Design Guide

MICROSOFT®
MODULAR
WINDOWS™

Microsoft® MODULAR WINDOWS™ SOFTWARE DEVELOPMENT KIT

Design Guide

Microsoft® Modular Windows™ Software Development Kit

Microsoft Corporation

Document No. MM37743-1192
Printed in the United States of America.

Contents

Introduction

This manual, *Microsoft Modular Windows Design Guide*, provides guidelines for developing user interfaces for applications that run in the Microsoft® Modular Windows™ Operating System. The following chapters describe each component of the Modular Windows user interface and the design principles programmers and designers should use when creating applications for Modular Windows:

- Chapter 1, "User-Interface Design," presents general guidelines for designing user interfaces for consumer entertainment and information software.
- Chapter 2, "Modular Windows Design Guidelines," presents specific guidelines for designing applications for Modular Windows.
- Chapter 3, "Designing for Televisions," identifies differences between televisions and computer monitors and presents guidelines for designing software for display on televisions.

Purpose

The purpose of this guide is to introduce programmers and designers to the user-interface of Modular Windows and to promote visual and functional consistency across applications. Consistency offers many advantages to applications:

- Consistency eases the learning process and reduces the time required for users to learn how to use a new application.
- Consistency gives users a sense of stability and continuity across applications, which increases their confidence in the reliability of the application and in all applications that use a similar interface.

Scope

Because of the evolving nature of applications and the relative newness of the TV-based platforms, this guide cannot provide specific recommendations for every possible interface issue. If an application requires elements or techniques not discussed in this guide, designers may extend the existing guidelines in accordance with the general principles described in Chapter 1, "User-Interface Design."

Implementation

Many of the elements and techniques described in this guide are implemented in samples contained in the Microsoft Modular Windows Software Development Kit. See the *Getting Started* manual for a description of these samples.

Recommendations

The purpose of this guide is to promote visual and functional consistency across applications for Modular Windows. The information presented here is provided as a tool for those who would like to use it. There is no conformance requirement, expressed or implied, in this set of guidelines. However, it is to your benefit and the benefit of the users of your application to provide a consistent user interface.

Document Conventions

The following conventions are used throughout this manual to define syntax:

Convention	Meaning
Bold text	Denotes a term or character to be typed literally, such as a function name (**CreateWindow**), data-structure name (**WNDCLASS**), or resource-definition statement (**ICON**). You must type these terms exactly as shown.
Italic text	Denotes a placeholder or variable; you must provide the actual value. For example, the statement **SetCursorPos**(*X*,*Y*) requires you to substitute values for the *X* and *Y* parameters.
[ ]	Encloses optional parameters.
|	Separates an either/or choice.
. . .	Specifies that the preceding item can be repeated.
. . .	Represents an omitted portion of sample code.

The following text conventions are also used in this guide:

Convention	**Meaning**
SMALL CAPITALS	Indicates the names of keys, key sequences, and key combinations—for example, ALT+SPACEBAR.
ALL CAPITALS	Indicates filenames and paths, most type and structure names (which are also bold), and constants.
`monospace`	Indicates C- and assembly-language source-code statements.

CHAPTER 1

User-Interface Design

User-interface (UI) design is an evolving discipline, growing with advances in technology and the expansion of markets to new customers. The graphical user interface (GUI) introduced in the 1980s is based on the technology of the bitmapped video display and the mouse. In the 1990s, technologies such as digital motion video, optical media, cellular and satellite communications, and the continuing miniaturization of electronic devices will move graphical user interfaces into new computing devices quite unlike the mouse- and keyboard-based computers to which users are accustomed.

This chapter presents general user-interface design guidelines for consumer entertainment and information software. For specific information about user-interface design for Microsoft® Modular Windows™, see Chapter 2, "Modular Windows Design Guidelines."

Basic UI Design Principles

The following are some basic design principles you should follow when designing the user interface for consumer information and entertainment software:

- Use conceptual models and common metaphors
- Keep it simple
- Design for longevity
- Design to create a flow experience
- Use sensory feedback to create a tactile interface
- Allow the user to feel that they are in control at all times

The remainder of this chapter explores each of these principles in greater detail.

Using Conceptual Models and Common Metaphors

Conceptual models and metaphors are powerful aids in user-interface design. They allow users to apply real-world knowledge to understand the structure of a system and how the system operates. For example, many applications use controls based on the transport controls from consumer tape recorders and VCRs (play, record, stop, pause, fast forward, and rewind). The function of these controls is understandable to anyone and requires no explanation. Use the following guidelines in creating conceptual models:

- Create an overall conceptual model for your application
- Design your user interface based on common, familiar metaphors
- Map each control movement to an easily recognizable action

The Conceptual Model

Finding the right conceptual model is perhaps the most important task in designing a multimedia application. The overall conceptual model for an application forms the basis for the design of all aspects of the application, including the user interface. A conceptual model allows users to apply knowledge gained from experience to understand the structure and purpose of the application as well as how to use it.

The conceptual model for an application must be well established before designing the user interface. If the conceptual model changes, it's likely that most of the user interface must change as well. Generally, an application should have a single conceptual model reflecting its structure and how it is to be used.

The Living Books series from Broderbund Software ® is a good example of a simple, yet elegant, conceptual model: a book. Everyone understands what books are and how to use them. Even though the Living Books titles are more than electronic representations of printed books, using a book as a conceptual model is instantly recognizable.

Using Common Metaphors

Using common, familiar metaphors is a powerful way to create easily understandable user interfaces. For example, users understand that pencils can be used for drawing lines, brushes for painting, and erasers for erasing. Most drawing applications use these common metaphors to represent different tools users can choose. Whenever possible, design the user interface using metaphors that will be understood by all users.

Metaphors are typically represented in user-interface controls with a *glyph.* A glyph is a simple symbol that communicates information nonverbally. It's very important that the glyph be easily recognizable or the metaphor will not be understood by the user. Adding a word or two of text to a glyph can clarify what the glyph represents, but this requires additional screen space. Designers must make a trade-off between the clarity gained by adding text to a glyph and the additional screen space used by the text.

Mapping Control Movements and Results

A *mapping* is the relationship between a control (its location, appearance, and how it is activated) and the result of activating the control. It's the relationship between what a user does and what happens as a result. These mappings should be logical, natural, and consistent. Use the following guidelines when designing control mappings:

- Maintain spatial integrity; for example, a stereo pan control should be a horizontal slider or a rotary control. Moving the slider to the left or the rotary control counterclockwise should move the pan to the left speaker.
- Use quantitative, not qualitative indications; for example, size, brightness, and loudness are generally better indicators than color, pitch, or type style.

The user interface should clearly show the relationships between actions and results, and between controls and the effects of the controls.

Keeping It Simple

Simplicity is one of the hallmarks of good user-interface design. While the underlying structure of an application can be complex, understanding what the application does and how to use it should a relatively simple task for even a novice user. Use the following guidelines to maintain simplicity in your user-interface design:

- Limit features and options to reduce the number of choices a user must make
- Use constraints to prevent users from choosing inappropriate actions
- Provide default choices whenever possible
- Make possible actions and results visible to the user
- Allow users to work directly with objects; don't use abstractions
- Keep the interface consistent and predictable

Limiting Features and Options

Compared to PC applications, TV-based player applications should provide fewer features and options and require considerably less documentation. Remember the following general guidelines when porting an existing application to a TV-based player:

- Complexity increases with the number of features.
- Eighty percent of an application's value is contained in twenty percent of its features.

Using Constraints and Defaults

Constraints encourage the user to make appropriate choices by restricting inappropriate choices. For example, the control allowing a user to save or pause a game can be constrained by not being enabled until the game is actually started. Edit controls can be constrained by not being activated until an object is selected.

Whenever possible, applications should supply default choices. Default choices reduce the number of decisions users must make and allow them to get past parts of the program they don't completely understand. Whenever possible, default choices should be logical and personalized to the user of the application.

Using Visibility and Direct Control

Generally, the conceptual model of the application and all actions available to a user should be visible and obvious. Don't hide controls in the interface. The results of actions should also be visible. The user should at all times be able to figure out what to do and determine what is happening in the application.

The exception to this guideline is with entertainment applications where the challenge is to discover the hidden actions that allow progress to the next level of play. Many adventure role-playing games are examples of designs that often intentionally obscure available choices.

Users also prefer to directly manipulate objects whenever possible. For example, picking up a key and putting it into a lock to unlock a door is more direct than typing "unlock door."

Keeping the Interface Consistent and Predictable

Use consistent conceptual models throughout the application. If a control works a certain way in one part of the application, it should work the same way in other parts of the application. Similar controls should function in the same way—if one control provides auditory feedback, the user will expect all controls to do the same. Keeping the interface consistent reduces the amount of information the user must remember to use an application. If similar controls function differently, the user must remember this difference in order to use the control.

In addition to being consistent, user interfaces should be predictable. Generally, it is not good design practice to provide a *modal interface*. In a modal interface, the action of individual controls depends on the state of the system.

Designing for Longevity

The longevity of an application, or how long users remain interested in using it, is important to customer satisfaction and in determining the selling price. The following factors affect the longevity of an application:

- User interface—A poor user interface will reduce the longevity of an otherwise excellent application.
- Usefulness—Applications such as an encyclopedia or other reference database will be useful over a long period of time. References containing topical, timely information, such as an annual almanac, automatically have a shorter longevity.
- Quality and depth of content—Shallow applications will be "used up" quickly. Applications with low-quality content probably will not be used at all. Remember that users are interested in the content of your application and not the interface. A good interface allows the user to complete their task efficiently without getting in the way of the user.
- Competitiveness provided by the application—Games should provide an ongoing challenge to the experienced user while still being accessible to novice users. Challenge levels and degree-of-difficulty settings allow users to control the competitiveness of game applications.

TV-based player applications will compete directly with broadcast and cable television channels, VCRs, video-game systems, and compact-disc players. Once the novelty wears off, applications must be perceived to be easy-to-use and of high value to compete with these established consumer-electronic devices.

