the conditions, and one or more pushbuttons. Message boxes use the Application Name in their window title.

#### Information Message

An information message tells users that a computer task is or has performed normally. **Figure 2.29** shows an example of an information message. The icon for information messages is a block capital *I* enclosed in a circle and is used in all information messages. Descriptive text appears to the right of the icon. Information messages have an OK pushbutton so users can acknowledge the message. When users select OK, the message is removed.

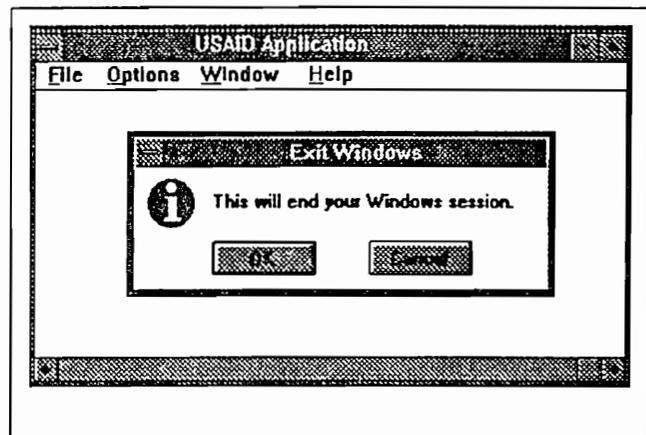


Figure 2.29 Information Message

A Help pushbutton is optional. No audible sound is required when an information message is displayed.

## Warning Message

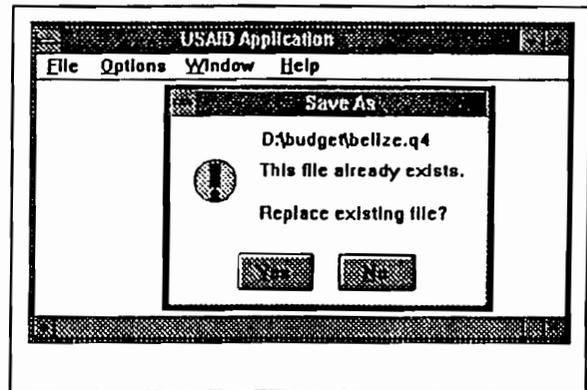
A warning message indicates that a potentially undesirable situation could occur. Users need to respond to the message to continue. However, corrective action may be required later to avoid an error situation. **Figure 2.30** is an example of a warning message. The icon is an exclamation point (!) enclosed in a circle and is unique for all warning messages. Descriptive text that allows the user to evaluate the message appears at the right of the icon.

In any application that requires a user to explicitly save changes to an object, then an Open, Close or Exit should generate a message that changes will be lost. A warning message should be generated that provides the option to save the changes. An example message is - *(ObjectName)* has changed. Save current changes? The appropriate pushbuttons are Yes, No and Cancel. Cancel places the user back in the application prior to the Open, Save or Exit action was requested.

Warning messages may contain the following pushbuttons:

OK (used with Cancel) -- Users select OK to tell the application that they received the message and want to continue. When users select this pushbutton, the message is removed.

Cancel -- Users select Cancel to remove the message. When users select Cancel, the application does not take any action except to remove the message.



**Figure 2.30** Warning Message

Yes (used with No) -- Users select Yes to give a positive response to a question.

No -- Users select No to give a negative response to a question. A Cancel pushbutton can be provided if Cancel provides an action that is different than the action provided by No.

A Help pushbutton is optional. An audible beep should sound (if turned on) when the application displays a warning message.

### Action Message

An action message tells users that an error or exception condition has occurred. Users must perform an action to correct the situation.

Action messages are used in situations from minor application-related conditions that stop users from continuing with the current dialogue to serious system related conditions that stop users from continuing to work with any application in the system.

The text of the message should indicate the severity of the error condition and should suggest any available action that will correct the situation.

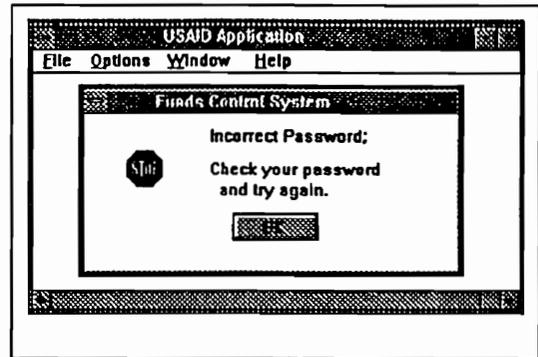


Figure 2.31 Action Message

**Figure 2.31** is an example of an action message. The stop-sign icon is unique for all action messages. Text that describes the condition that users must correct appears at the right of the icon.

Action messages may contain the following pushbuttons:

Retry -- The Retry action assumes that users have taken some action to correct the error situation and is used for device error messages. Selecting this pushbutton directs the application to attempt to complete the process that caused the message.

Cancel -- When users select Cancel, the system does not take any action except to remove the message.

A Help pushbutton is optional. The application should provide an audible beep (if turned on) when displaying an action message.

### Question Message

A Question message indicates that the application needs further guidance before performing an action, or a user has attempted to do something prior to selecting a

object. If a user attempts to save an empty file, or change the characteristics of existing text before identifying the text, the application may generate a Question message.

A Question message may be used to determine if a parameter change is permanent, or temporary, if the user wants to continue, or if the user wants to Cancel the request.

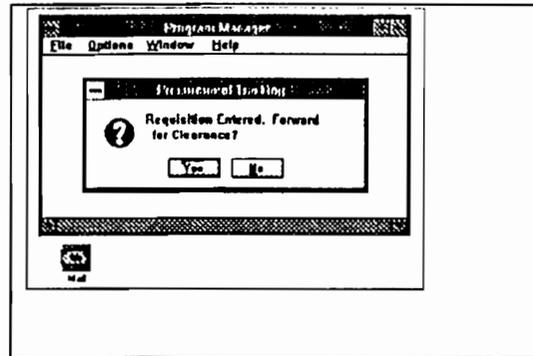


Figure 2.32 Question Mark

The text of the message should indicate what actions may be taken by the user to achieve the results that they have requested. Questions can be confusing to a user, so it is better to provide a declarative statement to describe the situation, and a question to clarify what the users options are.

**Figure 2.32** is an example of a Question message. The question mark icon (?) is used to indicate that the application user must make a decision. Text that describes the condition appears to the right of the icon.

Question messages may contain the following pushbuttons:

OK (used with Cancel) -- Acknowledges the message or allows the program to perform the action indicated.

Cancel -- When Cancel is selected, the system doesn't take any action, except to remove the message.

Yes (used with No) -- Users select Yes to give a positive response to a question.

No -- Users select No to give a negative response to a question. A Cancel pushbutton can be provided if Cancel provides an action that is different than the action provided by No.

### 2.3.2.3 Help

Providing help, within an application, allows users to easily and quickly access additional information about choices, fields, or how to proceed with the application.

The purpose of help is to provide online assistance. Help information is not meant to replace a tutorial system, but to supplement it. Tutorials teach new users how to use an application; help information assists users in recalling how to use an application.

There are several types of help to assist users in completing a specific action. *Contextual help* is based on where the cursor is located in the window. There is also *extended help* (general help about the application window), *a listing of keys* and their actions, an *index of help topics*, and *information about how to use help*.

An application may also provide access to a tutorial and may reference phrase help.

Users request help from an application by pressing the F1 key, selecting the Help pushbutton, or choosing one of the help actions from the Help menu bar pull-down. Requesting help displays a window that contains the information users requested.

After a help window is displayed, users can access additional help by accessing its Help menu bar pull-down.

Users end help by choosing Close from the system menu bar of the help window.

A help window may be removed if users remove the application window to which a help window is related.

Several types of help information should be provided for users. The application determines the content of each type of help information. Following are some general descriptions and guidelines:

*Contextual help* - provides specific information about the item on which the selection cursor is positioned. Contextual help information should describe the purpose of the item for which help is requested and tell users how to interact with that item. Contextual

help is accessed by moving the selection cursor to the item and pressing the F1 key or selecting the Help pushbutton.

The application must assign the F1 function key so users can access contextual help using the keyboard.

*Help for help* - Provides information about how to use the help facility. It identifies the levels of help and the other resources available and whether a tutorial can be accessed from help or is available elsewhere.

*Extended help* - Provides information about the contents of the application window from which users requested help. Extended help tells users about the tasks they can perform in the window from which they requested Extended help.

*Keys help* - Provides information on the key assignments of the application. This includes common key assignments as well as application or task specific keys.

*Help index* - Provides an alphabetic list of all the help index entries for the application.

*Tutorial* (optional) - provides access from the current window to a tutorial, if one is available.

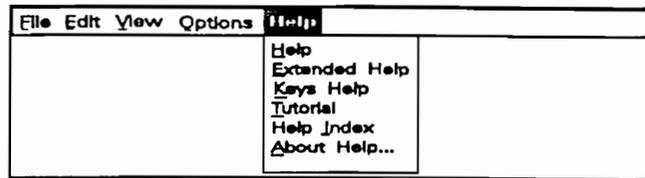
*Reference phrase help* - additional information about application defined words or phrases within a help window. The reference phrases are highlighted by displaying them in a different color.

To select a reference phrase with a mouse, users place the mouse pointer on the phrase and click the select button. To select with the keyboard, users press the Tab key to move the selection cursor to a reference phrase and select Enter.

To request help from the menu bar of the application, users select the Help menu bar choice and select the choice they want from the Help pull-down. A window is displayed containing the help information requested.

To promote consistency, application developers should provide the same help pull-down for all applications. The pull-down recommended to access the help facility is shown in **Figure 2.33**.

Each choice in an application window Help pull-down is followed by an ellipsis to tell users a help window appears when they select any Help pull-down choice.



**Figure 2.33 Pull Down Help Facility**

### 2.3.2.4 The Help Window

The help window title contains the title of the application, followed by Help. For example, if an application title is **U.S.A.I.D. APP**, a help window for that application has the title **U.S.A.I.D. APP - Help**.

At initial display, a help window is placed where it is completely visible on the screen and does not overlay any of the application window. Sometimes, however, the help window cannot fit on the screen without covering the application window. In that case, place the help window to overlay the application window in the position that covers the least amount of information in the application window. After displaying the help window, users may change the size and position of the window.



## 3.0 Text Presentation of Common User Interface

This chapter defines the text subset of the common user interface. A text based user interface may be required when enhancing older systems that are restricted to or must interface with non-programmable terminals (e.g. 3270). The techniques of a graphic interface may be used, in a character environment, to support the common user interface requirements.

This chapter includes only the elements that are fundamental or recommended for the text CUI as it may appear on a non-programmable terminal. If application designers can make use of the features of a programmable workstation, or want to extend the user interface described in this section (for example, using graphical equivalents of interface elements, or if you want to support a mouse), implement the extensions as similarly as possible to their descriptions in the graphical CUI. The text CUI supports and requires an *object oriented process sequence*.

The text interface CUI is designed for *ease of use* rather than for *ease of learning*. Users learn by exploring the system after some initial training. Once users have learned a few basic things about the interface, such as how to move the cursor to a choice and select it and how to get help, they should feel free to explore and continue to learn about an application.

An interface designed for *ease of learning* requires that the application lead users step by step through their activities and present information frequently to tell users what to do next and how to do it. While new users may find this very helpful, once they know how to use the applications, the continued presentation of tutorial information may become an annoyance. An example of an interface oriented to ease of use is a "walk up and use" application, such as a locator or bus schedule in a hotel lobby.

### 3.1 Presentation Elements of Text Interface

Presentation is the visual aspect of the interface; it is what users see on the screen. The main visual components of the user interface are the following:

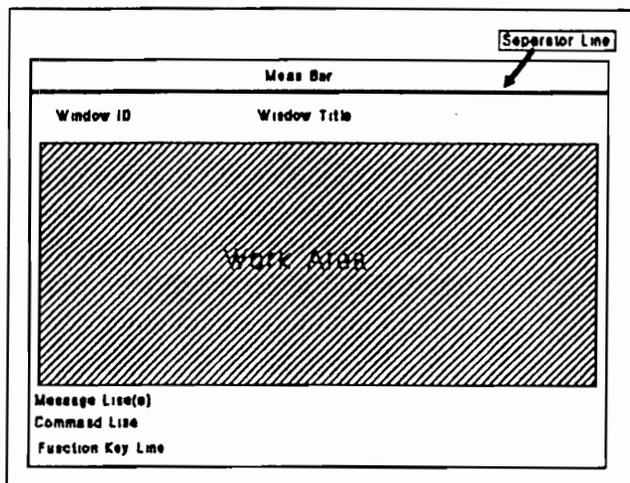
The *Primary Window*, which occupies the entire viewing screen and represents the users' view of the activities and selections currently available.

*Secondary or child windows or panels*, which are presented within the Primary Window.

*Pop-ups* which contain information or data for users to work with.

*The cursor* indicates the area of the screen that the user is focused on.

