



Palm OS® User Interface Guidelines

CONTRIBUTORS

Written by Jean Ostrem

Engineering contributions by Bob Ebert, Roger Flores, JB Parrett, Jesse Donaldson, Ron Fernandez, David Fedor, Greg Wilson, Clif Liu, Brian Maas, Ezekiel Sanborn De Asis, Brent Gossett, and Maurice Sharp
Special thanks to Catherine E. White, Neil Rhodes, Julie McKeehan, Alexander Hinds, and Lee Fyock

Copyright © 2003, PalmSource, Inc. and its affiliates. All rights reserved. This documentation may be printed and copied solely for use in developing products for Palm OS® software. In addition, two (2) copies of this documentation may be made for archival and backup purposes. Except for the foregoing, no part of this documentation may be reproduced or transmitted in any form or by any means or used to make any derivative work (such as translation, transformation or adaptation) without express written consent from PalmSource, Inc.

PalmSource, Inc. reserves the right to revise this documentation and to make changes in content from time to time without obligation on the part of PalmSource, Inc. to provide notification of such revision or changes.

PALMSOURCE, INC. AND ITS SUPPLIERS MAKE NO REPRESENTATIONS OR WARRANTIES THAT THE DOCUMENTATION IS FREE OF ERRORS OR THAT THE DOCUMENTATION IS SUITABLE FOR YOUR USE. THE DOCUMENTATION IS PROVIDED ON AN "AS IS" BASIS. PALMSOURCE, INC. AND ITS SUPPLIERS MAKE NO WARRANTIES, TERMS OR CONDITIONS, EXPRESS OR IMPLIED, EITHER IN FACT OR BY OPERATION OF LAW, STATUTORY OR OTHERWISE, INCLUDING WARRANTIES, TERMS, OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND SATISFACTORY QUALITY. TO THE FULL EXTENT ALLOWED BY LAW, PALMSOURCE, INC. ALSO EXCLUDES FOR ITSELF AND ITS SUPPLIERS ANY LIABILITY, WHETHER BASED IN CONTRACT OR TORT (INCLUDING NEGLIGENCE), FOR DIRECT, INCIDENTAL, CONSEQUENTIAL, INDIRECT, SPECIAL, OR PUNITIVE DAMAGES OF ANY KIND, OR FOR LOSS OF REVENUE OR PROFITS, LOSS OF BUSINESS, LOSS OF INFORMATION OR DATA, OR OTHER FINANCIAL LOSS ARISING OUT OF OR IN CONNECTION WITH THIS DOCUMENTATION, EVEN IF PALMSOURCE, INC. OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Palm OS, Palm Computing, HandFAX, HandSTAMP, HandWEB, Graffiti, HotSync, iMessenger, MultiMail, Palm.Net, PalmPak, PalmConnect, PalmGlove, PalmModem, PalmPoint, PalmPrint, and PalmSource are registered trademarks of PalmSource, Inc. or its affiliates. Palm, the Palm logo, MyPalm, PalmGear, PalmPix, PalmPower, AnyDay, EventClub, HandMAIL, the HotSync logo, Palm Powered, the Palm trade dress, Smartcode, Simply Palm, ThinAir, WeSync, and Wireless Refresh are trademarks of PalmSource, Inc. or its affiliates. All other product and brand names may be trademarks or registered trademarks of their respective owners.

IF THIS DOCUMENTATION IS PROVIDED ON A COMPACT DISC, THE OTHER SOFTWARE AND DOCUMENTATION ON THE COMPACT DISC ARE SUBJECT TO THE LICENSE AGREEMENT ACCOMPANYING THE COMPACT DISC.

Palm OS User Interface Guidelines
Document Number 3101-001-HW
February 24, 2003
For the latest version of this document, visit
<http://www.palmos.com/dev/support/docs/>.