Creating Flow

Imagine a child playing a video game, totally absorbed in the game. Time seems to stop. The child is lost in the *flow* of playing the game. During this time, the child is engaged in a very receptive learning state, encouraged by the flow state of the game. Flow is the experience of total involvement. The following is a list of guidelines that can help increase the flow experience provided by an application:

- Refine the user interface.

  Flow is interrupted if the user must stop for too long to think about how to do something.
- Provide a clear goal to the user.

  Some new users might be accustomed to the passivity of viewing television and not be ready for interactivity and control. Allow users to choose a goal corresponding to their skill level.
- Provide sensory feedback.

  Flow requires immediate sensory feedback so that the user perceives responses to particular actions. Users should always be aware of the status of reaching their chosen goal.
- Provide the correct challenge level.

  Flow is broken if the user doesn't feel challenged or feels that the challenge is too great.
- Optimize critical performance areas.

  Flow is broken if the user must wait too long for such tasks as loading the next skill level. During long operations, you can reduce the perceived wait time by playing music or providing something interesting to view.

Time spent in the flow state can be productive as well as enjoyable. Flow is an indicator of good application design—it should be an important design goal for any interactive application whether the application is for entertainment or education.

Increasing Responsiveness

The responsiveness of an application to a user's actions is very important in maintaining flow. CD-ROM provides vast inexpensive storage, but applications need to be clever about how they retrieve information or the responsiveness of the application will be impaired. The following is a list of guidelines for designing for CD-ROM:

- Operations that last longer than one second should be interruptible with a hand-control button. You should provide a status message and provide feedback to indicate the progress of the operation. The user should always know when to wait and when to proceed.

- Minimize the time it takes to load your application. Load the application in stages; for example, you might load some introductory graphics and music first, then load the rest of the application. Allow the user to easily bypass introductory credits and help screens.
- Anticipate time-consuming CD-ROM seeks and transfers by preloading those parts of the application required for providing feedback (especially sounds).
- Thoroughly test the application with novice users to understand their perception of the application's responsiveness.

Providing a Passive Mode

It's a good idea to provide a passive mode in applications. In passive mode, the application is active without any interaction from the user. For example, a multimedia encyclopedia might randomly or alphabetically browse through its topics or provide a slide show of images contained in the encyclopedia. A game application might play a simulated opponent. This passive mode can also be used as an in-store demonstration for the product.

Using Sensory Feedback

Each action taken by a user should have an immediate and visible effect. Sensory feedback contributes to maintaining the flow experience and lets the user feel greater control over the game or application. Feedback can be either visual, auditory, or both. All user-interface controls must provide visual feedback. Users coming from the world of video-game systems will expect auditory feedback with TV-based player applications. The following is a list of guidelines for providing feedback:

- Feedback must be immediate. Users must be able to associate their actions with corresponding results. A relatively simple interactive application with a responsive interface can be more appealing than a multimedia smorgasbord that is sluggish to respond to user control.
- Feedback should be unambiguous. The user must understand the meaning of the feedback. See the section "Mapping Control Movements and Results," earlier in this chapter, for more information.
- Negative feedback is sometimes appropriate. Producing a warning sound when the user makes an inappropriate action is an example of negative feedback.

Providing feedback in an application encompasses more than feedback from user-interface controls. Users also need feedback about their progress towards a given goal, comparisons with their previous attempts, and comparisons with other players.

Allowing the User to Maintain Control

It's important that the user always feel in control of an application. A well-designed, responsive user interface contributes much towards the user's perception of being in control. The following list gives some additional related design suggestions:

- Allow users to interrupt long operations. When a user changes their mind or makes a mistake, they shouldn't have to wait for an operation to complete.
- Discard meaningless user input during long operations. While waiting, users might randomly press buttons on the hand control. Be sure to discard the input intelligently—for example, don't discard input meant to interrupt the current operation.
- Allow users to quit and later resume use of the application at the point at which they quit.
- Test your application to ascertain if users feel that they are in control.

Bibliography

For more information on user-interface design and related topics, see the following publications:

Csikszentmihalyi, Mihaly. *Flow.* New York: Harper and Row, Inc., 1990.

Laurel, Brenda. *The Art of Human-Computer Interface Design.* Menlo Park: Addison-Wesley Publishing Company, Inc., 1990.

Laurel, Brenda. *Computers as Theatre.* Menlo Park: Addison-Wesley Publishing Company, Inc., 1991.

Microsoft Corporation. *The Windows Interface: An Application Design Guide.* Redmond: Microsoft Press, 1992.

Norman, Donald A. *The Design of Everyday Things.* New York: Doubleday, 1988.

Shibukawa, Ikuyoshi, and Yumi Takahashi. *Designers's Guide to Color 5.* San Fransisco: Chronicle Books, 1991.

CHAPTER 2

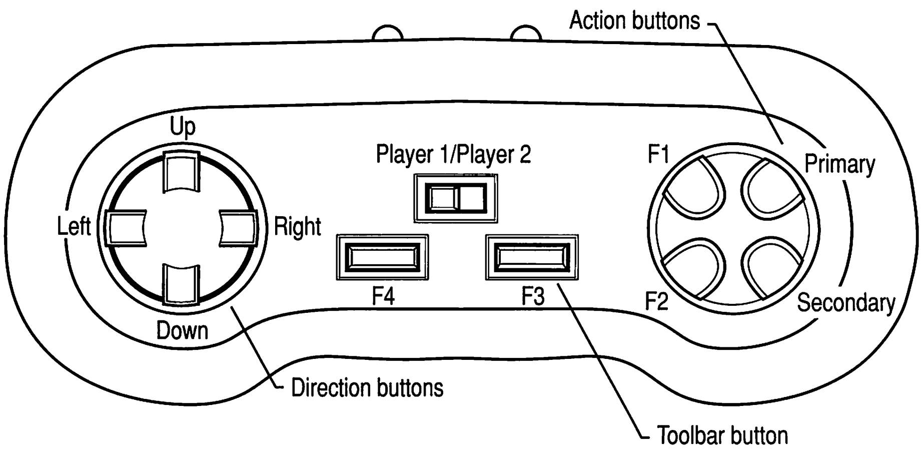
Modular Windows Design Guidelines

Designing applications for Microsoft Modular Windows presents challenges not encountered in designing applications for personal computers. From the user's point of view, TV-based multimedia players are more like video-game systems than computers. Therefore, applications for Modular Windows require a substantially different user interface than traditional PC applications.

This chapter presents design guidelines for applications with special emphasis on designing applications using the user-interface controls provided by Modular Windows. For general user-interface design guidelines, see Chapter 1, "User-Interface Design."

Understanding Input Devices for Modular Windows

The hand control is the primary input device for TV-based multimedia players. Many users will be accustomed to using hand controls provided with video-game systems and will expect similar tactile response with applications for TV-based players. For other users, TV-based players will provide their first experience using a hand control. The following illustration shows the button layout on a hand control:



Hand control for TV-based multimedia players

You should consider both groups of users when designing and programming an application. The following list outlines a few general guidelines for the use of the hand control:

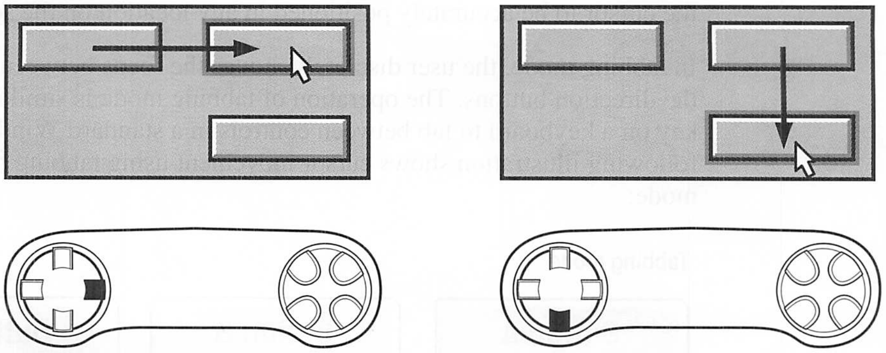
- Design the interface to use the hand control in an intuitive way.
- Provide documentation on how the application uses the hand control, which might include a quick-reference card identifying how the hand control works with the interface.
- Program the application to optimize tactile response.

See Chapter 2, "Hand-Control Services," in the *Modular Windows Programmer's Reference,* for information about programming an application to use the hand control as an input device.

Using the Hand Control with User-Interface Controls

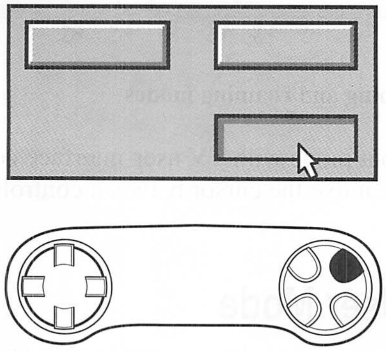
Modular Windows user-interface controls are designed for use with a hand control. User input with a hand control is a two step process:

1. The user presses the directional control buttons to move the cursor between controls, as shown in the following illustration:



Using the direction buttons to move the focus between controls

2. The user then presses the primary action button to initiate an action associated with the control:

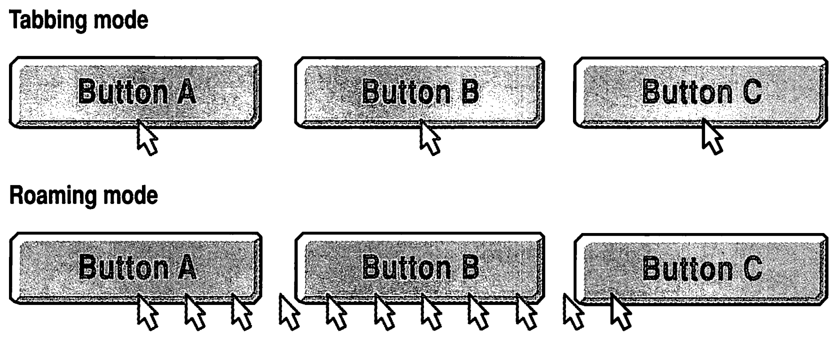


Using the action button to choose a control

Moving the Focus Using Roaming and Tabbing Modes

Modular Windows provides two modes for moving the cursor between controls: *roaming mode* and *tabbing mode*. In roaming mode, the user positions the cursor over a control by pressing the direction buttons to move the cursor in the corresponding direction. Positioning the cursor in this manner is an iterative process requiring several rounds of horizontal and vertical movement, but allows the cursor to be accurately positioned in any location on the screen.