Some of the visual elements that are common to most text CUI applications are illustrated in **Figure 3.1**. The space between the window title separator and the message area is the work area. **Section 2.1 Basic Windows Concepts**, contains a more detailed discussion of window organization.



**Figure 3.1 Common Visual Elements**

#### 3.1.1 Window or Panel

Windows are presented as rectangular areas within the Primary Window and as pop-ups. The concept of presenting a window in a window is illustrated in **Figure 3.2**. Information and application objects are presented in the work area of a window. Application actions are presented in the menu bar or function key area of a window. Every window in the primary window of a text application has a menu bar. Windows in pop-ups do not have menu bars. The window in the primary window is the main focal point for the users' work activity. Windows presented in pop-ups supplement the dialogue that is occurring in the primary window.

Displayed in a window are standard interface components that provide a consistent way to present information to users. When users become familiar with the interface, they can quickly identify components of the interface and feel comfortable when using new applications that support these components.

The window elements that each application must consistently support are the *window identifier*, *window title*, *window area separator*, *work area*, *message area*, *command area*, and *function key area*. Following is a definition of each of these elements.

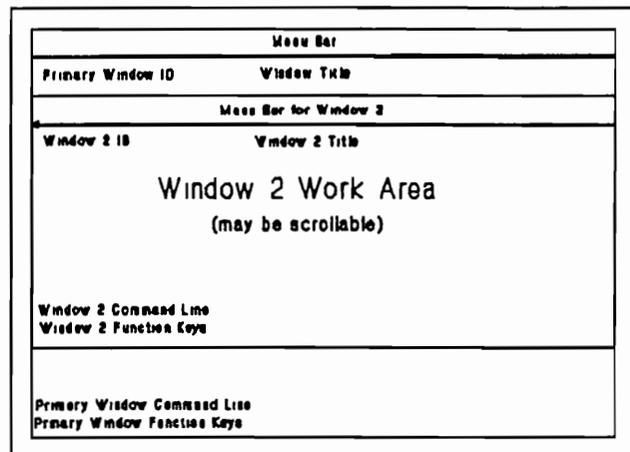


Figure 3.2 Primary and Secondary Windows

### 3.1.1.1 Window Title

In a primary window, the window title is one line of protected text. The window title contains the application name and, optionally, the file or object name. Other optional information may be included on this line. The title is centered, if possible. Display of date and time is optional, but if used it should be located flush right, on the same line as the title.

### 3.1.1.2 Window Identifier

The window Identifier is an alphanumeric, character-string identifier. Each application may optionally use window Identifiers. If they are used, they should be used consistently throughout the application. Refer to the various U.S.A.I.D. application standards for naming conventions.

### 3.1.1.3 Window Area Separators

These are used to separate the various window areas for clarity. A blank line, solid line, or dashed line may be used. Typically a solid or dashed line separates the Menu bar from the Window Title line and is used to define the limits of pull-down areas and pop-ups. The message line, which is normally blank, may be used as a separator for the command line.

#### **3.1.1.4 Work Area**

The work area is the space between the window title separator and the message area. It is the users' workspace and the focus of their attention. If the logical window is too large to be presented completely in the physical (screen) window, then all or part of the work area is made scrollable.

##### Scrolling information

Scrolling information is a visual cue to users that more information is available but is not currently visible and that the information can be brought into view using the scrolling keys. Scrolling information also tells users the direction in which the additional information is available.

##### Selection fields and selection lists

Selection fields and selection lists are sets of related choices. They may be either single-choice or multiple-choice. Selection fields are not scrollable and contain a fixed set of choices. Selection lists typically vary in content or number of choices.

##### Entry fields

Entry fields are spaces into which users type information.

##### Protected text

Protected text is information presented by the application that users cannot modify. The date and time display (1/07/91 at 12:12) is an example of protected text.

##### Headings and field prompts

Headings and field prompts identify entry fields, selection fields, selection lists, protected text, or related groups of those elements. The application should provide one of these types of identifiers for each of these elements or each group of them, unless there is only one element or group in the window and the window title is sufficient as an identifier.

### **3.1.1.5 Message Line**

The application must provide a message line and locate it immediately above the command area. If there is no command area, locate the message line immediately above the function key area. Messages located in the message area should be removed when the window is processed.

There may be three types of messages:

Information

Warning

Action.

When the application needs to provide a means within the message for users to respond to the message (for example, by providing a selection or entry field), or when it is especially important to get the users' attention, the message may be displayed in a message pop-up, or the message may be enhanced with visual or audible cues.

### **3.1.1.6 Command Area**

The application should provide a command interface if system designers want to allow users to issue system commands without leaving the application, or to type application commands into the command line. This is helpful for experienced users who prefer to enter an entire command at once.

CICS applications should use a Command area.

Every valid command equates to a process that may be reached via menu choice.

If the application has a command interface, users issue commands through the command area. A command line may be located in either of two places:

In the primary window immediately above the function key line.

In a pop-up, if the application needs to maximize the available space in the primary window, and if commands are expected to be used only occasionally.

If present, the command line should begin with the following field prompt:

Command =>

### 3.1.1.7 Function Key Line

The function key line appears at the bottom of the window to present common actions and application-defined actions that users can request by pressing function keys.

---

## 3.2 Window Title

The window title should show the program Identification and screen title. The program Identifier is an eight character name of the application that created this screen. The Window title is centered in the window title line. If the window has an menu bar, the window title appears on the next line below the menu bar separator line; otherwise it appears on the top line.

A pop-up window title identifies the function of the pop-up window.

---

## 3.3 Window Identifier

The window identifier is the eight character name of the screen. Left justify the window identifier on the same line as the window title. If used, the Window Identifier must be used on all windows. The option of displaying or not displaying may be given to the user through a function key or Menu bar pull-down.

---

## 3.4 Window Area Separators

Use separators to define the boundaries of areas of a window with different information. They can be used

Between the menu bar and the window title

Between the window title and the work area

---

Between different window areas that scroll.

A blank line, solid line, or dashed line may be used as a separator between window areas.

---

## 3.5 Work Area

The Work Area is the part of the primary window between the window title separator and the message line. It is the area between the top of the screen (menu bar) and bottom of the screen where the users get their work accomplished. The Work Area is usually the main focus of user interaction with the application. For some U.S.A.I.D. budget and office automation systems the work area may be customized, such as customized forms fill-in screens. For some types of applications the work area may be presented completely blank (e.g. word processors).

### 3.5.1 Instructions

Instructions tell users how to interact with a window or application. Protected text may be used elsewhere in the window to give users additional instructional information. The top of the work area should be used for instructions whenever possible.

User feedback during testing will help designers determine if they need to include instructions. If the system is following a known process or metaphor, fewer instructions may be required. A well designed and accessible Help system can also alleviate the need for instructions.

### 3.5.2 Headings

Headings identify columns and groups of related items. Constants and labels should be normal intensity. **Figure 3.3** is an example of group headings and field prompts.

#### 3.5.2.1 Column Headings

Column headings identify columns of entry fields, selection fields, selection lists, action lists, and protected text when all the items in the column are the same type.

---

**3.5.2.2  
Group  
Headings**

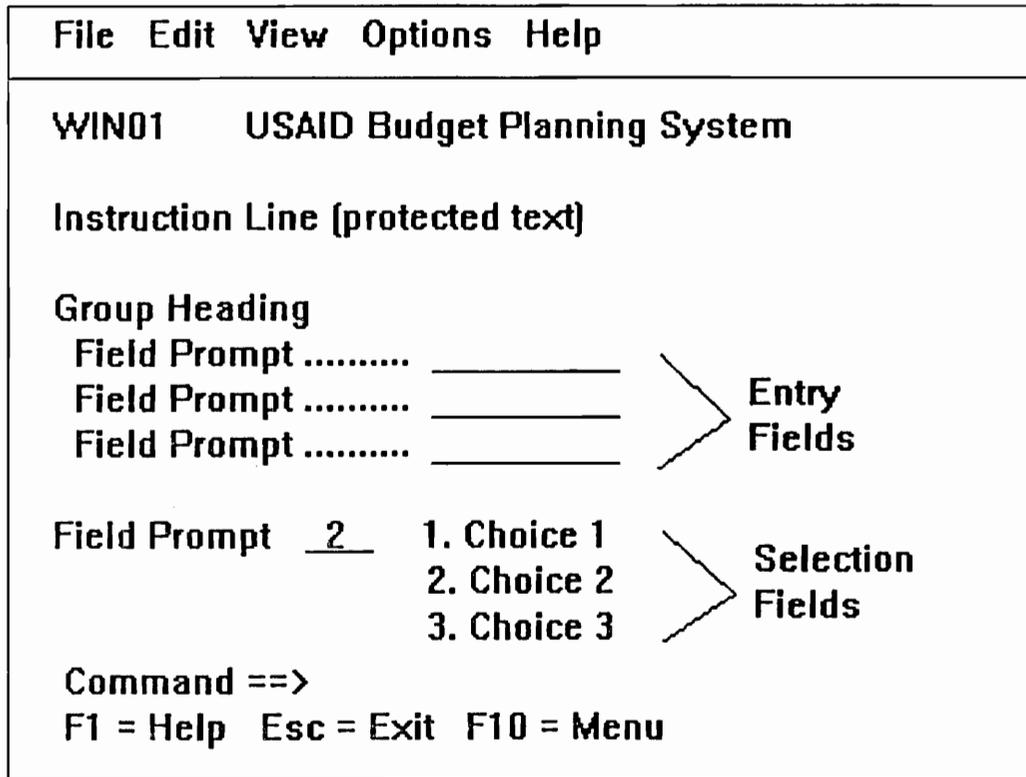


Figure 3.3 Group Headings and Field Prompts

Group headings identify related groups of entry fields, selection fields, and protected text. They may be used concurrently with field prompts when group headings identify a group of fields and field prompts identify the individual fields.

**3.5.2.3 Field Prompts**

Field prompts identify selection fields, entry fields, and variable output information. Data fields and special fields should be highlighted.

Use field prompts concurrently with group headings when the group headings identify groups of fields and the field prompts identify individual fields within a group.

Field prompts are located to the left of the fields they identify. They are left-aligned.

If the field prompts follow group headings, the field prompts should be indented under the group headings. When group headings are not used, indent field prompts under instructions.

Use leader dots (. . .) to connect field prompts and fields, so users can easily move their eyes from one side of the screen to the other.

Field prompts for variable output information should also be followed by a string of leader dots. Replace the last leader dot with a colon (:) to indicate that what follows is protected text.

### 3.5.3 Descriptive Text

Where it would be useful to users, additional descriptive information may be included about the entry field, in addition to the field prompt that identifies the field.

Descriptive text is placed after the end of the field. It should be brief, protected text. It should provide information about which values can be typed into the entry field. If the list is small, the actual values may be displayed as descriptive text, separated by commas and blanks, for example:

Graph type . . . . \_\_\_\_\_(Scatter plot, Bar, Line, Pie).

A range of values also may be indicated as descriptive text. For example:

Pages to print. . . . \_\_\_\_\_ (1 - 999).

### 3.5.4 Protected Text

Protected text may only be viewed, not selected or changed. Protected text may be text in a file, numeric data, help information, a message, or variable output information.

### 3.5.5 Scrolling

Users can scroll an entire panel or areas of a panel using either cursor-independent scrolling or cursor-dependent scrolling.

The following rules apply for both scrolling techniques:

If a panel has more than one scrollable area as an application option, put a separator between scrollable areas.

The menu bar and pull-downs from the menu bar are not scrollable. If necessary use more than one horizontal line for the menu bar.

Each scrollable area should have scrolling arrows that indicate to users the position of the information they are viewing in relation to the boundaries of the information. Designers may also place a visual indicator at the boundaries of the information.

A panel area stops scrolling at the boundary of the information. When users reach this boundary, de-activate panel area scrolling in that direction.

Scrollable information does not wrap.

### 3.5.6 Scrolling Actions

The scrolling actions are:

Backward--displays information above the currently visible information in the panel area.

Forward--displays information below the currently visible information in the panel area.

Left--displays information to the left of the currently visible information in the panel area.

Right--displays information to the right of the currently visible information in the panel area.

There are specific function key assignments for these four scrolling actions.

### 3.5.7 Cursor Independent Scrolling

With *cursor-independent* scrolling, or *page* scrolling, the information is scrolled in fixed increments regardless of the position of the cursor when users request one of the scrolling actions. As an application option, the cursor may be kept stationary on the screen or in the information. If the cursor remains stationary in the information and if the choice it was on scrolls out of view, it is an application

option either to stop the cursor at the boundary of the panel area or to move the cursor to the first visible choice or entry field in the panel area.

The increment, or amount, of information scrolled when users request one of the scrolling actions varies with the size of the visible area that is scrollable. The following may be used as guidelines for how far to move a scroll with page scrolling:

The visible area minus one item (for forward and backward scrolling)

The visible area minus one column (for left and right scrolling).

Other scrolling increments may be used, such as the full, visible area or a part of the visible area, for example, one-half or one-third of the visible area.

### 3.5.8 Cursor-Dependent Scrolling

With *cursor-dependent* scrolling, the initial position of the cursor determines the extent of the scrolling when users request one of the scrolling actions:

Backward--reposition the information so that the item containing the cursor is at the bottom of the scrollable area.