PalmSource, Inc.
1240 Crossman Avenue
Sunnyvale, CA 94089
USA
www.palmsource.com

Table of Contents

About This Document	ix
Why Follow Guidelines?	ix
How This Book Is Organized	xi
What This Book Does Not Cover	xi
Additional Resources	xii
 1 Palm OS Application Design	 1
Palm OS Design Principles	1
Pocket Size	2
Fast and Simple	3
Low Cost, Long Battery Life, and High Value	8
Seamless Connection with Desktops	9
The Design Process	10
The Usual Approach	11
The Recommended Approach	12
Decide on Design Goals	13
Know Your Users	14
Develop User Scenarios	15
Propose an Implementation	17
Develop the Initial Design Concept	19
Complete the Design	21
 2 Fitting In	 23
User Interaction with Palm Powered Handhelds	23
Graffiti or Graffiti 2 Writing	24
Onscreen Keyboard.	24
HotSync Operation	25
Hard Keys.	25
Icons in the Input Area	25
External Keyboard	26
Application Controls	26
Integrating with the Application Launcher	26
Application Icons	27
Version String	28

Default Application Category	29
General Application Layout Guidelines	30
Main Application Forms	31
Controls	32
Control Placement	34
Labels	34
Fonts	35
Graphical Controls	36
Custom Controls	37
Application Categories	38
General Application Behavior Guidelines	39
Launching the Application	39
Exiting the Application	40
Supporting Global Find	41
Respecting User Preferences	42
Allowing System Messages	43
Becoming Compatible Worldwide	43

3 Forms 45

Choosing between Types of Forms	45
Use Modal Forms Sparingly	47
Avoid Modal Forms for Lengthy Data Entry	47
Modeless Forms	48
System Supplied Behavior.	49
Look and Feel	50
Breaking the Rules	52
Modal Forms.	55
System Supplied Behavior.	56
Look and Feel	56
Breaking the Rules	59
Alert Dialogs.	60
Types of Alerts.	61
Look and Feel	62
Breaking the Rules	65
Progress Dialogs	65

About Dialogs	67
Tips Dialogs	69
4 Executing Commands	71
Choosing between Buttons and Menus	71
Limit the Total Number of Commands	72
Use Buttons for Important Tasks	73
Use Menus for Destructive Commands	74
Don't Duplicate Commands	74
Remember the Goal: Minimize Taps	75
Use Buttons for Commands Executed by New Users	77
Don't Provide Save or Exit Commands	77
Command Buttons	78
System Supplied Behavior.	79
Look and Feel	79
Breaking the Rules	83
Menus	86
System Supplied Behavior.	87
Look and Feel	88
Breaking the Rules	94
5 Presenting Options	99
Choosing Which Element to Use	99
Choosing Several of Many Options	100
Choosing One from Many Options	100
Implementing a Combo Box	104
Check Boxes	106
Pop-Up Lists	108
System Supplied Behavior.	110
Look and Feel	111
Breaking the Rules	112
Push Buttons.	116
System Supplied Behavior.	117
Breaking the Rules	118
Selector Triggers	119
System Supplied Behavior.	120

Look and Feel	120
Breaking the Rules	120
Sliders	123
System Supplied Behavior.	123
Look and Feel	124
6 Displaying Data	127
Choosing Which Element to Use	127
Fields	129
System Supplied Behavior.	130
Look and Feel	131
Breaking the Rules	136
Lists	137
System Supplied Behavior.	138
Look and Feel	139
Breaking the Rules	140
Tables	142
System Supplied Behavior.	143
Look and Feel	144
7 Scrolling	147
Choosing between Scroll Bars and Scroll Buttons	147
Scroll Bars	150
System Supplied Behavior.	151
Look and Feel	152
Breaking the Rules	153
Scroll Buttons	153
System Supplied Behavior.	154
Look and Feel	155
Breaking the Rules	156
8 Color and Graphics	159
Palm OS Color Support	159
Colors of User Interface Elements.	160
Graphics	161

A Ten Things to Remember	163
Index	167

About This Document

This book describes how to design applications for Palm Powered™ handhelds so that they conform to PalmSource, Inc's user interface guidelines. Read it if you are an application designer or a developer and you are considering creating applications that run on Palm OS®.

Why Follow Guidelines?

Users have come to know and love their Palm Powered handhelds. They expect the applications that run on them to look a certain way and behave a certain way.

Follow the guidelines presented in this book, and your application will look and feel like other Palm OS applications. Users will learn it more quickly, and you'll be well on your way to creating an application that they love and recommend to their friends and colleagues. As a bonus, you'll be able to focus on creating and improving the application itself because documentation requirements and calls to technical support are reduced if the application works the way users expect it to work.