In tabbing mode, the user discretely moves the focus between controls by using the direction buttons. The operation of tabbing mode is similar to using the TAB key on a keyboard to tab between controls in a standard Windows dialog box. The following illustration shows cursor movement using tabbing mode and roaming mode:



Cursor movement using tabbing and roaming modes

Tabbing is the preferred input mode with TV user-interface controls. It is much easier for users to discretely move the cursor between controls than to iteratively position the cursor.

Compound Focus and Power-User Mode

Some of the TV user-interface controls have compound focus, or more than one area that can have the focus. For example, the TV scroll-bar control has focus areas for up and down movement (vertical style) or left and right movement (horizontal style). Controls with compound focus include the TV scroll-bar, TV scroll-pad, TV list-box, and TV spin-button controls.

All of the controls with compound focus have a power-user mode that allows faster input using the hand control. Power-user mode is initiated when the user holds the action button down while pressing one of the direction buttons. As long as the action button is pressed, pressing a direction button not only changes the focus, but also simulates a button press on the action button. For example, with a scroll-bar control, a scroll event occurs each time a direction button is pressed while the action button is down.

Drag and Drop Guidelines for the Hand Control

Users prefer direct manipulation of objects whenever possible. For example, to make a move in a chess game, the player can select and move the chess piece directly instead of entering coordinates for the new position.

▶ **To implement the drag and drop operation with a hand control:**

1. Allow the user to move the focus to the object and to select it using the action button.
2. Indicate the selection with a visual cue, such as outlining or flashing the object.
3. If the user presses the action button before moving the object, cancel the selection of the object.
4. Move the selected object in the indicated direction as the user presses the direction buttons.
5. When the user presses the action button, place the object and cancel the selection of the object.

Identifying Hand-Control Buttons in Documentation

In the documentation for the application and in online help, you should present a diagram of the hand control illustrating how the application uses the various hand-control buttons. Don't rely on colors to identify buttons to users; button colors might vary between different TV-based players. Instead, use the following names to identify hand-control buttons:

Button Name	Description
Up	Up-direction control button
Down	Down-direction control button
Left	Left-direction control button
Right	Right-direction control button

Continued

Button Name	**Description**
A	Primary action button
B	Secondary action button
1	Function button 1
2	Function button 2
3	Function button 3 (toolbar button)
4	Function button 4

The TVVGA font includes special characters that can be used in online documentation to identify hand-control buttons. See "Button Accelerators," later in this chapter, for details on these special characters.

Mapping Unused Buttons

Most applications will not use all of the hand-control buttons, so you might consider mapping the unused buttons to other buttons that the application uses. For example, if the application does not use the secondary action button, it can be mapped to the primary action button. If the 4 button is not used, it can be mapped to the 3 (toolbar) button.

Designing the User Interface

The user interface is probably the most critical element of a TV-based multimedia player application—indeed, the user interface might be the key to the success or failure of an application. Users of TV-based players will range from people who are experienced with the latest computer software technology to people who have never used a computer to children who are accustomed to using video games. The user interface must be intuitive to inexperienced users, yet allow advanced users full access to the capabilities of the application.

The standard user-interface model of drop-down menus and multiple overlapping windows was designed to be used with a mouse and a keyboard as input devices. Much of this standard user-interface support is not available in Modular Windows—it has been replaced with a simpler user interface optimized for use with a hand control and designed to be displayed on a television.

The following elements of the standard Windows user interface are **not** available in Modular Windows:

- Menus, including system menus
- Movable, sizable windows
- Minimize/maximize buttons
- Non-client scroll bars
- Controls based on predefined window control classes
- Multiple Document Interface (MDI)
- Common dialog-box library

These user-interface elements, used in most computer applications, are too complex for users of TV-based players. Applications written for TV-based multimedia players must be designed with a very simple user interface.

Using Multiple Overlapping Windows

Because Modular Windows does not support a user interface for moving, resizing, minimizing, or maximizing windows, applications using multiple overlapping windows must implement their own user interface to handle these window-management activities. Generally, applications should avoid using multiple overlapping windows.

The focus manager does allow tabbing between child-window controls located in different parent windows. See Chapter 1, "User-Interface Controls," in the *Modular Windows Programmer's Reference*, for details on using the focus manager.

Using Dialog Boxes

Modular Windows provides full support for dialog boxes. Use the following guidelines when designing dialog boxes for TV-based player applications:

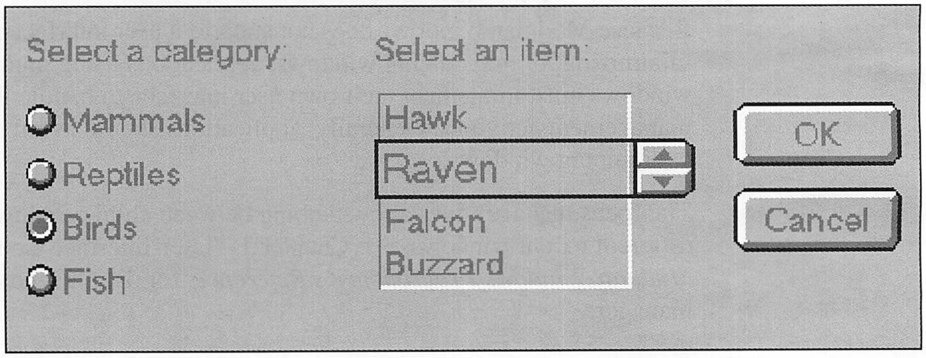
- Generally, applications should use modal dialog boxes. With modal dialog boxes, users must respond to the dialog box before continuing.
- If applications use non-modal dialog boxes, they must properly manage focus changes between dialog boxes and other windows. See Chapter 1, "User-Interface Controls," in the *Modular Windows Programmer's Reference*, for details on using the focus manager.
- Use caption bars for dialog-box applications.

Presenting a File-Selection Dialog Box

Modular Windows does not provide support for common dialog boxes such as the Open dialog box. Applications must create their own custom dialog boxes to allow users to select items stored on disc. Use the following guidelines when designing item-selection dialog boxes:

- Don't present users with standard 8.3 MS-DOS filenames, such as FILENAME.EXT. Instead, use descriptive text to identify files.
- Don't use the following terms: drive, file, directory, folder, or path.
- Don't use current directory (.) or parent directory (..) symbols in the interface.
- Don't expose users to a hierarchical file system. Instead, present categories of similar items.

The following illustration shows an example item-selection dialog box. This dialog box uses radio buttons to select a category and a list box to select an item from the chosen category.



Example of an item-selection dialog box

See "Using Modular Windows User-Interface Controls," later in this chapter, for information about list boxes, radio buttons, and other controls.

Providing Auditory Feedback

Users of TV-based players will expect auditory feedback from applications. Many users will be accustomed to video-game applications that fully use sound. Consider providing auditory feedback for the following events:

- When the user presses an *active* button on the hand control. An active button is any button that results in an action by the application. Buttons not used by the application should not produce sound.
- When the application is loading and during any extended period of disc access.

Be careful not to use too many different sounds as audio cues—users might have trouble distinguishing between a large number of different sounds. Also, you might want to provide users with a way to disable auditory feedback.

Using Text

When using text in an application, remember that the resolution of a television is not as sharp as a computer monitor. In addition, users might be viewing the application from a distance of four to twelve feet, or more. To use text effectively in an application, consider the following guidelines:

- Make text large enough to be legible on even a small television.
- Ensure text is legible for distant as well as close-up viewing.
- Use a sans-serif, proportional font.
- Use non-saturated colors to prevent color bleeding on the screen.
- Don't use a high-contrast background with text to prevent the text from smearing. Dark-gray text on a light-gray background works better than black text on a white background.

Modular Windows provides a sans-serif, proportional font in ROM. The font is named TVVGA and comes in two sizes, 14 and 18 point. All of the TV user-interface controls use the TVVGA font by default.

Reporting Errors

The only errors that applications for Modular Windows should report to users are errors users can easily remedy. Examples of such errors include not having the correct disc or memory cartridge inserted.

Using Modular Windows User-Interface Controls

Modular Windows provides the following user-interface controls, optimized for operation with a hand control and for display on a television:

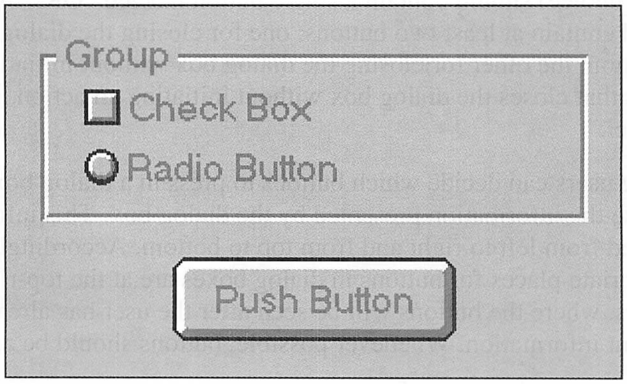
Control	Action
TV push button	Initiate an action.
TV check box	Turn an option on or off.
TV radio button	Choose one of several options or selections.
TV group box	Visually associate related controls. TV group boxes don't accept user input.
TV scroll bar	Position information displayed in a window or allow selection of a value from a range of values.
TV gauge	Display position or progress information. TV gauges are display-only controls and don't accept user input.
TV list box	Select an item from a list.
TV scroll pad	Horizontally and vertically position information displayed in a window.
TV spin button	Increment and decrement a value or selection.
TV static control	Display text or icons. TV static controls don't accept user input.
TV show box	Display text and bitmaps within a 3-D frame. TV show boxes don't accept user input.
TV keyboard and TV edit box	Enter text using a hand control.

TV Button Controls

Buttons are graphical controls that initiate actions and change data-object properties or the interface itself. Users can choose buttons by positioning the cursor over the button control and pressing the action button. The TV button control is similar to the standard Windows button control with the following changes:

- The TV button control appearance is optimized for display on televisions.
- Applications can customize the button appearance by changing colors and supplying bitmaps for different button elements.