Forward--reposition the information so that the item containing the cursor is at the top of the scrollable area.

Left--reposition the information so that the column containing the cursor is at the farthest right column of the scrollable area.

Right--reposition the information so that the column containing the cursor is at the farthest left column of the scrollable area.

When the cursor is at a vertical or horizontal boundary and users request scrolling, cursor-independent (page) scrolling is performed.

The application determines what quantity constitutes an item or a column, based on what is most appropriate for the environment in which scrolling is used.

Provide for cursor-dependent scrolling when users might want to position a specific item at a vertical or horizontal boundary (for example, if users might want to position a list of items so a specific item is at the top of the list).

### 3.5.9 Scrolling Indicators

Scrolling indicators tells users that more information exists outside the visible panel area, the position of the visible information in relation to the total amount of available information, and which direction to scroll the work area to display the unseen information. Scrolling information may appear in three forms:

Scrolling arrows -- Use arrows if available. If actual arrows are not available, use greater than (>) or less than (<) to indicate right or left scrolling and plus (+) and minus (-) to indicate down or up scrolling.

Textual scrolling information--may be used in combination with scrolling arrows.

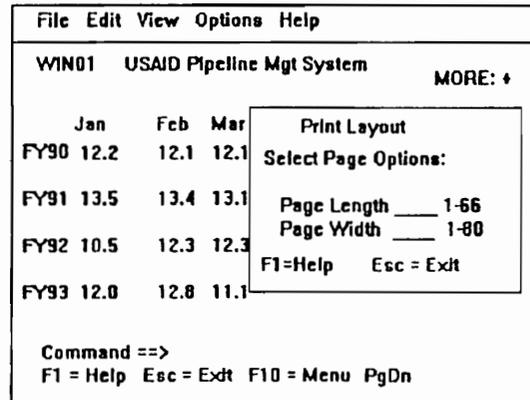


Figure 3.5 Pop-up with Entry Fields

### 3.5.10 Scrolling Arrows

Scrolling arrows indicate that additional information exists outside the visible panel area and shows users which direction to scroll to see that information.

Figure 3.4 is an example of scrolling usage.

Locate scrolling arrows on a line above the panel area to which they apply and below the textual scrolling location information, if present. Right-justify the arrows. If actual arrows cannot be displayed by some terminals, the application may use the following alternate characters for the scrolling arrows, in the following format:

More: << - + >>

<< indicates there is information to the left of the visible area.

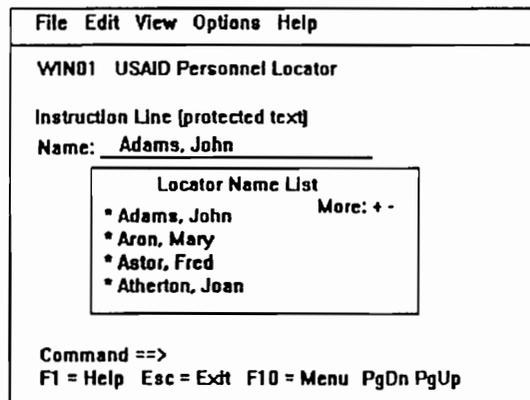


Figure 3.4 Scrolling Example

- indicates there is information above the visible area.
- + indicates there is information below the visible area.
- >> indicates there is information to the right of the visible area.

Lay out scrolling arrows with one space between the colon (:) and the first arrow position and one space between the arrows. For example:

More: >>

If the panel can be scrolled in all four directions, backward, forward, left, and right, reserve the entire space for scrolling arrows (the word *More*, the colon, and the four arrows or symbols), even if all the arrows or symbols are not currently used because users cannot currently scroll in all four directions. Maintain the space of the unused arrows or symbols with blanks.

If users currently cannot scroll in any direction, do not display the *More* tag.

Users press the Backward key to scroll up, the Forward key to scroll down. The Left and Right keys are used to view information that is to the left or right of the information that is currently displayed.

### 3.5.11 Textual Scrolling Information

Textual scrolling information may be used in combination with scrolling arrows. *Bottom* is displayed below the scrollable area when users are viewing the end of the information. *More...* is displayed below the scrollable area when users can scroll forward.

Locate *Bottom* or *More...* on the right side of the line below the scrollable area, using the following rules for determining which indicators to use:

If users can scroll forward, display *More...*

If users are viewing the end of the information, display *Bottom* on the right side of the line below the scrollable area.

### 3.5.12 Textual Scrolling Location Information

Optionally designers may include textual scrolling location information along with scrolling arrows for each panel area that scrolls. However, textual scrolling location information is not used alone. For example:

Lines 1 to 20 of 40

or

Columns 1 to 5 of 15

If used, place textual scrolling location information right-justified on a line above the scrolling arrows.

Optionally, the application may allow users to type over the first value (xx) in textual scrolling location information as a way to reposition the list of items relative to the new starting point the users want. This may be implemented by placing the value for xx in an entry field (xx). Textual scrolling location information with an entry field would look like the following:

Lines \_\_\_ 5 to 18 of 254

If the value exceeds the range in either direction, reposition the list at the top or bottom, in the proper direction.

---

## 3.6 Pop-Ups

A pop-up is an area of the screen enclosed by a border that expands on the users' dialogue with the panel in the primary window. A pop-up may occupy a portion of the screen or the entire screen. **Figure 3.5** is an example of a pop-up with entry fields. Pop-ups are associated with underlying panels and appear when the application wants to extend the dialogue in the underlying panel. For example, a pop-up may be used to provide

A help message or an explanatory message

A command line

An entry field, selection field, or selection list.

The application may have a series of overlapping pop-ups.

### **3.6.1 Pop-Up Positioning**

A pop-up is associated with an underlying panel, a pull-down, or another pop-up. A pop-up may extend beyond the boundary of a pull-down or another pop-up that it is associated with.

When a pop-up is related to an item in the underlying window, pull-down, or other pop-up, position the pop-up in a way that overlays the least amount of relevant information.

If a pop-up is related to an underlying panel or another pop-up but not to a specific item or window on it, use offset positioning to locate the pop-up.

Vertically offset the pop-up below the title of the underlying panel or pop-up so the title is visible. Horizontally offset the pop-up to the right of the left boundary of the underlying panel or pop-up.

### **3.6.2 Pop-Up Layout**

Use a solid line border for pop-ups if line characters are available. If line characters are not available, use hyphens or periods for horizontal lines, and capital "I" or colons for vertical lines.

### **3.6.3 Pop-Up Content**

Pop-ups should contain a function key line. Pop-ups also may contain most types of panel elements (entry fields, selection fields, selection lists, action lists, protected text, a message window, or a command window), depending on the need to extend the dialogue from an underlying panel.

### **3.6.4 Pop-Up Use**

When a pop-up is displayed, the pop-up becomes active and the cursor is positioned in the pop-up. Users must finish interacting with the pop-up before continuing with the dialogue in the underlying panel or in another pop-up, unless the pop-up is a help panel.

A pop-up usually appears as the result of an action taken by users. A pop-up may be removed by requesting the Cancel common action. Help should be available for a pop-up.

The Enter action removes a pop-up unless the pop-up invokes another pop-up. In that case, when the last pop-up is completed, all generated pop-ups are removed.

If users move the cursor out of a pop-up using the arrow keys and press the Enter key or a function key, the cursor is returned to the pop-up and no action is processed.

Exit is not supported in pop-ups, except in help pop-ups.

---

### 3.7 Menu bar

The menu bar is the element at the top of a window that consists of a list of choices that represent groups of related actions. Use of the menu bar is optional. A group of actions appears in a pull-down, located immediately below the menu bar, when users request a choice.

The actions typically affect information displayed in the work window or in some way control the users' dialogues with the application.

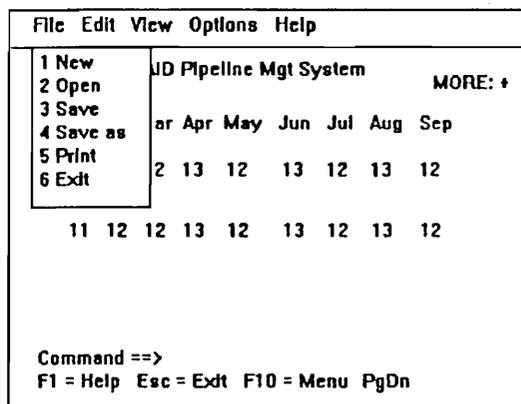


Figure 3.6 Menu Bar Pull-Down

Function keys may be used by the application in place of an menu bar. Application specific uses of the function keys must not conflict with those defined for common dialogue actions.

#### 3.7.1 Menu bar Layout

The menu bar stretches across the full width of a window, regardless of the number of items in the menu bar. Choices are listed horizontally on one or more lines.

#### 3.7.2 Menu bar Content

The menu bar contains choices in the form of single or multiple words.

---

A set of standard menu bar choices are *File*, *Edit*, and *Help*. Any of these standard choices that are valid for the application should be provided in the menu bar. Optionally *View* and *Option* may be included on the menu bar line.

If the standard choices are not adequate for the application, other menu bar choices may be defined. Each menu bar choice must have an associated pull-down. Selecting a choice on the menu bar should always result in a pull-down; it should not perform the action directly.

The last (farthest right) choice in the menu bar should be *Help*.

Exit should always be the last choice in the first (farthest left) pull-down. If standard menu bar choices are used, *Exit* will be the last action in the File pull-down.

Each type of object should have a menu bar that supports the actions for that object type. The application, however, may support multiple object types. For example, a water application may support both lake objects and river objects. In that case, all panels for lake objects would have the same menu bar, containing actions that apply to lakes, while all panels for river objects would have the same menu bar, containing actions that apply to rivers.

### 3.7.3 Selection of Menu bar Options

The cursor should be positioned initially in the blank space immediately to the left of the first choice. As users tab from one choice to another, the cursor is positioned in the blank space immediately to the left of each choice.

### 3.7.4 Menu bar Pull-Down Layout

Locate the menu bar pull-down directly below the bottom of the menu bar. Position the pull-down choices so that the choice entry field is directly below the first nonblank character of its menu bar choice.

If this alignment is not possible because the pull-down is wider than the space allowed to display it, which might happen if the last choice in the menu bar is close to the right edge of the screen, reposition the pull-down horizontally to make sure it is fully visible.

Use a solid line border for a pull-down if solid line characters are available. If solid lines are not available, use hyphens or periods for horizontal lines, and

capital "I" or colons for vertical lines. **Figure 3.6** Menu bar is an example of the Menu bar with the File pull-down activated.

### 3.7.5 Menu bar Pull-Down Content

Both single-choice and multiple-choice selection fields may be used in pull-downs.

Pull-downs should not contain selection lists.

Number choices in single-choice selection fields.

Do not use instructions or a function key line in pull-downs.

Place an ellipsis (...) after each choice in a pull-down that results in a pop-up. For example:

#### 1. Open File....

The Cancel action should be supported in all pull-downs, even though pull-downs do not have a function key window showing ESC=Cancel .

System designers may assign function keys as accelerators to the choices in a pull-down. Display the function keys to the right of the pull-down choices. Any unreserved function keys may be used. Function keys that are used as accelerators do not have to be displayed in the function key window.

Assigned function keys are always active, whether or not the pull-down is displayed. If a mnemonic key is assigned to a pull-down choice, display it in the pull-down.

When users press a key that is assigned to a choice, the associated action occurs even if users have not switched to the menu bar or displayed the pull-down containing the choice. If any pop-ups are associated with the choice, the first pop-up appears when users press the function key assigned to that choice.

In addition to defining standard menu bar choices, the **U.S.A.I.D. CUI Guidelines** also defines standard pull-down choices and accelerator keys for some of those standard pull-down choices.

### 3.7.6 Menu bar Emphasis

Selected emphasis, such as reverse video or hi-intensity is used in the menu bar to provide visual feedback that an menu bar choice has been selected. A selected menu bar choice remains displayed with selected emphasis while its pull-down is visible.

An menu bar choice is never displayed as unavailable, even when all of its pull-down choices are unavailable. An menu bar choice must always be available so that users can request Help on the unavailable pull-down choices.

### 3.7.7 Users' Interaction with Menu bar

To use the menu bar, users first must move the cursor to the menu bar from another window of the panel. This can be done by using the Move-to-action-bar (Menu) function key request. If the cursor is in the menu bar or a menu bar pull-down, requesting Menu again returns the cursor to where it was located in the panel before the previous Move to menu bar was requested.

The cursor may also be moved to the menu bar by pressing the arrow, Tab, or New line keys. None of these keys causes a host interrupt, so the system does not know where the cursor was positioned before the move to the menu bar. When the action is completed and the cursor moves from the menu bar back to the panel window, the cursor is positioned in a location determined by the application.

Moving the cursor to the menu bar with the Move-to-action-bar function key, therefore, has the advantage of telling the system where the cursor was located when users requested the switch.

Menu bar choices are selected by placing the cursor in the space prior to the action request.