Ignore the guidelines, and you invite user frustration and confusion. Your users may adapt to your application eventually, but you run the risk of alienating a customer base. Sales can suffer.

At the same time, guidelines are not hard and fast rules. A guideline can be broken, but you should do so only after you know why it is there, you have considered all factors involved, and you believe that breaking the guideline is more beneficial to your users than following it would be.

Designing a proper user interface may feel at first like you are "wasting" a lot of time agonizing over the user interface rather than focusing on the nuts and bolts of your application. While most developers acknowledge the importance of user interface design,

About This Document

Why Follow Guidelines?

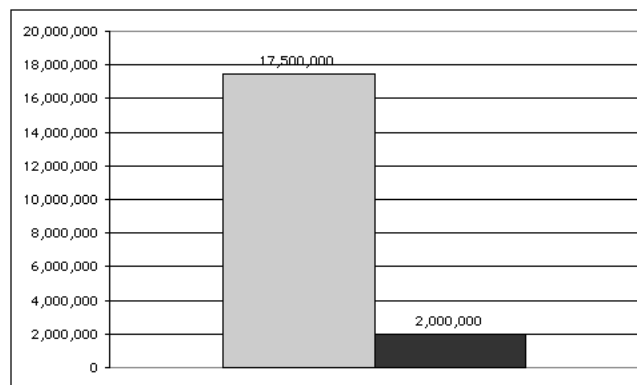
it's easy to let design take a back seat until the application is working properly. If you do take the time to get the interface right, however, you'll achieve greater success in the long run.

Consider as a case study the success the Palm Powered handheld has had compared to its major competitor, the Pocket PC handheld. The Palm Powered handheld was not the first handheld on the market, but it was the first one to be successful because it focused on giving users what they wanted. Palm OS was designed from the ground up with users in mind. Designers focused on what users would want to do with a handheld and provided just that functionality. More features are added only when doing so does not diminish the overall user experience.

In contrast, the first versions of the Windows Pocket PC operating system were designed by stuffing as many features of a desktop Windows system as possible into a pocket-sized form factor. The resulting user experience has been so negative that it has forced the designers back to the drawing board more than once.

Windows Pocket PC handhelds have been on the market for more than three years. Only recently have those devices begun to gain acceptance. In the meantime, 17.5 million Palm Powered handhelds have been sold compared to 2 million Pocket PC handhelds (see [Figure 1](#)).

Figure 1 Sales of Palm Powered vs. Pocket PC handhelds



Source: Dataquest estimate

You can see that if you design your interface right the first time, you get a big head start.

How This Book Is Organized

This book is organized as follows:

- [Chapter 1](#) describes the basic principles behind the Palm OS user interface guidelines, and it describes the design process used at PalmSource, Inc. Read it to get a good grounding in Palm OS interface design that prepares you for the chapters that follow.
- [Chapter 2](#) tells you how to design an application so that it fits in well with other Palm OS applications.
- [Chapter 3](#) through [Chapter 8](#) help you with user interface element selection.

Each chapter describes a group of elements that have a common purpose: displaying data, entering commands, and so on. The first section in each chapter describes how to determine which of the elements in the chapter is best for your situation.

The remaining sections give appearance and behavior guidelines specific to each element. Within those sections, you'll find specific display recommendations, descriptions of the behavior Palm OS supplies for the element, and descriptions of the behavior your code must provide.

Because user interface design is not an exact science, many sections end with examples of when to break the rules.

- [Appendix A](#) lists the ten most important things to remember about Palm OS application design.

What This Book Does Not Cover

This book focuses solely on design. It does not provide instructions on how to use a development environment to create user interface resources, and it does not discuss which API calls to use to produce a desired behavior from a user interface element.

Augment the information in this book with the following:

- The *Palm OS Programmer's Companion*, which provides conceptual and task-based information for developers

About This Document

Additional Resources

- The *Palm OS Programmer's API Reference*, which contains reference material for Palm OS developers
- Documentation for the development environment of your choice

Additional Resources

- Documentation
PalmSource publishes its latest versions of this and other documents for Palm OS developers at
<http://www.palmos.com/dev/support/docs/>
- Training
PalmSource and its partners host training classes for Palm OS developers. For topics and schedules, check
<http://www.palmos.com/dev/training>
- Knowledge Base
The Knowledge Base is a fast, web-based database of technical information. Search for frequently asked questions (FAQs), sample code, white papers, and the development documentation at
<http://www.palmos.com/dev/support/kb/>

Palm OS Application Design

This chapter provides some principles that you should follow when designing your Palm OS® application. It begins by describing how a Palm Powered™ handheld differs from other types of computers and how these differences affect your user interface design. Then it outlines a design process that helps you to create a successful user interface.