The following illustration shows the appearance of TV push-button controls along with other styles of TV button controls—the group-box, check-box, and the radio-button:



TV button controls: group-box, check-box, radio-button, and push-button

Three-dimensional buttons should be shown in the pressed position whenever the specified property is in effect. When a button is inactive (that is, when the user cannot choose it because the associated action or setting is unavailable), the button label should be dimmed. The button controls available are push buttons (also known as command buttons), radio buttons (also known as option buttons), and check boxes.

Push Buttons

A push button is a rectangular shape containing a label that specifies the action or response represented by the button. Button labels are usually textual, but graphical labels can also be used. Graphical labels are particularly appropriate when the function of the button cannot be concisely represented with a textual label.

The user can choose a push button by pressing the action button while the cursor is over the button control. The action associated with the push button is initiated when the button on the hand control is released.

If the function of a push button changes depending on the state of the application, its label should change accordingly. For example, once the user performs any action that cannot be canceled, the label on the Cancel button should change to reflect that the action cannot be undone.

Using Push Buttons in Dialog Boxes

Every dialog box should contain at least one button that closes the dialog box. Message dialog boxes that require only an acknowledgment from the user (rather than a choice) should contain a single button labeled "OK." All other dialog boxes should contain at least two buttons: one for closing the dialog box and initiating the action, the other for closing the dialog box without initiating any action. The button that closes the dialog box without initiating an action is usually labeled "Cancel."

Before users can decide which buttons to press in a dialog box, they must first analyze the information presented by the dialog box. This information is usually scanned from left to right and from top to bottom. Accordingly, the most appropriate places for buttons in dialog boxes are at the top-right or bottom-left corners, where the buttons will be seen after the user has already scanned the relevant information. Whenever possible, buttons should be arranged as follows:

- Stacked along the right border of the dialog box starting in the top-right corner. In this case, buttons should generally be the same width—as wide as necessary to accommodate the longest button text. If there is an OK button, OK and Cancel should be grouped together, separated from other action buttons. If there is no OK button, Cancel should be grouped with other action buttons.
- Across the bottom of the dialog box, grouping the buttons according to action. Normally the buttons should all be the same width, but individual buttons can be made wider to accommodate long text.
- If an OK button is present, place the Cancel button immediately after it; otherwise, place the Cancel button after all other buttons.
- Other arrangements can be used if there is a compelling reason, such as a natural mapping relationship. For example, it would make sense to place buttons labeled "North," "South," "East," and "West" in a compass-like layout.
- If the message requires the user to make a choice, the dialog box should contain a button for each option. The clearest way to present several actions is to ask the user a question and provide a button for each response. When possible, questions should require a Yes or No button choice.
- Message dialog boxes can contain Help buttons to allow the user to obtain further information or suggestions. However, the interface should be simple enough that the user doesn't need extra help to interpret options or messages. If used, the Help button is placed after all other buttons so that it will be located near the bottom-right corner of the dialog box, where it will be visible to users after they scan the dialog box.

Radio Buttons

Radio buttons (also known as option buttons) represent a single choice in a limited set of mutually exclusive choices. Accordingly, in a group of radio buttons, the user can only select one button at any time. Radio buttons are represented by circles. When an option is selected, the circle is filled; when the choice is not selected, the circle is empty. If the number of radio buttons in a group exceeds four, the buttons can be replaced by a list to save space. If space is not at a premium, however, radio buttons provide easier access to choices.

A group of radio buttons can be used to choose among a fixed set of attributes for a selection. Whenever the user makes a new selection, the radio buttons corresponding to the current attribute should be filled and the other radio buttons should remain empty.

Check Boxes

Check boxes control individual choices that are either selected or cleared. When the choice is selected, the check box contains an X. When the choice is cleared, the check box is blank. If you need to provide a quick way to clear all check boxes in a group, you can add an additional check box to the group that performs this function.

Check boxes can be used to set properties of a selection. As with radio buttons, check boxes should correctly reflect the properties of each new selection.

As with buttons, when a check box is inactive, its label should be dimmed.

Group Boxes

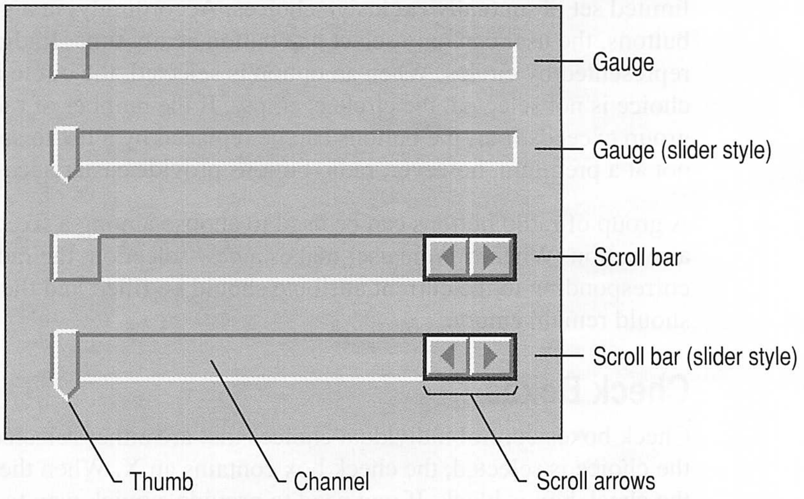
Group boxes, though technically considered controls, don't process any input. They can be used to provide visual grouping of related controls. Group boxes consist of a rectangular frame with a label at the upper-left corner.

TV Scroll-Bar and TV-Gauge Controls

The TV scroll-bar control is similar to the standard Windows scroll-bar control with the following changes:

- There is a new style of scroll bar called a *gauge*. Gauges don't accept user input.
- Applications can supply bitmaps and change colors to customize the appearance of scroll bars and gauges.

The following illustration shows the appearance of TV scroll-bar controls (only horizontal controls are shown):



TV scroll-bar controls

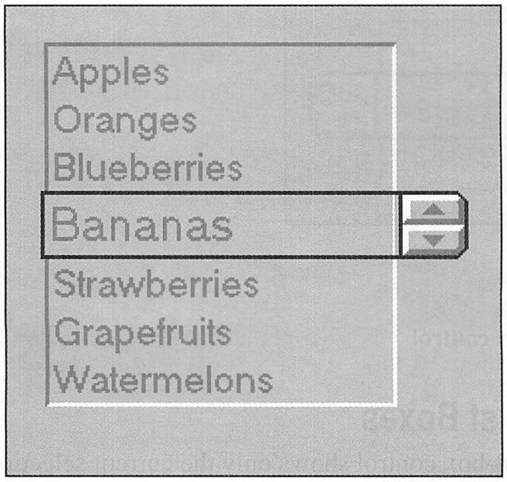
Gauge controls are used to display and adjust values on continuous dimensions such as pitch, loudness, and brightness. A gauge control consists of a bar containing notches or measurement markings, plus an indicator perpendicular to the bar. The indicator shows the present value and can be dragged along the bar with the cursor to set a new value. See "Drag and Drop Guidelines for the Hand Control," earlier in this chapter, for more information. Gauges can be positioned horizontally or vertically.

TV List-Box Control

The TV list-box control is similar to the standard Windows list-box control with the following changes:

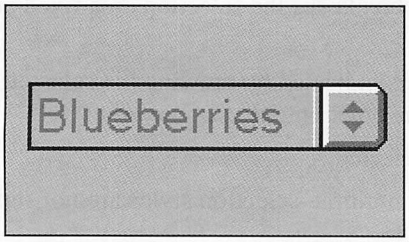
- Instead of keeping list-box entries in a fixed location while moving the highlight, the TV list box keeps the selection area fixed and moves the list-box entries.
- Selection of an entry is indicated by drawing the selected text or bitmap slightly larger than the other entries.
- One item is always selected.
- Multiple-column and multiple-selection styles are not supported.

The following illustration shows the appearance of a standard TV list box (the current selection is "Bananas"):

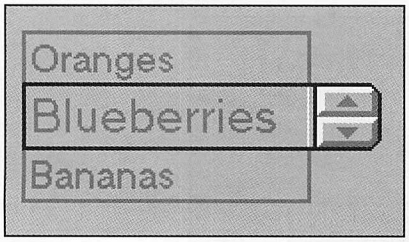


Standard TV list-box control

There are two variations of the standard TV list box: popup and spin field. The popup list-box control displays only the current selection until it gets the focus and is activated by the primary action button. Once activated, it expands to appear like a standard list box. The following illustration shows the appearance of a popup list box before and after it is activated:



Before activation

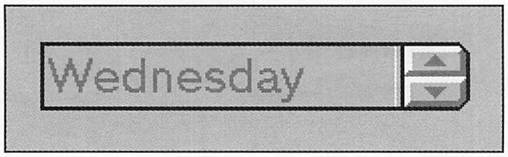


After activation

Popup TV list-box control

Spin-Field List Boxes

The spin-field list-box control shows only the current selection—it does not expand to show other entries like the popup list box. Spin-field list boxes should be used when the list-box entries are obvious to the user, such as for days of the week or months of the year. The following illustration shows a spin-field list box:



Spin-field list-box control

Using List Boxes

List boxes are used to display choices for the user. The choices can be represented by text, color, or graphics (bitmaps or graphics objects).

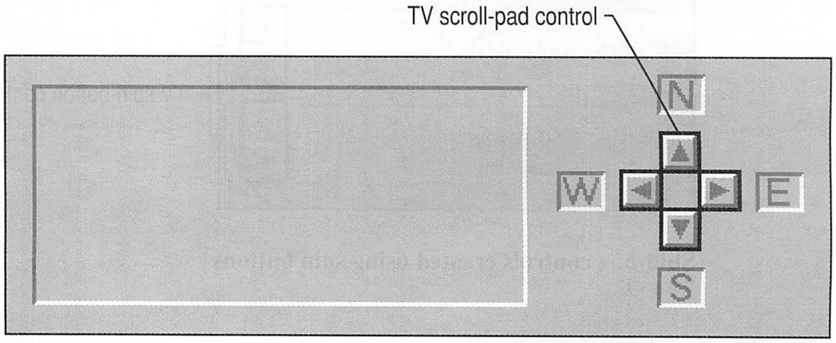
If a particular choice is not available in the current context, it should generally be omitted from the list. However, if it is important to communicate to the user both the existence and the current inaccessibility of a list item, the item can be dimmed rather than omitted.