Users press the arrow keys or the Tab key to move the cursor in the menu bar. The arrow keys move the cursor one character at a time, and the Tab key moves it from choice to choice.

Users move the cursor and select choices in a pull-down the same way they do in other windows of the panel. When users press the Tab key, the cursor moves from left-to-right, top-to-bottom through the menu bar choices or through pull-down entry fields. While pull-downs are displayed, fields in other panel windows are protected. Users cannot tab to the other panel windows.

If users type over any character in an menu bar choice, the application ignores what was typed and restores the text on the next host interrupt.

If a pull-down is displayed, cursor movement with the Tab key is restricted to the menu bar choices and pull-down entry fields.

If the cursor is moved out of a pull-down using the arrow keys, the cursor is returned to the pull-down when users press the Enter key.

Users switch out of the menu bar by

Pressing the Menu (Move-to-menu-bar) key or the Cancel key.

Pressing the New line, Tab, or arrow keys. If users did not take an action that caused a pull-down or a pop-up to appear, all fields are still active when the cursor returns to the work window.

Users request that a pull-down appear by pressing the Enter key while the cursor is positioned on a menu bar choice.

The pull-down is displayed with the cursor in the choice entry field that precedes the first choice in the pull-down.

Users can request that another pull-down appear by pressing the Tab or arrow keys to move the cursor to another menu bar choice and then pressing Enter.

If users request Cancel from a pull-down, the currently displayed pull-down disappears immediately. If, however, the cursor movement keys are used to leave a pull-down, the currently displayed pull-down is removed when users request another menu bar choice. The newly selected pull-down then appears.

When users request Cancel from the first pop-up that results from a pull-down, the cursor returns to its original position in the underlying panel if users switched to the menu bar using the Move-to-menu-bar action key. If Move-to-menu-bar key was not used, the cursor is positioned in a location determined by the application.

If a pop-up appears, the Tab key moves the cursor from field to field within the pop-up; all other windows are protected.

Users can select choices from a pull-down in one of the following ways:

Typing the choice number

Typing the selection character in one or more choice entry fields in a multiple-choice selection field.

## 3.8 Message Line

Messages are feedback that tell users that something has happened because of a request they made.

### 3.8.1 Types of Messages

U.S.A.I.D. CUI Guidelines defines three types of messages:

Information

Warning

Action.

An information message tells users that a requested activity is being performed normally or has been completed normally.

A warning message tells users that a potentially undesirable situation could occur. Users do not need to correct the condition immediately.

An action message tells users that an exceptional condition has occurred. Users must perform an action to correct the situation.

### 3.8.2 Message Layout and Content

Messages are generally displayed in two forms:

In a message line on the panel

As a pop-up.

Provide a message line on all panels. Locate it immediately above the command line. If there is no command line, locate the message line immediately above the function key line.

If the entire message will not fit in one line, truncate the message. Display the full text when users request Help for the message. If the users' first Help request on a message results in the display of the full message text, a second Help request displays help for the message.

The application may allow users to perform some actions, such as scrolling, without removing the message. A help request should not cause the removal of a warning or action message.

### 3.8.3 Message Pop-up

Display a message panel in a pop-up if the application needs to provide a means within the message for users to respond to the message. For example, the application may provide an entry field or a selection list. The application may also display a pop-up message if it is especially important to get the users' attention.

Panels created for message pop-ups may contain any combination of protected text, entry fields, selection fields, or selection lists. Arrange the elements in message pop-ups as in other panels. **Figure 3.7** is an example of a message pop-up with entry fields.

Message pop-ups that contain entry fields, selection fields, or selection lists must have Cancel in the function key line.

### 3.8.4 Message Removal

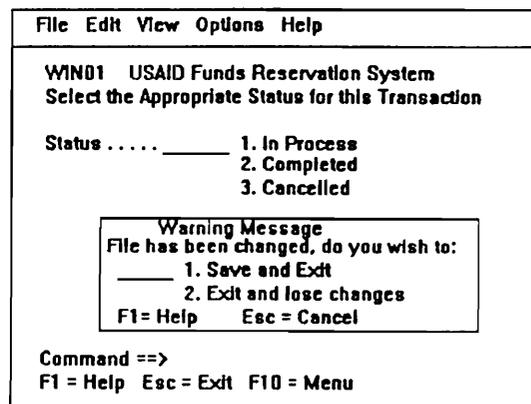
Follow these rules and guidelines for the removal of messages from the message line:

Remove an information or warning message when users take an action that the application can detect, such as pressing the Enter key or a function key, or when the message is no longer needed.

Remove an action message only when the application detects that users have corrected the condition that caused the message to be displayed.

Remove a warning or action message that allows input to the message pop-up when users request the Cancel action or supply any required input and press the Enter key.

If users try to continue the application without correcting the condition that caused an action message to be



**Figure 3.7** Message Pop-up with Entry Fields

displayed, re-display the message until the condition is corrected. This is an exception to the ceaseless operation that is usually recommended.

### 3.8.5 Audible Feedback

On terminals capable of making sounds, a sound should accompany warning and action messages, unless the user has turned off the audible feedback.

### 3.8.6 Guidelines for Creating Messages

The usability of the applications depends to a large degree on how well users understand and respond to its messages. Consider these guidelines for creating usable messages:

Provide help for messages in pop-ups, as it is provided for in other panels.

If the application developers provide a System Manual with additional help for messages, then include an alphanumeric identifier in the online message to help users locate the explanatory information in the manual.

In messages that indicate an error, tell users what is wrong and how to fix the problem.

Write messages and prompts in concise, complete sentences. Use short, simple words and the active voice. Avoid jargon, abbreviations, and acronyms.

Use the same terminology in all messages.

Emphasize any names defined by the system or supplied by users that subsequently appear in message text. For example,

You have selected the "FY 91 Budget Data" file.

Issue an information message to tell users that an action has been completed if there is no other visible indication. For example,

Search complete. String "Accounts \$" not found.

Issue an information message to tell users that processing is proceeding normally so they will not be concerned about a longer than normal wait. For example,

Processing - Please wait.

When an action will take a long time, issue an information message to tell users that the action is partially complete. If possible, tell them how much is still left to do. For example,

Processing 20% complete.

If necessary, tell users what action to take to complete the request. For example,

Type in your user ID and password and press Enter to continue.

Avoid using Yes and No choices when asking users to confirm an action. Users might misinterpret the message. Use short phrases that describe the actions available. For example,

Choose one.

1. Save and exit
2. Exit without saving

is explicit. While

Do you want to exit without saving?\_\_ (Y/N)

1. Yes
2. No

requires users to read the message carefully to interpret its meaning.

---

## 3.9 Command Line

If users want to process system commands without leaving an application, the **U.S.A.I.D. CUI Guidelines** recommend that the application provide a command line to allow users to request those actions directly. Application actions also may

be supported through the command line, giving users an alternative to using the menu bar and pull-downs.

### 3.9.1 Command Line Layout

A command line may be located in the primary window below the message line and above the function key line, or it may be in a pop-up.

All command lines should contain a field prompt and an entry field. For example,

Command => (Line one of entry field is here)

(Optional second line is here)

### 3.9.2 Command Line Interaction

Users can interact with the command line using four common actions: Enter, Command, Prompt, and Retrieve. Help should also be available for the command line.

#### 3.9.2.1 The Enter Action

The Enter action processes a command that has been typed into the command line of a primary window or a pop-up.

After successfully processing a command, the application should clear the Command window. If the Command window is located in a pop-up, the pop-up remains displayed until users request Cancel.

#### 3.9.2.2 The Command Action

The Command action allows users to request a pop-up that contains a Command window. The pop-up appears after users request the Command action or a Command pull-down choice.

#### 3.9.2.3 The Prompt Action

As an application option, users may request the Prompt action for commands when the cursor is in the command line.

The following rules apply for the Prompt action:

When users request Prompt for the command line and no command has been entered, a prompt list pop-up for the command line is displayed, containing a list of the available commands. Users can search the list of commands using global search characters.

When users select a command from the prompt list pop-up and press the Enter key, a pop-up is displayed to help users complete that command.

The application should display a pop-up telling users when the command is complete and that they can press the Enter key to process the command. The Prompt should never process a command immediately.

### 3.9.2.4 The Retrieve Action

Users request the Retrieve action to re-display the last command that was issued. The previous command appears in the command line entry field. Users should be able to change the command, add parameters to it, or press the Enter key to reissue it.

### 3.9.3 Applications having a Command Line and a Menu bar

If the application supports both an menu bar and a command line, make sure that the two interface elements are not at odds with each another.

Make sure that functions available from both the menu bar and the command line are consistent. If the action in the pull-down is Exit, the command name for that action also should be Exit. It would be confusing to call the same action Exit in one place and Quit in the other. Do not use the same name for different functions.

---

## 3.10 Function Key Line

The function key line is the bottom line of a window where available actions and their key assignments are listed. Some of the actions that appear in the function key line are common actions and have common meanings in all applications. Other actions that appear in the function key line are unique to the application. Individual applications determine the meanings of the application-specific actions.

Always define the contents of the function key line for each panel.

Function keys should only appear in the function key line when they are appropriate for the current context (e.g., PgUp should not appear when the cursor is at the beginning of the data).

The **U.S.A.I.D. CUI standard** assumes a keyboard with 12 function keys. Even though U.S.A.I.D. has many proprietary Wang keyboards with 16 function keys, application programmers should not use more than the first 12.

If the application requires more than 12 function keys, the first twelve may be reused in combination with the shift, Ctrl, or Alt keys.

Excessive use of function keys should be avoided. A long list of active keys suggests that it may be appropriate to break the screen into simpler parts with fewer choices.

### 3.10.1 Function Key Line Content

The following function key line common actions are required for all application panels in the primary line if the function is supported on the panel:

ESC = Cancel

F1 = Help

F2 =

F3 =

Alt + F4 = Exit

F5 =

F6 = Next Window Pane

F7 =

F8 =

F9 =

F10 = Menu

F11 =

F12 =

### 3.10.2 Function Key Line Action Definitions

Function key line common actions are actions that have common meaning in all applications. Assign those actions to keys so users can request the actions by pressing the appropriate keys. Some of the actions also can be requested from an menu bar pull-down or by a command that users type into the command line.

Following is a description of each of the common actions.

#### PgUp and PgDn

The Page Up and Page Down actions must be provided for all panels that contain any window that can be scrolled vertically (up and down).

#### Cancel

Cancel allows users to back up in the dialogue one panel at a time or to back up from a pull-down to the menu bar.

Repeated Cancel requests let users back out of an individual function or an entire application panel-by-panel, until users reach the highest-level panel. At that point, another Cancel request has the same effect as the Exit action.

*If the application determines that significant information could be lost because of the Cancel action, a confirmation message appears, prompting users to save or discard information.*

When users Cancel a panel, the information in the panel is either discarded or retained, depending on how the application designers want to establish the default panel values for the next display of the panel.

#### Exit

Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. Repeated Exit requests return the dialogue to the highest level in the panel hierarchy.

Exit is not used in pop-ups, except for help pop-ups, because Exit applies to the function or application as a whole, not just to the sequence of pop-ups. Users may Cancel a pop-up.

If operator-entered data could be lost, the application must provide a warning and an opportunity to return to the screen and finish the process.

### Help

There are several help actions to assist users while they are using an application. The help actions are the following:

Help (F1): Should be context sensitive and provide information about a specific item or field, an application panel, or the help facility, as follows:

F1 provides contextual help when the cursor is on a choice or in an entry field in an application panel.

F1 provides information about the application panel, known as Extended help, when the cursor is not on a choice or in an entry field.

F1 provides information about the help facility when F1 is requested from a help panel.

Help is required in all application panels.

Extended help (Shift-F2): Provides information about the entire application panel from which users requested help.

An Extended help panel is required for every application panel and all help panels, except Extended help panels.

### Move-to-Menu-Bar

The Move-to-menu-bar action allows users to switch the cursor back and forth between the menu bar and other panel windows.

### Undo

The Undo action reverses the most recently executed user action. Because the Undo action deals with hidden objects, the text to be undone should be displayed to reflect exactly what is being undone.

### Enter

When users are finished interacting with a panel that contains entry fields, selection fields, a selection list, or an action list, the panel must be submitted to the application with a specific action request, such as an action selected from a pull-down or the Enter action. The Enter key is used by the operator to process the entire panel. Users request Enter by pressing the Enter key. Only one key functions as Enter, and that particular key must be identified for each keyboard. The Tab, back-Tab and Arrow keys are used to navigate among entry fields in a window.

If a process has multiple screens, the effect of Enter is to get to the next logical screen. For example, a browse may have multiple screens: the user may press Enter to proceed to the next screen or press F8=Forward.

Enter should always result in a visible response to the user.

---

## 3.11 U.S.A.I.D. and System Identifiers

U.S.A.I.D. system identifiers should appear as the first screen when an application is invoked. This is the equivalent of the logo screen in the graphic interface. The application developer may choose to implement a character based logo, or just include the following information:

- U.S.A.I.D. Organization or system name
- Purpose of this application
- Application version number
- Application logo, if appropriate
- Copyright statement, if appropriate

Optionally, the application may display information about the system, such as minimum requirements or required memory and hardware configuration.

# Microsoft Windows

---

## 1. Topic

In this appendix some of the features of Microsoft's Windows are discussed. Two perspectives are presented: that of the user and that of the product as an application development package. Windows provides function libraries that provide developers access to most of the features defined in **U.S.A.I.D. Common User Interface Guidelines**, as well as other capabilities that are accessible by designing a Windows-compatible application.

Microsoft Windows has proven to be an effective tool at a reasonable cost to implement the U.S.A.I.D. Common User Interface Guidelines. The Guidelines do not, however, require use of any specific tool, but only require that applications be developed with the functionality specified in the body of this document.

---

## 2. Summary

Microsoft Windows is an operating environment that runs on the PC-DOS and MS-DOS operating system. It is a graphic, multitasking, windowing interface that replaces the traditional DOS interface.

All applications written for the Windows interface will have a common appearance and common command structure that provides some of the benefits of a common user interface (CUI). The user sees a an interface that is consistent across applications and that uses common elements and controls, such as sizable windows, menus, pop-ups, dialogue boxes, and selection lists that adhere to the basic principles of a CUI.

Application developers can use the Windows interface for the built-in subroutines that allow easy implementation of sizable windows, pop-up menus, scroll bars, icons, dialogue boxes and other features of a graphical interface. Windows provides control over scarce resources, such as memory, cpu time, I/O devices and messaging. This also allows the developer to treat the keyboard, mouse, video display, serial port, and printer in a device-independent context and means that the same application can run on a variety of hardware configurations without any change in application code.

---

## 3. User Considerations

The Windows operating environment provides the user with a friendly, forgiving, consistent, graphical means of communicating with the hardware, in place of the rigid, command line-oriented PC-DOS environment. Windows provides the user with a standardized user interface, context switching, data sharing, multitasking and memory management.

---

### 3.1 Standardized User Interface

The standardized user interface is the most noticeable and important user feature of Windows and Windows-compatible applications. The interface uses pictures or **icons** to simulate an office-like environment. Icons are used to represent files, disk drives, applications and system commands. Programs are identified by title bars, and many of the basic file manipulation functions are accessed through menus, using a mouse. Once familiar with Windows the user can quickly learn new Windows applications because of the similarity and common forgiving approach.

---

### 3.2 Context Switching

Context switching is the Windows-provided ability to switch between multiple Windows or Windows-compatible applications without having to stop one and start the other each time. Context switching allows a user to have multiple applications running at the same time. For example, a user in WordPerfect can context switch to Lotus by pressing a "hot key" (ALT+TAB) or using the mouse, and invoking Lotus, obtaining the information they need and "hot keying" back to WordPerfect. They do not have to save and close the WordPerfect file, invoke Lotus and retrieve a file, close Lotus, and return to WordPerfect and reopen the original file. Context switching works with native Windows applications and with Windows-compatible DOS applications. The utility of context switching is demonstrated by the popularity of such TSR (Terminate and Stay Resident) programs as Sidekick and PCTools.

---

---

### 3.3 Clipboard (Data Sharing)

Windows provides a clipboard function that allows users to cut material from one application and paste it into another. This may become one of Windows most used functions. Clipboard is the main data exchange feature of Microsoft Windows. It is a common area to store data handles through which applications can exchange formatted data. For example it allows a user to cut pages from a report and paste them into a letter or spreadsheet. The clipboard can also be used to exchange bitmapped data, so a user may cut graphs from one application and paste it into another.

---

### 3.4 Multitasking

A multitasking operating system allows users to have several applications (or several copies of the same application) running concurrently. This can be important when applications consume a great deal of computer time but require little user interaction, such as spreadsheet recalculations, large print jobs or database queries or reports. While the computer is printing, the user can initiate another application. Although more than one application can be **running** simultaneously, only one is actually **processing**, (i.e. has control of the processor) at a given time. A task that the user is currently working on is referred to as the **active** task. Just as there can be only one application processing at a time, so there is only one active application, but there can be several tasks concurrently running. Partitioning the microprocessor's time is the responsibility of Windows. It controls microprocessor sharing by using queued input and messaging

---

### 3.5 Memory Management

Memory is one of the most important and scarce resources in any microprocessor. Under DOS a maximum of 640 Kilobytes (K) is addressable by an application program. After network and application TSR's are loaded an amount significantly less than 640K might remain available. This means that spreadsheets, documents and project schedules are limited to the size of available memory. When multiple applications are running, the applications must cooperate in order to share available memory. Windows provides for this memory management. As new programs start up and old ones terminate, Windows consolidates free memory and even permits applications to "over commit" memory. When an application "over commits"

---

memory, it contains more code than will fit into available memory. Windows will swap code out to disk, or discard code and reload it later from the application's EXE file. In this manner a processor with only four megabytes of actual memory can accept applications that may require up to 16 megabytes of virtual memory. By providing these services, Windows allows both the developer and the user to focus on the application and not worry about scarce resources.

---

## 4. Developer's Considerations

Microsoft Windows is an operating environment that occupies a position between the user and PC-DOS. Windows provides built-in subroutines for developers to use in building Windows applications. These subroutines provide for common user interface, device independence, memory management and message handling. This frees the application developer to devote time to addressing the application logic, and requires less attention to the hardware details. This also constrains the developer from writing code that directly addresses these items. For example, Windows passes messages to the application that are addressed to the application, so the developer should not try to read directly from the keyboard buffer.

---

### 4.1 Standard User Interface

Windows communicates to the user through the basic elements of the CUI environment; windows, menus and dialogue boxes. Windows also provides access through subroutines to other elements of CUI. Most of these subroutines accept parameters that define the location, color and content of the element on the screen. The appearance of the element is defined by Windows. This relieves the developer from coding the details of a graphic user interface. The developer can focus on using the proper tools for the proper job. For example, using Microsoft's Software Developer Kit (SDK) to define a primary window, the application programmer issues a call to the **CreateWindow** function and specifies such items as the text for the title bar, style, position and size on the screen, and owner of the window (if it is not primary). In order to display the window, a call is made to a function called **ShowWindow**. Windows supplies all of the code necessary to describe the window, put it at the proper position on the screen and see that it meets the standards. Windows also handles manipulation of presentation elements and controls, for instance if a control (e.g. list box) is moved outside the parent work area, Windows will clip, or size, the control. If Windows paints, moves, or destroys the parent window, it also paints, moves or destroys the controls within the window.

---

Most programming languages, such as C or COBOL, has access to a single screen surface. An advantage to developing a Windows application is that Windows can create a number of overlapping windows on which to display information and directions. Windows provides screen management for the developer, and ensures that no two applications attempt to access the same part of the screen display at the same time.

---

### 4.2 Device Independence

Another major feature provided by Windows is device independence. Currently an application package must define the hardware devices that it will support, (monitors, input devices and printers) and supply drivers to handle each device available to the prospective user. As new devices are introduced, or existing devices are fitted with additional features, the application developer must update and test new driver code. This is nonproductive since it usually does not affect or enhance the application logic, and each developer must produce similar code changes. Windows frees the developer from having to deal with every potential variety of monitor, printer and input device available.

In a Windows environment each device driver is written only once. Typically the hardware company supplies the driver for the system. Microsoft includes many drivers with Windows, and many others are available from hardware manufacturers. When Windows is installed, the user can install as many device drivers as necessary to describe their set-up, even if the device isn't currently connected to the computer. Whenever a print or draw command is issued by the application, Windows sends it to the currently selected driver.

This means that the application developer can focus on the logic of the problem and not worry about writing literally dozens of device drivers. The application interacts with Windows, rather than any specific device. The application doesn't need to know what printer is hooked up. Conversely, each device driver works with every application. Developers save time and money, and users can be assured that their current devices are supported by any new software acquired.

Windows accomplishes this task by specifying the minimum capabilities required by the hardware. The Software Developers Kit (SDK) from Microsoft supplies routines that can be broken into the minimal set of operations required by a device. For example, a plotter may not be able to draw a specific geometric figure, such as an oval. The application developer can still use the SDK to draw an oval, since each plotter connected to Windows must be capable of drawing a line. Windows can break the oval, drawing it into a series of small lines.

Another example of how Windows handles different interfaces is the keyboard. Windows predefines a set of legal keystrokes to ensure that your application only receives valid input (the set is similar to all the keystrokes produced by the IBM PC keyboard). If a manufacturer designs a new keyboard containing new codes to be interpreted, the manufacturer should also supply the software necessary to convert the additional keystrokes into Windows-recognized keystrokes. These input definitions cover any legal input device, such as a mouse, track ball, light pen or pointer. If a manufacturer develops a new device (a four button mouse), they must supply the code to convert its signals into Windows predefined set of legal mouse-button clicks.

---

### 4.3 Dynamic Link Libraries

Microsoft Windows provides special libraries, called "Dynamic Link Libraries" (DLLs) that allow applications to share code and resources. A DLL is an executable module containing Windows functions. DLLs are linked to an application when it is executed (dynamic linking), as opposed to static linking, which is linking application files using the linker (**LINK**).

The purpose of dynamic linking is to provide an efficient multitasking environment. A statically linked executable has all of its library function routines included (linked) in the executable. If two or more copies of an application are running at the same time there will be a copy of all statically linked functions for each application.

DLLs allow several applications to share a single copy of a routine. Every standard Windows function resides in one of three DLLs. If two or more Windows applications are running at the same time, all share a single copy of the executable code for a particular function.

In addition to letting applications share code, DLLs are used to share other resources, such as data and hardware. All Windows libraries are DLLs. Application developers can write custom DLLs to share code, data or hardware among their applications.

Along with static link libraries and dynamic link libraries, there is a third type, called an import library, and it is important if your application uses DLLs. An import library contains information that allows Windows to locate code in a DLL. During linking, the linker uses static link libraries and import libraries to resolve references to external routines. When an application uses a routine from a DLL, the linker does not copy code into the executable. Instead it copies information

from the import library which indicates where the DLL routine can be found at run time.

---

## 4.4 Queued Input

In a DOS environment, a program reads from the keyboard by making an explicit call to a function, such as *getchar* or *fscanf*. The function waits until the user presses a key before returning a character code to the application. In Windows, keyboard, mouse and other device input are shared resources. Windows receives all input from the keyboard, mouse and timer and places it in the appropriate program's "message queue". When an application is ready to process input, it reads from its queue and dispatches a message to the appropriate window function.

In Windows, an application receives input in the form of "input messages" that windows sends it. A Windows input message contains information in a uniform format, and contain significantly more information than is available in a standard DOS environment. Windows messages contain the time, position of the mouse, state of the keyboard, scan code of any pressed key, mouse button state, and device generating messages.

All keyboard, mouse and timer messages have identical formats and all are processed in the same way. Additionally, with each message, Windows provides a device independent virtual key code that identifies the key, the device independent scan code generated by the keyboard, and the status of other keys on the keyboard, such as *Num Lock*, *Alt*, *Shift* and *Ctrl*.

---

## 4.5 Dynamic Data Exchange

Microsoft Windows provides several methods for transferring data between applications. In general, the Windows environment supports three mechanisms that applications can use to exchange data with one another:

- \* Clipboard transfers
- \* Dynamic Link Libraries
- \* Dynamic Data Exchange

The clipboard is the primary data exchange feature of Microsoft Windows. The clipboard lets the user transfer data between applications in the system. One

---

application is used to copy formatted text into a common area and stores handles through which applications can access the data. Another application may then ask to paste the data from the clipboard into its workspace. The clipboard requires direct involvement by the user to transfer the data.

A Dynamic Link Library (DLL) can be used to store data between applications. The DLL provides an application interface for storing and retrieving data. The data is stored in the DLL's local heap or in the static area of its memory segment. Handles or addresses to this data can be passed to the appropriate application.

Windows Dynamic Data Exchange (DDE) is a standard for cooperating applications that allows them to exchange data and invoke remote commands by means of Windows messages. Windows is a message-based architecture, and the message is the primary means of passing information to an application. However, Windows messages only contain two parameters for passing data. These messages can only pass a few words at a time. If more information is to be passed, then the parameters must refer indirectly to the other pieces of data. The DDE protocol defines exactly how the *wParam* and *lParam* are used to pass larger pieces of data.

DDE is most appropriate for data exchange that do not require ongoing user interaction. Normally an application provides a method for the user to establish the link between the applications exchanging data, but once the link is in place the applications can exchange data without further user involvement. DDE can be used for a wide range of application functions, including:

- \* Linking to real-time data, such as process control or instrumentation.
- \* Creating compound documents, where graphs and charts dynamically change based on changes in the underlying data.
- \* Performing data queries between applications, such as linking spreadsheets and databases.

---

## 4.6 DOS Applications Compatibility

Many existing DOS applications can be easily run under Windows, although it is designed to easily share resources and features only with applications that are written specifically for it. DOS applications can be classified as **compatible**, or **incompatible** with Windows. Generally if an application uses DOS or BIOS interrupts to read the keyboard or generate display, it will operate within the Windows environment. If an application writes directly to a video display, uses

graphics or takes control of the hardware keyboard interrupts, it will not be compatible with Windows and may cause conflicts (e.g. system may lock up) to occur.