Because single-selection lists allow the user to select only one item, the items in the list are functionally similar to radio buttons. Applications should use single-selection lists rather than radio buttons when the size or composition of a set of choices is variable, when the set of choices is large (more than four or five items), or when space or layout considerations make radio buttons impractical.

The currently selected item in a list box will appear slightly larger. To select another item, you use the direction buttons on the hand control to position the cursor over the scroll arrows. Pressing the action button moves the list in the indicated direction, thereby changing the selected list item.

TV Scroll-Pad Control

The scroll-pad control allows both horizontal and vertical scrolling by using a single control. The following illustration shows the appearance of the TV scroll-pad control:



TV scroll-pad control

The scroll pad has four focus areas indicating up, down, left, and right scroll directions. Pressing the direction buttons on the hand control moves the focus between these four areas. Pressing the action button initiates a scroll event in the indicated direction.

Applications should provide scroll pads for all windows in which the size of the data can exceed the size of the window in both the horizontal and vertical dimensions. Scroll arrows appear at each end of the scroll pad, pointing in opposite directions away from the center of the scroll pad. The scroll arrows point in the direction in which the window "moves" over the data in the window, as if the data is fixed.

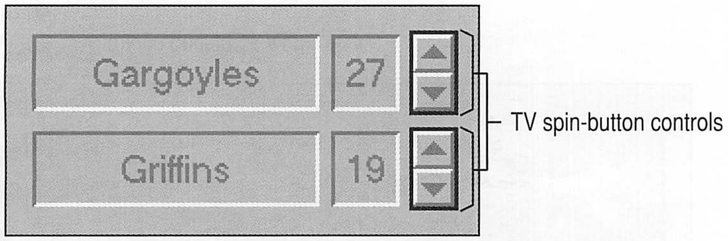
When the user chooses a scroll arrow, the data in the window appears to move in the opposite direction of the arrow (for example, one line for text applications). When the window cannot be scrolled any further in a particular direction, the associated scroll arrow should be dimmed.

TV Spin-Button Control

The TV spin-button control allows users to indicate an incremental value or position change. Spin-button controls are usually used as part of a compound control, such as a scroll-bar or a spin-box control.

Spin boxes are specialized text boxes that accept only a limited set of discrete, ordered input values. A spin box consists of a text box (implemented with a static or show-box control) along with a spin-button control attached to the side of the text box.

The following illustration shows the appearance of spin-box controls created from two show-box controls and a spin-button control:



Spin-box controls created using spin buttons

The user can position the cursor over the upward-pointing arrow and press the action button to increase the value, or position the cursor over the downward-pointing arrow and press the action button to decrease the value. In effect, the arrows function like scroll arrows for a hidden list of values sorted in descending order (in contrast to actual list controls, in which entries are sorted in ascending order).

Spin boxes can be used to display values that consist of several subcomponents (for example, time, which consists of hours, minutes, and seconds). In such cases, the text box is divided into several subfields, with each subfield separated by suitable separators (for example, "/" for date). The arrows affect the selected subfield; if no subfield is selected, the arrows affect the subfield representing the smallest unit of measurement.

TV Static Control

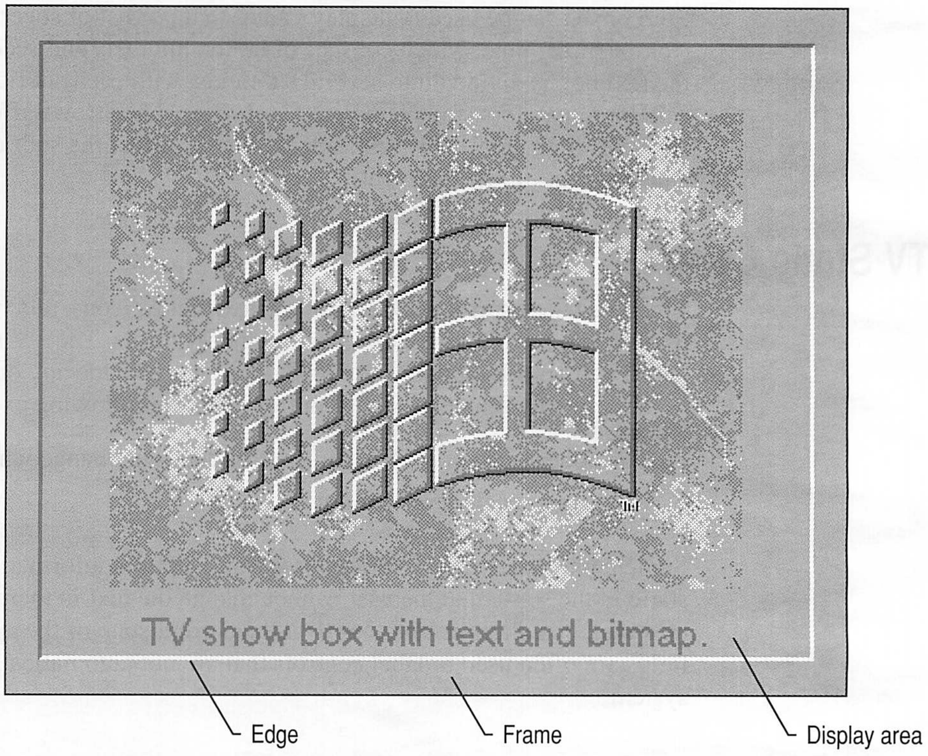
The TV static control provides simple text fields, icons, and rectangles that can be used to label, box, group, or separate other controls. The TV static control is not an active control—it does not accept input nor provide any output. It is identical to the standard Windows static control, with the following exceptions:

- The appearance of a TV static control does not change when it is disabled.
- Keyboard accelerators are not supported.

Static controls are used to present read-only textual information. These fields are static in the sense that the user cannot change the text in them. The application, however, can alter the text to reflect the current state of the application. Static text fields are often used to label controls that are not automatically labeled by the system.

TV Show-Box Control

The TV show-box control provides a single line of text and a bitmap within a 3-D frame. It is not an active control—it does not accept input nor provide any output. The following illustration shows a TV show-box control containing a bitmap and text:



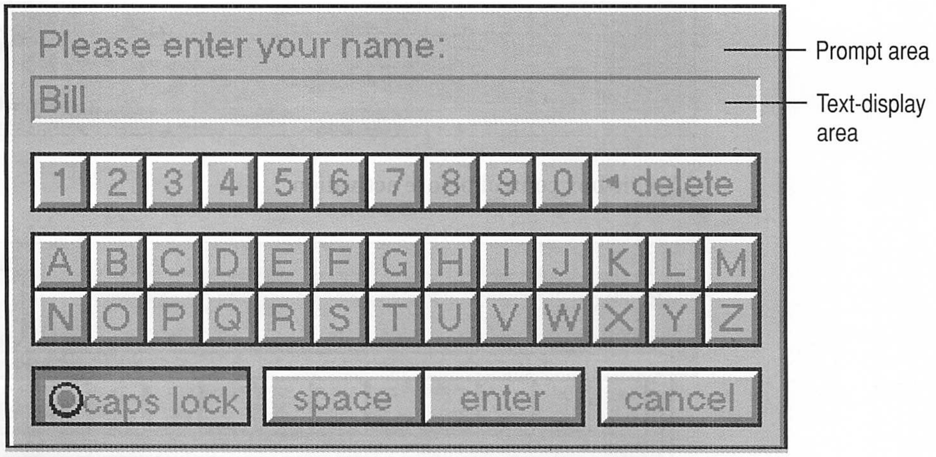
TV show-box control

As shown in the previous illustration, TV show-box controls have three components: a frame, a 3-D edge, and a display area. The frame and edge areas are optional, depending on the style given when the show box is created.

TV Keyboard Control

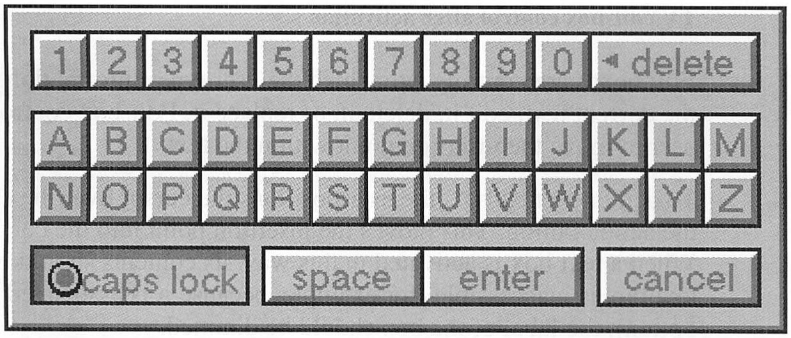
The TV keyboard control allows users to enter text using only a hand control. The TV keyboard control has two basic styles: with and without a prompt and text display area.