---

## 5.0 Summary

As more PC users become aware of the benefits of graphical user interfaces, the task of developing reliable custom software becomes more complex and difficult. While Windows provides benefits to the user, it complicates the programmer's job.

An application developer can use any of several software development packages to build programs that use the Windows environment. Attached is a list of some of the development tools currently available for use with Windows:

Actor  
The Whitewater Resource Toolkit  
The Whitewater Group  
1800 Ridge Avenue  
Evanston, IL 60201  
(800) 869-1144

Knowledge Pro  
Knowledge Garden  
473A Malden Bridge Rd  
Nassau, NY 12123  
(518) 766-3000

PLUS  
Spinnaker Software  
201 Broadway  
Cambridge, MA 02139  
(617) 494-1200

Tool Book  
Asymetrix Corp.  
P.O. Box 40419  
Bellevue, WA 98004-0419  
(206) 455-3071

Windows Software Development Kit  
Microsoft Corp  
1 Microsoft Way  
Redmond, WA 98052-6399  
(206) 882-8080  
(800) 323-3577

Visual Basic  
Microsoft Corp  
1 Microsoft Way  
Redmond, WA 98052-6399  
(206) 882-8080  
(800) 323-3577

## U.S.A.I.D. Function Key Assignments

Function Key	Standard Usage	Shifted	Ctrl+Key	Alt+key
F1	Help	Extended Help		
F2				
F3				
F4			Close Doc Window	Close Application
F5				
F6	Move Clockwise to next Window Pane	Move Counter-clockwise to next window pane	Next Document Window	Next non-doc window
F7				
F8				
F9				
F10	Menu Bar			
F11				
F12				

- Notes: 1. Empty cells indicate this key option is available for application use.  
 2. If keyboard has additional function keys, they should not be used.

U.S.A.I.D. Common User Interface Guidelines

---

Function Required	CUI Standard Function Key	CUI Mouse Interaction	Recommended Menu Pick
Help	F1	Menu Selection	<b>H</b> option on main menu
Exit or Quit	Alt+F4	Double click on upper left control box	<u>F</u> ile, <u>E</u> xit
Page Up (back)	Page Up	Move Scroll Bar Up	
Page Down (forward)	Page Down	Move Scroll Bar Down	
Go to Beginning of Line	Home	Point and click	
Go to Beginning of Data	Ctrl+Home	Scroll Bar	<u>E</u> dit, <u>G</u> o To
Go to End of Line	End	Point and Click	
Go to End of Data	Ctrl+End	Scroll Bar	<u>E</u> dit, <u>G</u> o To
Toggle to another Appl. in Progress	Ctrl+ESC		
Menu Bar	F10	Move Mouse	
Menu Pick, when Bar Active	Press Underlined Letter	Click Left Button	

U.S.A.I.D. Common User Interface Guidelines

---

<b>Function Required</b>	<b>CUI Standard Function Key</b>	<b>CUI Mouse Interaction</b>	<b>Recommended Menu Pick</b>
Menu Pick - Bar not Active	Press Alt + Underlined Letter	Click Left Button	
Accept Current Option	Enter	Click Left Button	
Accept Multiple Options from List		Ctrl+left click	
Move cursor to next option or field	Tab	Move mouse	
Accept Default Option - usually highlighted	Enter		
Cancel Dialog	ESC or cancel button		



## Word Processing Functions

This Appendix contains a table of the major word processing functions available through WordPerfect. It is recommended that the Windows version of WordPerfect be used as a standard. Both the DOS version and the Windows version of WordPerfect allow the user to redefine the meaning of all keys on the keyboard. WordPerfect supplies a CUA compatible keyboard option and template which is recommended for use at U.S.A.I.D. These codes are provided for clarification and can be used to validate the functionality of a wordprocessing function, as well as provide guidance that may be used in select cases.

Key Strokes	Word Perfect Default	Word Perfect CUA Option
<b>Word Processing Functions</b>		
F1	Cancel	Help
F2	Search->	Search
F3	Help	Save as
F4	->Indent	Open
F5	List Files	Print
F6	Bold	Next Window Pane
F7	Exit	Indent
F8	Underline	Select
F9	Merge R	Font
F10	Save	Menu Bar
F11	Reveal Codes	Figure Retrieve
F12	Block	Mark Text
<i>Ctrl</i> F1	Shell	Speller
<i>Ctrl</i> F2	Spell	Replace

U.S.A.I.D. Common User Interface Guidelines

<b>Key Strokes</b>	<b>Word Perfect Default</b>	<b>Word Perfect CUA Option</b>
<i>Ctrl F3</i>	Screen	Redisplay
<i>Ctrl F4</i>	Move	Close
<i>Ctrl F5</i>	Text in/Out	Date Text
<i>Ctrl F6</i>	Tab Align	Next Document
<i>Ctrl F7</i>	Footnote	Hanging Indent
<i>Ctrl F8</i>	Font	Margins
<i>Ctrl F9</i>	Merge/Sort	Tables
<i>Ctrl F10</i>	Macro define	Macro Record
<i>Ctrl F11</i>	Large	Horizontal Line
<i>Ctrl F12</i>	Block	Merge
<i>Alt F1</i>	Thesaurus	Thesaurus
<i>Alt F2</i>	Replace	Search Prev
<i>Alt F3</i>	Reveal Codes	Reveal Codes
<i>Alt F4</i>	Block	Exit
<i>Alt F5</i>	Mark text	Paragraph Num
<i>Alt F6</i>	Flush right	Next Window
<i>Alt F7</i>	Math/Columns	Flush Right
<i>Alt F8</i>	Style	Styles
<i>Alt F9</i>	Graphics	Page
<i>Alt F10</i>	Macro	Macro Play
<i>Alt F11</i>		Text Box Create
<i>Alt F12</i>		Generate
<i>Shift F1</i>	Setup	Help: What Is?
<i>Shift F2</i>	<-Search	Search Next
<i>Shift F3</i>	Switch	Save

U.S.A.I.D. Common User Interface Guidelines

Key Strokes	Word Perfect Default	Word Perfect CUA Option
<i>Shift F4</i>	->Indent<-	New
<i>Shift F5</i>	Date/Outline	Print Preview
<i>Shift F6</i>	center	Prev Window Pane
<i>Shift F7</i>	Print	Center
<i>Shift F8</i>	Format	Select Cell
<i>Shift F9</i>	Merge Codes	Line
<i>Shift F10</i>	Retrieve	
<i>Shift F11</i>	Italics	Figure Edit
<i>Shift F12</i>		Define
<b>Cursor Movement Commands</b>		
Home, Home, Home (↑)	Top of Document	
Home, Home, Home (↓)	End of Document	
Home, Home (→)	End of Line	
Home, Home (←)	Beginning of Line	
Home (→)	Right side of screen	
Home (←)	Left side of screen	
Home (↑)	Top of screen	
Home (↓)	Bottom of screen	
↓	Down one sentence	
<i>Alt ←</i>	Left one column	
<i>Alt →</i>	Right one column	
↑	Up one sentence	
<i>Ctrl(home)</i>	Go to (page)	Go to Beginning of Doc

Key Strokes	Word Perfect Default	Word Perfect CUA Option
←	Left one character	
→	Right one character	
<i>Ctrl</i> ←	Move word right	
<i>Ctrl</i> →	Move word left	
<i>Ctrl</i> ↑	Up one paragraph	
<i>Ctrl</i> ↓	Down one paragraph	
Page Down	First line of next page	
Page Up	First line of previous page	
<b>Delete Commands</b>		
Delete	Delete one character	
Delete+End	Delete to end of line	
<i>Ctrl</i> +bkspc	Delete word	
<i>Ctrl</i> +PgeDown	Delete to end of page	
Home+bkspc	Delete from cursor to beginning of word	
Home+delete	Delete from cursor to end of word	

# Glossary

**active** - describes the window or icon to which the next keystroke or command will apply. If a window is active, its title bar changes color to differentiate visually from other open windows. If an icon is active, the Control menu appears.

**application icon** - a graphic that appears only after you start an application and then minimize it. Application icons are the only icons that appear on the desktop, outside window borders.

**application shortcut key** - a key combination that brings an application to the foreground when running windows in 386 enhanced mode.

**application window** - a window that contains a running application. The name of the application appears at the top of this window. An application window may contain multiple document windows.

**associate** - to identify a filename extension as "belonging" to a certain application. When you select a file with an extension that has been associated with an application. That application is opened automatically.

**background** - the area behind the active window.

**bitmap** - an image stored as an array of bits.

**boot** - to start your computer, or to restart it, loading the disk operating system (DOS).

**branch** - a segment of the File Manager Directory Tree representing a directory and any subdirectories within it.

**built-in font** - (also known as resident or hardware font) - font that is built into the read-only memory (ROM) of a printer.

**cascade** - a way of arranging open windows on the desktop so that they overlap each other with the title bar of each window remaining visible.

**cascading menu** - a menu that opens from a command on another menu.

**check box** - a small square box that appears in a dialogue box and that can be selected or cleared. When the check box is selected, an X appears in the box. A check box represents an option that you can set.

**choose** - to use a mouse or key combinations to pick an item that begins an action in windows.

**click** - to press and release a mouse button quickly.

**Clipboard** - A temporary storage location used to transfer data between documents and between applications.

**close** - to remove a document window or application window from the desktop. You can choose to save or abandon the document in a document window before you close the application.

**collapse** - to "hide" additional directory levels below a selected directory in File Manager.

**command** - a word or phrase usually found in a menu that you choose in order to carry out an action.

**command button** - a button in a dialogue box. It carries out or cancels the selected action. Two common command buttons are labeled Cancel and OK.

**confirmation message** - a message displayed by window when you have specified a destructive action, asking if you are sure you want to proceed. For example, windows displays a confirmation message when you tell it to delete a file.

**contextual help** - A help screen that provides information relevant to the specific area in which the cursor is located, in the context of the current application. Contrast to *extended help* and *tutorial*.

**Control menu** - the menu appearing on every application that runs in a window and on some non-window applications. Icons, some dialogue boxes and windows within an application workspace also have Control menus. For applications running in a window and for icons and dialogue boxes, Control menu commands move, change the size of, and close windows. For non-windows applications, the Control-menu commands transfer information and perform other miscellaneous functions. Also known as System menu.

**Control-menu box** - the icon that opens the Control menu for the window. It is always at the left of the title bar.

**copy** - to put a copy of the selected text or item on the Clipboard so you can transfer it to another location. Most windows applications have a Copy command that performs this task.

**current directory** - the directory that is currently highlighted in the Directory Tree or whose directory window is the active window.

**cursor** - a visual cue that shows the location of the current focus of activity. The cursors are the *selection cursor* and the *text entry cursor*.

**cut** - to move text from a document into a temporary storage area called the Clipboard.

**data file** - any file created within an application: a word processing document, a spreadsheet, a database file, a chart, and so forth.

**default button** - the command button in some dialogue boxes selected as the most logical or safest choice. This button has a bold border when the dialogue box appears. and pressing ENTER chooses the button.

**default printer** - the printer that windows applications automatically use when you choose the Print command. You can have only one default printer and the default printer must also be an active printer.

**desktop** - the screen background for windows on which windows, icons, and dialogue boxes appear.

**desktop pattern** - a geometric pattern that appears across the desktop. You can use Control Panel to design your own pattern or choose one from patterns provided by windows.

**destination directory** - the directory to which you intend to copy or move one or more files.

**dialogue box** - a rectangular box that either requests or provides information. Many dialogue boxes present options to choose among before windows can carry out a command. Some dialogue boxes present warnings or explain why a command cant be completed.

**DIRECTION keys** - the four arrow keys on your computer keyboard. Each arrow key is named for the direction the key points: UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW.

**directory** - a collection of files and subdirectories that are stored at the same location on a disk. The name of the directory identifies its location. Part of the structure for organizing your files on a disk. See also subdirectory.

**Directory Tree** - a graphic display in File Manager of the directory structure of a disk. The directories on the disk are shown as a branching structure that resembles a tree. Directories are shown as branches off the top-level directory, known as the root directory.

**directory window A** - File Manager window that lists the contents of a directory. Each directory window shows all the files and other directories contained in the directory.

**disk drive** - a device used for storing and retrieving data on disks.

**disk-drive icon** - an icon in File Manager that represents a disk drive. Different icons depict floppy disk drives, hard disk drives, network disk drives, RAM drives, and CD-ROM drives.

**document window** - a window within an application window. A document window contains a document you create or modify by using an application. There can be more than one document window in an application window.

**double-click** - to rapidly press and release a mouse button twice without moving the mouse. Double clicking carries out an action, such as opening an icon.

**drag** - to move an item on the screen by holding down the mouse button while moving the mouse. For example, you can move a window to another location on the screen by dragging its title bar.

**drop-down list box** - a single-line dialogue box that opens to display a list of choices.

**exclusive application** - an application that has sole use of the computer's resources while it is running in the foreground with windows in 386 enhanced mode. When an exclusive application is running in a window, it gets most, but not all, of the resources.