The form of the TV keyboard control with a prompt and text display area allows a user to edit a single line of text and then choose the Enter or Cancel button. The application can also provide a line of text prompting or querying the user to supply a particular action. When notified, the application can then retrieve the text from the keyboard control and either destroy the control or reactivate it and provide another prompt. The following illustration shows the appearance of a TV keyboard control with a prompt and text display area:



TV keyboard control with prompt and text display

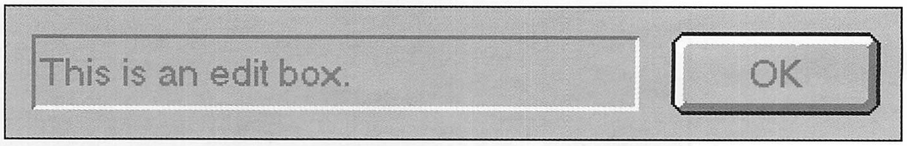
The basic TV keyboard control does not have a text display area—characters are sent to the application as they are entered. The application is responsible for displaying the text as it is entered. The following illustration shows the appearance of the basic TV keyboard control:


1 2 3 4 5 6 7 8 9 0 delete
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
caps lock
space
enter
cancel

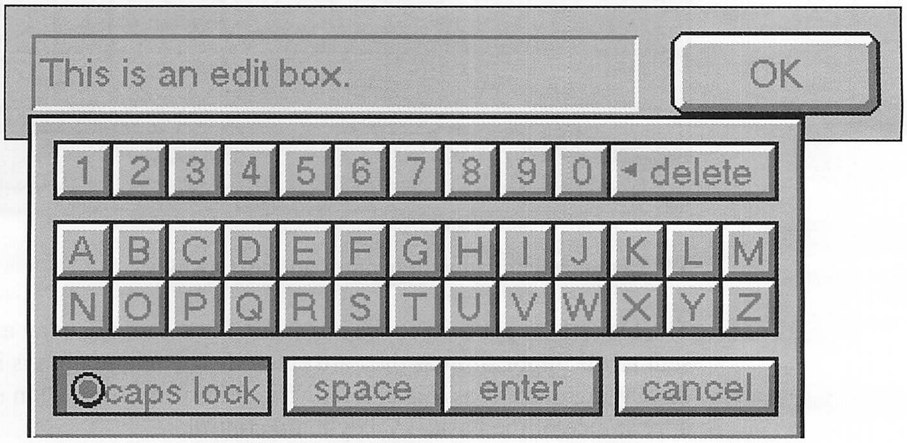
Basic TV keyboard control

TV Edit-Box Control

The TV edit-box control appears as a single line of text and, when activated, displays a TV keyboard control that allows the user to enter and edit text. The edit-box control is activated when the action button on a hand control is pressed while the edit-box control has the focus. The following illustrations show an edit-box control before and after it is activated:



TV edit-box control before activation


This is an edit box.
OK
1 2 3 4 5 6 7 8 9 0 ◂ delete
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
caps lock
space
enter
cancel

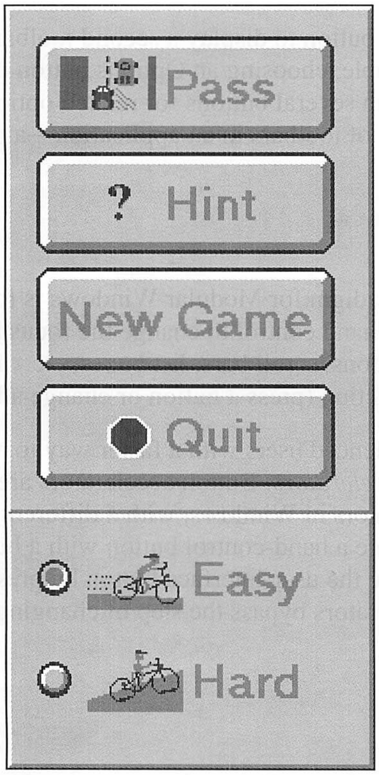
TV edit-box control after activation

Edit boxes are controls into which the user types information. The user can accept the current text, edit it, delete it, or replace it. Edit-box controls don't allow the user to move the insertion point within the text.

To activate a text box, the user positions the cursor over the edit box and presses the action button. This moves the insertion point into the contents of the box. When a text box is activated in this way, its contents should be highlighted. If a text box is inaccessible (for example, because it is associated with an unselected option), the label of the box should be dimmed.

Toolbars

Modular Windows does not support conventional drop-down or popup menus. Applications should be designed to use a *toolbar* if they need to present a hierarchical command structure. A toolbar is a set of buttons in a popup window. Toolbar buttons usually contain bitmaps and text and can be drawn in a horizontal or vertical row or in a palette-style configuration. The following illustration shows an example of a toolbar:



Example of a vertically-oriented toolbar

The appearance and the implementation of toolbars is left to the designers and programmers of the application—Modular Windows provides no direct support for toolbars other than the TV user-interface controls. Most applications will require a toolbar, if only to allow the user to quit the application. The following is a list of guidelines to follow when designing a toolbar:

- If the toolbar is not always visible, the 3 button on the hand control should alternate between showing and hiding the toolbar.
- Toolbars should have a Quit button to allow the user to quit the application.
- Toolbars should have a Help button to allow users to get online Help information.

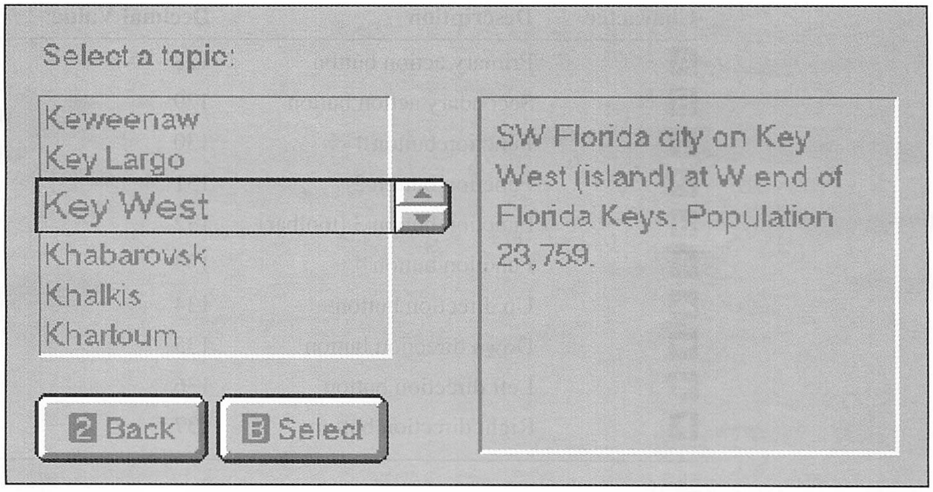
It's acceptable for a toolbar button to display a second toolbar with additional related selections. For example, choosing an Options button on a toolbar might display a second toolbar with several buttons for related options. You should minimize the number levels of toolbars in an application—a maximum of two levels is recommended.

Button Accelerators

The basic user-interface paradigm for Modular Windows is for the user to press the direction buttons on the hand control to change the focus between various user-interface elements (buttons, scroll bars, list boxes, etc.) and then use the action button to initiate an action (press a button or change a list-box selection).

You can also provide experienced users with a faster way to activate certain controls by using *button accelerators*. Button accelerators are conceptually similar to the menu accelerators in Windows, with a different implementation. Buttons accelerators associate a hand-control button with a button in the user interface—users can activate the user-interface button by pressing the associated hand-control button. Accelerators bypass the step of changing the focus between control elements.

A browser is a good example of an interface that benefits from button accelerators. The following illustration is an example of a simple browser interface:


Select a topic:
Keweenaw
Key Largo
Key West
Khabarovsk
Khalkis
Khartoum
SW Florida city on Key West (island) at W end of Florida Keys. Population 23,759.
2 Back
B Select

Example of a browser dialog box with button accelerators

To use this browser, the user presses the direction buttons and the primary action button to scroll through the topics in the list box. To make a selection from the list box, the user moves the focus to the Select button and presses the action button. Alternately, the user can make the selection by pressing the B (secondary action) button on the hand control. To move up in the hierarchy of topics, the user presses the 2 button on the hand control.

The TVVGA font includes special characters that can be used in button text to identify button accelerators. The following table shows these special characters along with their corresponding character codes:

Character	Description	Decimal Value	Octal Value
$\boxed{A}$	Primary action button	128	200
$\boxed{B}$	Secondary action button	129	201
$\boxed{1}$	Function button 1	130	202
$\boxed{2}$	Function button 2	131	203
$\boxed{3}$	Function button 3 (toolbar)	132	204
$\boxed{4}$	Function button 4	133	205
▲	Up direction button	134	206
▼	Down direction button	135	207
◀	Left direction button	136	210
▶	Right direction button	137	211

Applications should never define accelerators for the primary action button or the four direction buttons. These characters can also be used in online documentation to refer to hand-control buttons.

Managing Focus Changes Between Applications

Applications can execute other auxiliary applications (such as a sound mixer to allow the user to alter the audio mix) using the **WinExec** function. However, if an application does execute another application, it is responsible for managing focus changes between the main application and the auxiliary applications. Generally, when an application executes an auxiliary application, control should not return to the application until the user exits the auxiliary application.

Launching and Exiting an Application

When launched, applications should perform an action that indicates to the user that the application is loading. A screen introducing the application along with some music can be very effective for this purpose. MIDI music is a good choice because of its limited data-transfer requirements. If the application takes longer than 20 or 30 seconds to load and initialize, it's a good idea to provide the user with the status of the loading process. As soon as the application is loaded and ready to be used, the application should notify the user (if it is not clear from the interface) that it is ready to be used.

Writing a Custom Multi-Application Launch Shell

If a single disc includes multiple applications (such as a collection of video games), the disc can either use the ROM-based shell or supply a custom shell to launch the various applications. The launch shell is a simple application for Windows that uses the **WinExec** function to run applications selected by the user. The shell should be designed using the following guidelines:

- The shell should load quickly.
- The shell should provide an icon along with the title of each application.
- The shell should allow the user to get a description of each application.
- Optionally, the shell should play background music or an animation sequence until the user makes a selection.

Exiting an Application

There are several ways users can exit TV-based player applications: by ejecting the disc from the player, by choosing an application-supplied "Quit" button, or by turning off the power switch on the player. Applications may not notified when they are terminated—they must be prepared to be exited without notice.

Ejecting the disc from the player will probably be the way most users will choose to exit applications. Applications such as interactive games that provide users with the capability of saving the application state before exiting should provide a Quit button on their toolbars. For more information, see "Toolbars," earlier in this chapter.

When a user exits an application by ejecting the disc from the player, the application does not receive any notification that the user is exiting (it does not receive a WM_QUIT message). Applications should save user-preference information at the time the preferences are set by the user. The application should not rely on an orderly shutdown procedure to save user-preference information.

Note Tandy® VIS™ players provide a BIOS function that allows applications to prevent the user from exiting the application when ejecting the disc from the player.

Using Supplemental Data Discs

All executable program files, including dynamic-link libraries, must reside on the same CD-ROM disc. Applications can access additional discs containing data by prompting the user to insert the desired disc.

▶ **To prompt the user for a disc:**

1. Use the **SetErrorMode** function to enable the application to process critical and file-open errors (by default, Windows processes these errors).
2. Prompt the user for the desired disc.
3. Read the volume label to verify that disc is the correct one.
4. If an incorrect disc has been inserted, eject it and prompt the user again.

Be sure all code segments required to read and process the data contained on data discs are loaded before ejecting the program disc.

Storing Multimedia Data

Use the Resource Interchange File Format (RIFF) to store multimedia data used by an application. RIFF files can contain chunks useful for providing a textual description of the contents of the file, as well as copyright information, revision history, and other information. WAVE files (waveform audio) and AVI files (audio video interleaved) are examples of RIFF files. See the *Multimedia Programmer's Reference* in the Windows Software Development Kit for details on RIFF.

CHAPTER 3

Designing for Televisions

TV-based multimedia players use a standard home television as a display device. While Modular Windows video drivers provide device-independence to programmers, designers must still consider the differences between computer monitors and televisions. This chapter identifies these differences and presents solutions and guidelines for designing applications for display on televisions.

About Television Signals

The National Television Systems Committee (NTSC) television broadcasting system was adopted in 1953 for general use in the United States, Japan, and much of South America. Since then, various modifications to the NTSC system have produced other systems, such as PAL (widely used in Europe) and SECAM, which both provide improved picture quality.

The NTSC system uses three fundamental colors: red, green, and blue. By combining these colors in correct proportions, every color in the spectrum is available. It's important to use these colors appropriately. On a computer monitor, you can select any color you want to use, and you can use colors in almost any combination you want. However, when you're using a television set as a display, you should use colors that work within the limited bandwidth of the television signal.

The NTSC system provides only a 6 MHz bandwidth for each television channel. NTSC divides the video signal into three components: Y, I, and Q. The Y component represents luminance; I and Q represent chrominance. Luminance, or intensity, refers to how much light is in the color—luminance is zero for whites, grays, and blacks, low for pastel colors, and usually high for pure colors. Chrominance refers to the level of color—bright red or dark red, for example. Chrominance also denotes the hue of the color.

Picture Quality Considerations

The quality of a picture on a television depends on many factors. When you develop an application that will be displayed on a television, make sure you test the application on a wide range of televisions with a wide range of color accuracy.

Adjusting Televisions

When you create an application that uses color, it's important to make sure the television is displaying the colors you expect from it. There are several sources of test material available from video sources. You can also use real objects to further verify the color response of particular system.

The following are a few tools you can use to check the quality of color reproduction on the television:

- A mechanical color-bar generator. This device produces streams of pure red, green, and blue (RGB). Make sure your television displays these colors correctly.
- Software that generates color bars. You can use the output of the program to make sure your television correctly displays the colors you have selected.
- Videotaped color bars. Your television station displays color bars just before coming on the air or going off the air. You can record these bars onto a video tape and then play the tape back through the television that you want to test for color accuracy.
- Printed color-bar charts. You can purchase these charts, position them in front of a properly adjusted camera and feed the video signal from the camera into the television.
- Your personal preferences. Every individual has different preferences for color settings. You should adjust the signal on the television until it looks right to you.
- Testing. Don't rely on the color accuracy of the viewer's television—instead, test your application on several televisions so that you know different colors are reproduced satisfactorily on different televisions. If the test television has brightness and contrast controls, try them at different settings as you test.

The following procedure can be used to adjust the television for the best picture.

▶ **To adjust the television during testing:**

1. Send the best available video signal to the television. Try to find a signal that uses flesh tones (live shows such as newscasts are good sources).
2. Adjust the color and tint (hue) controls to produce a black-and-white picture.

3. Adjust the brightness and contrast to get crisp whites, solid blacks, and smooth gray transitions in between.
4. Turn the color control to its maximum setting, then adjust the hue control to get the best possible picture.
5. Adjust the color control until the picture looks good.

Display Artifacts Introduced by Televisions

Using some high-contrast color pairs in certain ways can produce undesirable effects on a television. These effects appear because of the limited bandwidth available for transmitting color information. This section describes some issues related to using adjacent high-contrast colors.

Color at Edges of Black and White Stripes

Alternating vertical stripes of black and white can produce colors at the stripe edges. This phenomenon occurs because the change in luminance from a black stripe to a white stripe is too great to be transmitted within the allotted NTSC bandwidth. To avoid this problem, you should avoid using extremely white whites and extremely black blacks. You can also use a very thin gray border between stripes. This creates two transitions (black-to-gray and gray-to-white) rather than a single, drastic black-to-white transition. Whenever possible, avoid vertical stripes of black and white.

Horizontal stripes generally don't produce this type of distortion.

Chroma Crawl

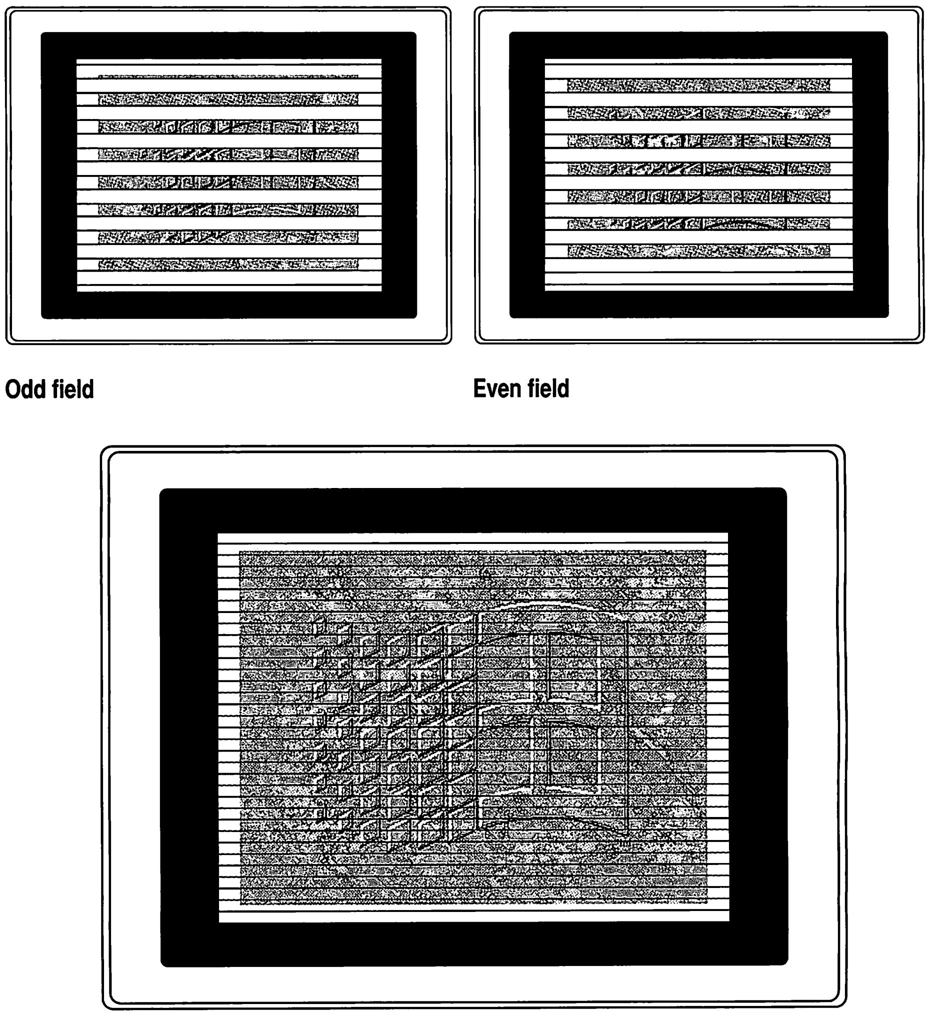
If you look at a brightly-colored object on television, you might notice a dark spot, usually near the edge of the object, that tends to move or crawl slowly up the edge of the object. This phenomenon is called *chroma crawl.* Vertical-color areas adjacent to each other are particularly susceptible to chroma crawl.

Chroma is the amount of color in a given area compared to a similar white area. For example, red has high chroma while gray has low chroma.

If you experience chroma crawl in your application, try choosing slightly different shades of color. This will change the contrast between adjacent colors, reducing chroma crawl. Experiment with shades of red in the application—red is particularly susceptible to chroma crawl.

Flicker

A television picture is basically a sequence of images, or frames. Each frame is composed of a consistent number of horizontal stripes, called *scan lines.* When less scan lines and lower frame rates are used, the eye can easily detect the transition from one frame to another. The result is screen flicker, in which the screen flashes visibly. A technique called *interlacing* increases the effective frame rate so that flicker is not perceptible. Interlacing is the process of combining two parts of an image to create a single picture, as shown in the following illustration:



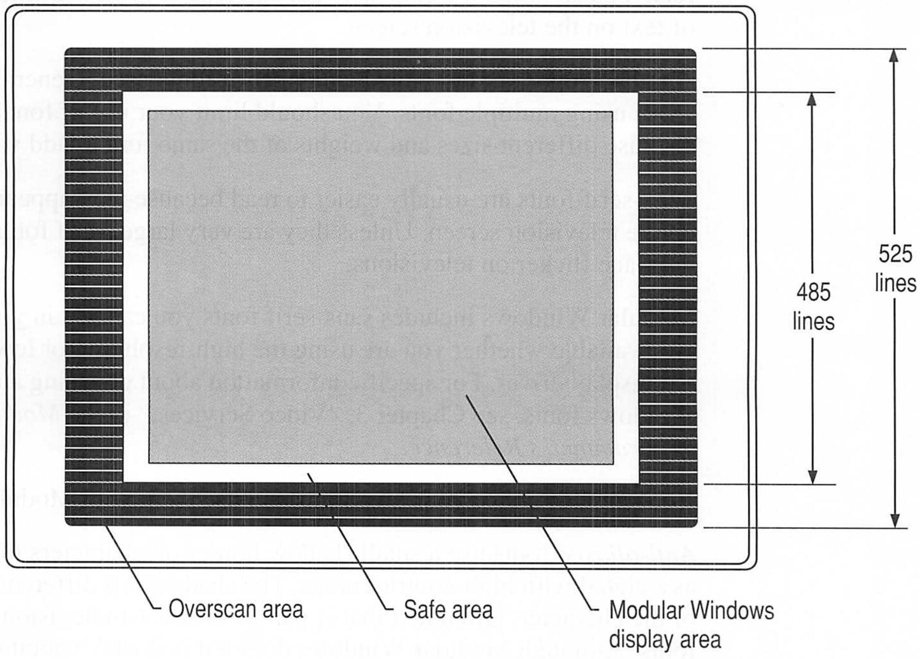
How televisions use interlacing to double the effective frame rate

Color televisions in the United States use 525 lines per frame and display 30 frames per second. In Europe, televisions use 625 lines per frame and display 25 frames per second. Ordinarily, neither of these frame rates is sufficient to eliminate screen flicker. In an interlaced picture, each picture is divided into two images—one uses odd-numbered lines and the other uses even-numbered lines. Each image is called a *field*. The electron beam traces the odd-numbered field first, then traces the even-numbered field. Therefore, 30 pictures per second is the same as 60 fields per second. At this display rate, screen flicker is reduced to an acceptable level.