**expand** - to show currently hidden levels in the Directory Tree. With File Manager. You can expand a single directory level, one branch of the tree, or all branches at once.

**expanded memory** - memory in addition to conventional memory that is available for applications. It is allocated in 16K blocks. windows will make use of expanded memory only if the /R switch is used at start up. windows running in 386 enhanced mode simulates expanded memory for the applications that need it. windows running in standard mode or 386 enhanced mode allows applications to use expanded memory, but it does not use expanded memory itself in the management of applications.

**extend selection** - to select more than one object. For example. you can select a group of files to be moved or copied with File Manager.

**extended help** - A choice in the help menu where the user is offered general information about the application, as opposed to help with the specific task at hand found in *contextual help*.

**extended memory** - memory in addition to conventional memory that is not readily accessible to MS-DOS or MS-DOS applications. Extended memory can not be used on 8086/88 computers. windows running in standard mode or 386 enhanced mode uses extended memory to manage and run applications.

**extension** - the period and three letters at the end of a filename. An extension identifies the kind of information a file contains. For example, files created with Calendar have the extension .CAL.

**file** - a document or application that has been given a name. All documents are stored as files in windows.

**file attribute** - a characteristic of a file--for example. the read-only attribute--that can be changed using File Maintenance

**file format** - the structure or arrangement of data stored in a file.

**filename** - the name of a file windows uses DOS filenaming conventions.

**fixed-width font A font** - in which all characters have uniform widths.

**flow control** - the processes and procedures used to regulate the rate at which data is transferred from one device to another.

**font** - a graphic design applied to all numerals, symbols, and characters in the alphabet. A font usually comes in different sizes and provides different styles. such as hold, italic, and underlining for emphasizing text.

**font cartridge** - a piece of hardware that is plugged into a printer to supply one or more fonts.

**font family** - a group designation that describes the general look of a font. For example, the Roman font family contains fonts with serifs and variable character widths, such as Tms Rmn.

**font set** - a group of fonts designed for use with a specific device resolution. windows includes seven sets of fonts.

**font size** - See point size.

**footer** - text that appears at the bottom of every page of a document when it is printed.

**foreground** - the area of the screen occupied by the active window.

**format (1)** - the appearance of text on the pages of a document. **(2)** To prepare a disk so that it can hold information. Formatting a disk erases all information that was previously on it.

**full-screen application** - any non-windows application that occupies the whole screen rather than running in a window.

**graphics resolution** - the level of quality at which windows prints graphic. The higher the resolution, the better the quality of the printed graphics (and the slower the printing).

**group** - a collection of programs in Program Manager. Grouping your programs makes them easier to find when you want to start them.

**group icon** - the graphic that represents a Program Manager group that is minimized. Double-clicking the group icon opens the group window.

**group window** - a window that displays the items in a group within Program Manager. These items can be applications or data files associated with applications.

**handshake** - a flow-control or "go ahead." signal sent by a local computer to a remote computer when working with a communications program such as Terminal. XON/OFF is the standard software handshaking method, although it can't be used with remote systems that use a hardware handshaking method.

**header** - text that appears at the top of every page of a document when it is printed.

**hidden file** - a system file that cannot be viewed, such as your MS-DOS BIOS file.

**high memory area** - The first 64K of extended memory. This area used by some applications.

**highlighted** - indicates that an object or text is selected and will be affected by your next action. Highlighted text appears in reverse video on monochrome displays or in color on some color displays. Highlighted objects might change color or be surrounded by a selection cursor.

**icon** - a graphical representation of various elements in windows, such as disk drives, applications, and documents.

**inactive window** - any open window that you are not currently working in.

**insertion point** - the place where text will be inserted when you type. The insertion point usually appears as a flashing vertical bar in an applications window or in a dialogue box. The text you type appears to the left of the insertion point, which is pushed to the right as you type.

**list box** - within a dialogue box, a box listing available choices--for example, the list of all available files in a directory. If all the choices wont fit, the list box has a vertical scroll bar.

**macro** - a series of actions recorded using the Recorder accessory. When you run the macro, Recorder carries out all the recorded actions. Macros can be assigned to special keys, called shortcut keys. or to longer descriptive names.

**mark** - to select text in a non-windows application.

**Maximize button** - the small box containing an Up arrow at the right of the title bar. Mouse users can click the Maximize button to enlarge a window to its maximum size. Other users can use the Maximize command on the Control menu.

**memory-resident software** - software that is loaded into memory and is available for use even when another application is active. Also known as TSR software.

**menu** - a list of items, most of which are windows commands. Menu names appear in the menu bar near the top of the window. You use a command on a menu by selecting the menu name, then choosing the command.

**menu bar** - the horizontal bar containing the names of all the application menus. It appears below the title bar.

**Minimize button** - the small box containing a Down arrow at the right of the title bar. Mouse users can click the Minimize button to shrink a window to an icon. Other users can use the Minimize command on the Control menu.

**network** - a group of computers connected by cables and using special software that allows them to share equipment (such as printers) and exchange information.

**network disk drive** - a disk drive that is available for public use on a network. Network disk drives are often used to store data files for many people in a work group.

**non-windows application** - programs that were not designed to run within the windows environment. Non-windows applications do not necessarily follow any of the windows user interface conventions. Although most non-windows applications can run with windows, there may be limitations regarding how many of windows' features they can take advantage of.

**open** - to display the contents of a file in a window or to enlarge an icon to a window.

**option** - a choice in a dialogue box. An option affects the way in which a command is carried out. Dialogue boxes have several kinds of options, including mutually exclusive option buttons and non-exclusive check boxes.

**option button** - A small round button that appears in a dialogue box and is used to select an option. Within a group of related option buttons, you can select only one button.

**parallel interface** - an interface between a computer and a printer in which the computer sends multiple bits of information to the printer simultaneously. Parallel and Centronics interfaces are the same type.

**parallel port** - a connection on a computer, usually LPT1, where you plug in the cable for a parallel printer. windows supports parallel ports LPT1 through LPT3.

**parameter** - information added to the command that starts an application. This information determines how the application will run. For example, to start Microsoft Word in character mode, you would type word /c at the DOS prompt. The /c is the parameter. A parameter can be a filename or any type of information up to 62 characters in length.

**paste** - to transfer the contents of the Clipboard to an application. Many applications have a Paste command that performs this task.

**pathname** - the directions to a directory or file within your system. For example, C:\windows3\APPTS\JUNE.CAL is the pathname of the JUNE.CAL file in the APPTS subdirectory in the windows3 directory on drive C.

**personal windows directory** - the directory usually located on your hard disk, that contains the windows files that were loaded to your system when you ran windows Setup.

**PIF (program information file)** - a file that provides information that windows needs to run a non-windows application. For example, you use a PIF to tell windows to run a non-windows

**pixels** - the smallest graphic units on the screen. Also known as picture elements (pels).

**point** - to move the pointer on the screen until it rests on the item you want to select or choose.

**point size** - the height of a printed character. A point equals 1/2 of an inch.

**pointer** - the arrow-shaped cursor on the screen that indicates the position of the mouse.

**port** - a connection on a computer where you plug in the cable that carries data to another device.

**print queue** - A list of files that have been sent to a particular printer. The list indicates the file currently printing and those waiting to be printed.

**Printer Cartridge Metrics file (PCM)** - a file that contains Printer Fonts Metrics files for each font on a font cartridge.

**printer driver** - software that controls how your computer and printer interact. A printer-driver file supplies windows with information such as the printing interface, descriptions of fonts, special features, and so on.

**Printer Font Metrics file (PFM)** - A file that supplies a windows printer driver with information about a font, such as family, point size, widths of individual characters, and more.

**program item icon** - the graphic that represents an application that you can start from Program Manager. A program item icon is contained in a group window.

**proportional font** - a font in which different characters have varying widths.

**protected mode** - the operating mode of a computer that is capable of addressing extended memory directly.

**protocol** - a set of rules that define how computers communicate with each other.

**pushbutton** - A rounded-corner rectangle with text inside. Pushbuttons are used for actions that occur immediately when the button is selected.

**radio button** - A control that consists of a circle and choice text. Radio buttons are combined to show users a fixed number of mutually exclusive choices.

**RAM (random access memory)** - the memory that is used to run applications and perform other necessary tasks while the computer is on. When you turn the computer off, all information in RAM is lost. See also virtual memory.

**raster font** - a font created as a graphic bitmap image. It is available only in a fixed size--not scalable. A raster font is used mostly on the screen, but is also used by some dot-matrix printers and built into some laser printers.

**read-only file** - a file that you can only open and read. You cannot edit a read-only file and save it again.

**real mode** - an operating mode that windows runs in to provide maximum compatibility with versions of windows applications prior to 3.0. Real mode is the only mode available to computers with less than 1 MB of extended memory.

**Restore button** - the small box containing a Down arrow and an Up arrow at the right of the title bar. The Restore button appears after you have enlarged a window to its full size. Mouse users can click the Restore button to return the window to its previous size. Other users can use the Restore command on the Control menu.

**root directory** - the highest directory of a disk. The root directory is created when you format the disk. From the root directory, you can create other directories.

**scaled point size** - a point size that approximates a specified point size for use on the screen.

**screen font** - a raster font designed to duplicate a printer font on the screen. See also raster font.

**scroll** - to move text or graphics up or down, or left or right, in order to see parts of the file that cannot fit on the screen.

**scroll bars** - the bars at the bottom and right edge of a window whose contents are not entirely visible. Each scroll bar contains a small box, called a scroll box, and two scroll arrows to allow different types of scrolling.

**scroll buffer** - in Terminal, the buffer that holds typed or received information that does not fit on the screen.

**select** - to highlight an item by clicking it with the mouse or using key combinations. Selecting does not initiate an action. After selecting an item, you choose the action you want to affect the item. See also choose and highlighted.

**selection cursor** - the marking device that shows where you are in a window, menu, or dialogue box. The selection cursor can appear as a highlight or as a dotted rectangle around the text in a dialogue box option.

**serial interface** - an interface between a computer and a printer in which the computer sends single bits of information to the printer, one after the other. Serial, asynchronous, and RS232 interfaces are all the same type.

**serial port** - a connection on a computer. usually COM1, where you plug in the cable for a serial printer or another serial communications device, such as a modem.

**share** - a partition of a network disk drive.

**shortcut key** - a key combination that carries out some action in windows. For example, pressing ALT + ESC switches among loaded applications.

**shrink** - to reduce a window to an icon at the bottom of the desktop using the Minimize button. The document or application remains open and can be the active document or application.

**soft font** - a font that is downloaded to your printers memory from a disk provided by the font manufacturer.

**solid color** - the color that appears on a display when all pixels are the same color. On a monochrome display, there are only two solid colors: black and white.

**source directory** - the directory that contains the file or files you intend to copy or move.

**spool** - to print a document or file in the background while working on something else.

**standard mode** - the normal operating mode for running windows. This mode provides access to extended memory and also lets you switch among non-windows applications.

**standard setting** - standard settings are settings shipped with windows. For example, if you print a drawing or document from one of the windows accessories without entering new margin settings, the accessory uses standard margin settings to print.

**subdirectory** - a directory contained within another directory. All directories are subdirectories of the root directory.

**swap file** - an area of your hard disk that is set aside for exclusive use by windows in 386 enhanced mode. This area is used only when your system runs low on memory.

**switch** - See parameter.

**System menu** - See Control menu.

**system time** - the time set by your computers internal clock.

**terminal emulation** - a setting specified with Terminal that causes your computer to emulate a remote computer. Terminal emulation allows your computer to display data it receives and to use features of the remote computer.

**text box** - a box within a dialogue box where you type information needed to carry out the chosen command. The text box may be blank when the dialogue box appears or may contain text.

**text file** - a file containing only letters, digits and symbols. A text file usually consists of characters coded from the ASCII character set.

**tile** - a way of arranging open windows on the desktop so that no windows overlap but all windows are visible. Each window takes up a portion of the screen.

**title bar** - the horizontal bar located at the top of a window and containing the title of the window. On many windows, the title bar also contains the Control-menu box and Maximize and Minimize buttons.

**TSR software** - See memory-resident software.

**vector font** - a series of dots connected by lines that can be scaled to different sizes. Plotters typically use vector fonts. Also known as stroke font.

**virtual machine** - an environment created by windows running in 386 enhanced mode in which an application can run and behave as if it had an entire machine all to itself. windows in 386 enhanced mode can have multiple applications running in their own separate virtual machines at the same time.

**virtual memory** - a memory management system used by windows running in 386 enhanced mode. which allows windows to behave as if there were more memory than is actually present in the system. Virtual memory equals the amount of free RAM plus the amount of disk space allocated to a swap file that windows uses to simulate additional RAM.

**volume label** - a name that identifies a disk. The volume label appears in the title bar of the Directory Tree window.

**wallpaper** - picture or bitmapped pattern that appears as the windows desktop.

**wildcard character** - a character that represents another character. In filenames, you can use the asterisk (\*) as a wildcard character to indicate any character or group of characters that might match that position in other filenames. For example, \*.EXE represents all files that end with the .EXE filename extension.