Interlace flicker occurs when there is a high level of contrast between a single scan line and the two scan lines adjacent to it. When you are designing applications for televisions, you can avoid interlace flicker by using horizontal lines whose widths are an even number of pixels. In high-resolution mode, Modular Windows includes pre-defined objects that meet this requirement. If you create your own objects, be sure to specify appropriate line widths. In low-resolution mode (320-by-200), interlace flicker is not a problem because the video hardware operates in non-interlaced mode.

About TV-Based Video Hardware

Most televisions don't use the full 525-line bandwidth available in a broadcast signal. NTSC standards specify a *safe area* that is guaranteed to be visible on all televisions. The safe area has a vertical resolution of 485 lines, as shown in the following illustration:



Television safe area and display area used by Modular Windows

Modular Windows places its entire display area within the safe area so that designers need not be concerned with ensuring that critical parts of the display will be visible on all televisions. A good design tip, however, is to leave some space between controls and objects near the edges of the display. Viewers feel less crowded with this format. Placing objects too near the edge creates a cluttered look.

Choosing a Display Resolution

Modular Windows supports both high-resolution and low-resolution display modes. The high resolution mode is 640-by-400 and the low-resolution mode is 320-by-200. Both modes provide 256 colors. The high-resolution mode is best for applications that use the Modular Windows user-interface controls. Applications that provide their own user-interface controls, such as many games, might be better suited for the low-resolution mode. The low-resolution mode will also help reduce flicker.

Selecting a Font

The most important factor in selecting a font for use on a television display is the size. The font should be clear and easy to read from a viewing distance of 8 to 10 feet. You should use a font size that corresponds to a maximum of about 15 lines of text on the television screen.

Another important factor is the number of fonts used. Generally, less is better when using multiple fonts. You should limit your use of fonts to two. You can also use different sizes and weights of the same font to add variety and emphasis.

Sans-serif fonts are usually easier to read because they appear sharper and crisper on the television screen. Unless they are very large, serif fonts are prone to interlace flicker on televisions.

Modular Windows includes sans-serif fonts you can use in your application. Fonts are available whether you are using the high-resolution or low-resolution mode of the display driver. For specific information about choosing and using Modular Windows fonts, see Chapter 3, "Video Services," in the *Modular Windows Programmer's Reference.*

You can also create your own fonts and use them with Modular Windows.

Anti-aliased fonts use a small shadow border on characters to avoid the flicker associated with high-contrast areas. The shadow is a different color from the body of the character. Often text that is part of broadcast television uses anti-aliased fonts. Although Modular Windows does not provide explicit support for anti-aliased fonts, you might want to experiment with providing this capability in your application.

Using Color

Color is a powerful design tool. Used appropriately, it can make your application more interesting and more effective. Used inappropriately, however, it can make your application confusing and frustrating. This section provides some general information about using color on a television and offers some general recommendations about using color.

Recommendations for Using Color

A good design rule when using color is to keep it simple. Remember that televisions are designed to be viewed from a greater distance than a monitor. You should assume the application will be viewed at a distance of 8 to 10 feet.

You should keep in mind the following guidelines when adding color to your application:

- Be consistent.
- Use a minimum number of colors.
- Use color because users expect color.
- Choose contrasting colors carefully.
- Use bright colors to direct the user's attention.

Be Consistent

Choose colors that are familiar to and have appropriate meaning for your audience, using colors consistently throughout the application.

Be aware that color associations might be culture-dependent. For example, in some cultures, red is associated with danger, stop, hot, and fire; yellow with caution, slow, and test; green with go. If your audience is the United States, don't use green to indicate an error or a severe warning, and don't use red to indicate that everything is normal.

Similarly, warm colors generally suggest that an action should be taken, that a response is expected. Cool colors indicate background information. Grays, white, and blue suggest neutrality.

Use a Minimum Number of Colors

A good general rule is to use five colors, and no more than seven, in an application. Most viewers remember meanings and color relationships when you limit the number of colors.

Use Colors as Users Expect Them to be Used

Most viewers select red, green, and blue (cyan) to represent the front, middle, and back layers of a multi-layer object. If you need to use multiple colors, you should arrange them in the following order: yellow, green, blue, indigo, and violet.

Choose Contrasting Colors Carefully

There are several points to consider in this area. Remember that some viewers have a color-perception deficiency. To accommodate this portion of your audience, avoid contrast pairs that are difficult for these viewers to distinguish. For example, viewers with a red-green color-perception deficiency see both red and green as brown.

The human eye contains less blue-sensitive receptors. Therefore, you should use blue for large areas, such as slide and screen backgrounds. Don't use blue for text type, thin lines, or small shapes.

The retina of the eye is less sensitive to red or green at the edges of an image than at the center. As much as possible, use red or green in the visual center of the application, not at the peripheral edges. If you use red or green at the edges, try to include an additional visual cue, such as blinking or size change to make sure you get the viewer's attention.

Strong contrasts might create vibrations, the illusion of shadows, or afterimages. Red-green, blue-yellow, green-blue, and red-blue contrasts are susceptible to these problems.

When working with televisions, the best approach is to minimize adjacent use of high-intensity, high-brightness colors. For example, a bright-red flower in the midst of green forest will reproduce very well on a television, but a flag with adjacent stripes of bright red, blue, and yellow might not.

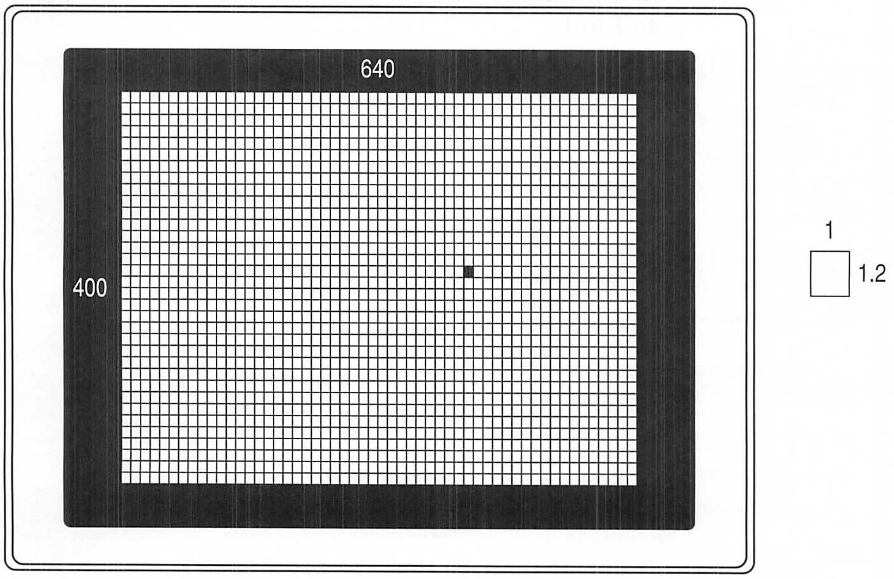
Use Bright Colors to Focus Attention

Bright colors get the viewer's attention. Bright red or blue attracts attention more quickly than yellow or yellow-orange. Brighter colors also help older viewers distinguish colors, as well as viewers of all ages who view the screen for extended periods of time.

Pixel Aspect-Ratio Considerations

Aspect ratio is a comparison of horizontal and vertical proportions. A true square has an aspect ratio of 1 because the vertical lines in a square are the same length as the horizontal lines. For example, a television screen is not a square, but a rectangle with an aspect ratio of 4 to 3. This indicates that the horizontal width of the screen is 1.33 times the vertical height of the screen.

Pixel aspect ratio is a comparison of the horizontal and vertical dimensions of a single pixel. In computer systems, the pixel aspect ratio under VGA is 1 to 1. This means that pixels are square on VGA monitors. However, as the following illustration shows, the pixel aspect ratio on televisions is 1 to 1.2, which means that pictures created on a VGA monitor will appear to be stretched when you display them on a television:



Pixel aspect ratio on a television

Aspect ratio is another area where testing on the greatest number of televisions and the widest possible range of television types will help you ensure image clarity. The best approach is to assume your images will vary on different televisions. What appears as a perfect square or circle on your computer monitor might not be perfect on all televisions you test.

Generally, aspect ratio is not a critical design factor. Although you might observe slight irregularities in image shapes, it is not usually sufficient to be distracting to viewers. Again, testing on a variety of televisions helps you produce the best possible shapes on the greatest number of televisions.

Designing for Other Television Systems

If you design applications for European TV-based players, you can expect additional differences, such as the following:

- Video will be more prone to flicker. This is because NTSC (the United States standard) refreshes at about 30 frames per second, while PAL (the European standard) only uses about 25 frames per second.
- Color accuracy is often better. For example, blues are much more consistent in PAL than they are in NTSC.
- PAL uses more lines per frame. This means images designed for NTSC will have more black space at the top and bottom of the screen.
- The increased number of lines per frame creates a different aspect ratio. Pixels are shorter in PAL than in NTSC, which means the pixel aspect ratio is closer to 1 to 1.

Index

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®

* 3 7 7 4 3 *