**window** - a rectangular area on your screen in which you view an application or document. See also application window and document window.

**windows application** - any application that was designed especially for windows and will not run without windows. All windows applications follow the same conventions for arrangement of menus, style of dialogue boxes, use of the keyboard and mouse, and so on.

**word wrap** - a feature that moves text from the end of a line to the beginning of a new line as you type. With word wrap, you do not press ENTER at the end of each line in a paragraph

**workspace** - the area of a window that displays the information contained in the application or document you are working with.

**X-Windows** - a portable software standard developed at Massachusetts Institute of Technology. It controls the displays of workstations, typically in the UNIX environment and provides a standard environment for applications development.

## Index

Accelerator keys	27, 80
Active window	16, 19, 27, 111, 113, 117
Alt	27, 89, 94, 99, 103-105, 108, 109, 124
Apple	iii, 1-3, 17
Application design	5
Application program	20-22, 95
Arrow keys	24, 38, 50, 78, 81, 82, 92, 114
Audible	8, 53-55, 57, 67, 85
Background	1, 16, 43, 111, 114, 125
Beep	28, 57
Cancel	7, 10, 18, 42, 45, 46, 54, 56-58, 77, 80, 82, 84, 87, 89-91, 105, 107, 112
Card	5
Cascade	19, 112
Change	i, iii, 8-12, 18, 23, 30, 40, 44, 45, 48, 51-53, 58, 61, 88, 93, 100, 113, 118
Characters	13, 44, 74, 77, 79, 88, 117, 121, 122, 126, 127
Check Box	33, 36, 37, 112
Choosing	ii, 3, 5, 39, 59, 119
CICS	67
Click	21, 38, 50, 52, 60, 104, 105, 112, 115, 119, 120, 123
Commands	i, 4, 8, 26, 67, 86-88, 94, 100, 109, 110, 113, 119
Common User	1, ii, iii, 2, 4, 6, 11, 14, 15, 63, 93, 96
Confirm	86
Control	9, 18, 19, 21-25, 32, 36, 39, 42, 51, 52, 78, 93, 95, 96, 100, 101, 104, 111, 113, 114, 117-120, 122, 123, 125, 126
Copy	10, 29-31, 98, 100, 113, 114, 125
Ctrl	27, 50, 89, 99, 103-105, 107-110
CUA	iii, 17, 107
Cursor	10, 19, 24, 27, 31, 38, 43-46, 50, 54, 59, 60, 63, 64, 71-73, 77-79, 81, 82, 87, 89, 91, 105, 109, 110, 113, 118, 121, 124
Cut	10, 30, 31, 95, 113
Data	1, 2, 7, 8, 11, 20, 22, 28, 29, 43, 45, 55, 64, 70, 71, 85, 89, 91, 94, 95, 98-100, 104, 112, 114, 115, 116-118, 120, 122, 126
Default	8, 29, 33, 38, 39, 42-45, 90, 105, 107, 114
Delete	8, 9, 29, 31, 44, 45, 110, 113
Desktop	1, 2, 5, 17, 111, 112, 114, 124, 126, 127
Directories	5, 33, 37, 51, 52, 115, 123, 125
Directory	33, 51, 52, 111-116, 119, 121, 123, 125, 127
Disk Drive	115, 120, 124

## U.S.A.I.D. Common User Interface Guidelines

---

Display . . . . .	1, 10, 21, 22, 25, 28, 29, 34, 35, 37, 41, 42, 44-48, 52, 61, 65, 66, 74, 75, 79, 80, 83-85, 88, 90, 92, 93, 96, 97, 100, 115, 120, 125, 126
Document . . . . .	10, 18, 30, 32, 48, 93, 103, 108, 109, 111-118, 124, 125, 127
DOS . . . . .	19, 93-96, 99, 100, 107, 111, 116-118, 121
Dotted . . . . .	38, 124
Double clicking . . . . .	115
Drop-down . . . . .	33, 39, 40, 39, 40, 115
EASEL . . . . .	22
Edit . . . . .	9, 18, 26, 28-30, 79, 104, 109, 123
Error . . . . .	8, 11, 22, 54, 56, 57, 85
Esc . . . . .	80, 89, 104, 105, 124
Exit . . . . .	9, 12, 21, 27, 29, 30, 45, 56, 78, 79, 86, 88-91, 104, 107, 108
Expand . . . . .	116
F1 . . . . .	4, 31, 60, 89, 91, 103, 104, 107, 108
F2 . . . . .	89, 91, 103, 107, 108
F3 . . . . .	89, 103, 107, 108
F4 . . . . .	89, 103, 104, 107-109
F5 . . . . .	27, 89, 103, 107-109
F6 . . . . .	89, 103, 107-109
F7 . . . . .	89, 103, 107-109
F8 . . . . .	89, 92, 103, 107-109
F9 . . . . .	89, 103, 107-109
F10 . . . . .	27, 89, 103, 104, 107-109
F11 . . . . .	90, 103, 107-109
F12 . . . . .	90, 103, 107-109
File . . . . .	5, 6, 9, 10, 25-30, 33, 40, 41, 51-53, 58, 65, 71, 79, 80, 85, 94, 96, 104, 111-118, 120-123, 125, 126
File Manager . . . . .	111, 112, 115, 116
Find . . . . .	9, 33, 42, 63, 118
Fonts . . . . .	22, 117, 122, 126
Footer . . . . .	117
Foreground . . . . .	111, 115, 117
Format . . . . .	32, 74, 99, 109, 116, 117, 123
Function Keys . . . . .	i, 8, 12, 28, 68, 78, 80, 89, 103
Graphics . . . . .	22, 27, 30, 31, 38, 101, 108, 118, 123
Header . . . . .	118
Help . . . . .	i, 4, 5, 7, 18, 22, 26-29, 31, 42, 46, 47, 50-52, 55, 57, 59-61, 60, 61, 63, 69, 71, 76-79, 81, 83-85, 87-89, 91, 103, 104, 107, 108, 113, 116
Hidden . . . . .	28, 30, 34, 39, 91, 116, 118
IBM . . . . .	ii, iii, 2, 3, 17, 98
Icon . . . . .	1, 20, 55-58, 111, 113, 115, 118-120, 122, 124
Image . . . . .	111, 123

---

## U.S.A.I.D. Common User Interface Guidelines

---

Inactive	49, 119
Include	ii, 1, 2, 28, 69, 76, 85, 92
Index	46, 59, 60, 128
Interface	1, i, ii, iii, 1-9, 11-16, 19-24, 26, 29, 35, 50, 63-65, 67, 88, 92-94, 96, 100, 120, 122, 124
Keyboard	12, 19, 22, 27, 38, 49, 50, 60, 89, 92, 93, 96, 98-101, 103, 107, 114, 127
Label	127
List	9, 26, 28, 29, 33, 35, 37-41, 50-52, 60, 71, 73, 76, 78, 84, 88, 89, 92, 96, 101, 105, 107, 115, 119, 122
List box	33, 35, 37, 39, 40, 51, 52, 96, 115, 119
Macro	108, 119
Main	ii, 3, 5, 16, 17, 19, 64, 69, 95, 104
Mark	28, 31, 58, 107, 108, 119
Maximize	20, 25, 67, 119, 126
Menu Bar	8, 10, 13, 17, 18, 23, 24, 26-29, 34, 44, 59, 60, 64, 65, 68, 69, 72, 78-82, 87, 88, 90, 91, 103, 104, 107, 119
Menus	12, 23, 93, 94, 96, 113, 119, 127
Messages	21, 22, 25, 42, 50, 53-58, 67, 83-85, 96, 99, 100
Microsoft	i, iii, 2, 3, 14, 19, 22, 93, 95-99, 102, 121
Minimize	20, 25, 111, 120, 124, 126
Mode	7, 9, 44, 111, 115, 116, 121-123, 125, 126
Modes	7, 44
Mouse	1, 4, 12, 14, 16, 19, 21, 24, 25, 27, 38, 49, 50, 53, 60, 63, 93, 94, 98, 99, 104, 105, 112, 115, 119, 120, 121, 123, 124, 127
Mouse button	12, 24, 99, 112, 115
Move	i, 10, 12, 20, 24, 25, 29, 30, 34, 41, 43, 44, 60, 63, 71, 73, 78, 81, 82, 91, 103-105, 108, 110, 113, 114, 115, 121, 123, 125
MS-DOS	93, 116, 118
Name	8, 25, 29, 33, 34, 38, 41, 47, 51, 55, 65, 68, 88, 92, 111, 114, 116, 117, 119, 127
New	i, iii, 2, 14, 16, 29, 30, 44, 49, 59, 63, 65, 76, 81, 82, 94, 95, 97, 98, 109, 125, 127
Nonprogrammable	13, 14
Number	9, 10, 28, 34, 47, 48, 66, 78, 80, 82, 92, 97, 122
Open	ii, iii, 2, 29, 34, 35, 37, 45, 51, 56, 80, 107, 111, 112, 119, 120, 123, 124, 126
Option	29, 47, 56, 68, 72, 73, 79, 87, 103-105, 107, 112, 120, 124
OS/2	19
PgDn	50, 90
PgUp	50, 89, 90
Pop-up	29, 67, 68, 74, 76-78, 80, 82-84, 87, 88, 91, 93
Presentation	iii, 3, 4, 6-9, 13, 14, 19-24, 32, 33, 39, 48, 63, 64, 96
Presentation Manager	iii, 3, 19-22
Print	29, 30, 52, 71, 95, 97, 107, 109, 114, 122, 125
Printer	33, 40, 53, 93, 97, 112, 114, 117, 120, 122-124

## U.S.A.I.D. Common User Interface Guidelines

---

Program	i, iii, 12, 19-22, 58, 68, 95, 99, 118, 121, 122
Protocol	100, 122
Quit	12, 88, 104
Read	20, 86, 96, 100, 112, 116, 123
Read-only	112, 116, 123
Record	9, 42, 108
SAA	ii, iii, 17
Save	8, 10, 12, 22, 27, 29, 41, 56, 58, 86, 90, 94, 97, 107, 108, 112, 123
Save as	29, 107
Scroll	23, 24, 34, 35, 37, 38, 41, 43, 47-50, 69, 71, 73-75, 93, 104, 119, 123, 124
Scroll bars	23, 24, 34, 37, 43, 47-50, 93, 124
Search	33, 42, 51, 52, 85, 88, 107, 108
Select	6, 8-11, 24, 25, 27, 29, 30, 32, 33, 36-38, 40, 42, 43, 45, 47, 52, 55-58, 60, 61, 63, 81, 82, 88, 107, 109, 111, 116, 119-121, 124
Set	i, ii, 3, 4, 16, 20, 33, 35-38, 45, 50, 52, 66, 79, 97, 98, 112, 117, 122, 125, 126
Shortcut	111, 119, 124
Size	18-20, 23, 25, 43, 46, 48, 49, 61, 73, 95, 96, 113, 117, 119, 121-123
System	ii, iii, 1, 3-5, 7, 8, 12, 21-25, 29, 30, 33, 42, 47, 50, 54, 57-59, 63, 67, 69, 80, 81, 85, 86, 92-95, 97, 99, 101, 111, 113, 118, 121, 125, 126
Table	1, 107
Task	ii, 5, 7, 21, 29, 52, 54, 55, 60, 95, 97, 101, 113, 116, 121
Terminal	12, 13, 63, 118, 124, 126
Text	i, 9, 13, 14, 28, 30, 31, 33, 35-39, 42-45, 50, 54-58, 63-66, 69-71, 77, 81, 83-85, 91, 96, 100, 107, 108, 113, 117-119, 122-124, 126, 127
Time	i, 2, 12, 16, 27, 41, 42, 46, 47, 50, 54, 65, 66, 81, 86, 90, 93-100, 125, 126
Title	6, 12, 23-25, 35, 36, 55, 61, 64-66, 68, 69, 77, 94, 96, 111-113, 115, 119, 120, 123, 126, 127
Title Bar	12, 23, 24, 96, 111-113, 115, 119, 120, 123, 126, 127
Undo	7, 30, 91
View	18, 24, 26, 28, 29, 31, 32, 47, 49, 64, 66, 72, 75, 79, 127
Wang	14, 89
Warning	7, 8, 29, 54-57, 67, 83-85, 91
Window	1, 6, 10, 12, 15-19, 18-25, 25, 27, 28, 31, 32, 34, 36, 40-47, 46-49, 54, 55, 59-61, 64-69, 76-78, 80-82, 87-90, 92, 96, 99, 103, 107-109, 111-113, 115, 117-120, 122-124, 126, 127
Window Title	6, 23-25, 55, 61, 64-66, 68, 69
Windows	i, iii, 2, 3, 14-25, 30, 32, 34, 35, 41, 44, 48, 49, 64, 65, 68, 81, 82, 90, 91, 93-102, 107, 111, 112, 113-127
Word	9, 17, 18, 26, 31, 69, 75, 107, 110, 112, 114, 121, 127
Workstation	19, 63
X-Windows	22, 127
Xerox	3, 5

0.1.