This chapter supplies only very general design principles. Read it to give yourself a grounding in Palm OS design. Specific guidelines are given in the remaining chapters of this book.

Palm OS Design Principles

A Palm OS application should provide information that users will want to access when they are away from their desks. Its user interface should allow the user to get to the most relevant information as quickly as possible.

It's common for beginning Palm OS programmers and designers to believe that a Palm Powered handheld is simply a very small laptop. These designers try to port their existing desktop application's look and feel directly over to the handheld, resulting in a very complex and often unusable interface.

Instead of duplicating a desktop application's look and feel, it is better to provide only the features that your users will want on the handheld. Handhelds are designed and used for different purposes than laptop computers. Good UI design for any platform begins by considering what the user needs to accomplish, and the Palm OS platform is no different in this regard. If you simply try to duplicate your desktop user interface on the handheld, your users may become frustrated and give up on your application.

This section describes some of the handheld's key characteristics and how they should influence the design of your user interface.

- [Pocket Size](#)
- [Fast and Simple](#)
- [Low Cost, Long Battery Life, and High Value](#)
- [Seamless Connection with Desktops](#)

Pocket Size

Most Palm Powered handhelds have a small screen and no keyboard—they are designed to fit in a shirt pocket. The small device size has the following effects on application design.

Applications Must Limit Data Entry

Do not require users to enter a lot of data on the device itself. Because there is no keyboard, users mainly enter characters using Graffiti® writing, Graffiti 2 writing, or a keyboard dialog. While these are useful ways of entering data, they are not as convenient as using the full-sized desktop computer with its keyboard and mouse. External keyboards exist and there are some Palm Powered handhelds with small built-in keyboards, but neither of these should be considered a requirement.

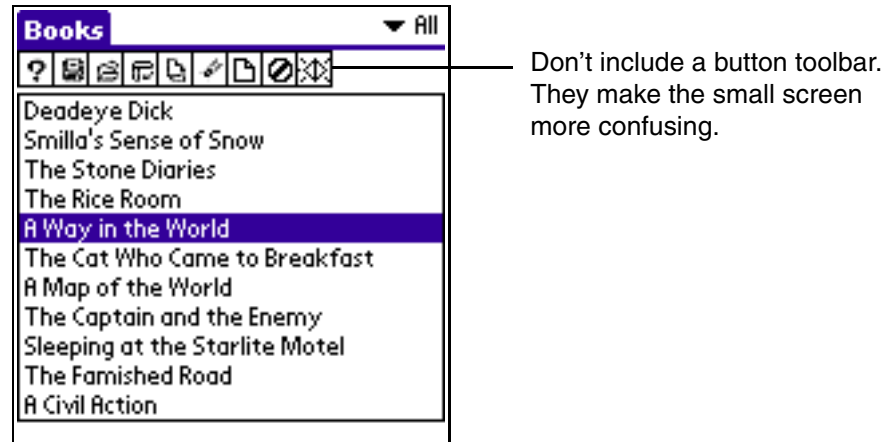
Menus Are Hidden

To allow more room for data, the menu bar is not displayed by default. Instead, the menu bar is displayed by tapping an icon on the handheld. Because menus are hidden, many users do not realize that they are available. You'll need to limit your use of menus to power commands not necessary for the basic use of the application.

No Button Toolbars

It's common to see a long strip of buttons at the top of a desktop application. These button toolbars are not ideal for the Palm OS platform because the buttons are too small to tell apart (see [Figure 1.1](#)). Instead, you must provide buttons only for the essential commands.

Figure 1.1 Applications should not have toolbars



Less Is More

To save space, sacrifice features that don't belong on a handheld. On a desktop system, users use 20% of an application's features 80% of the time. Your application should provide only that top 20% of features. Save any other functionality for the desktop application.

On a desktop system, it's easy to just add another button to the button toolbar or add another menu. For a Palm OS application, you'll have to resist that temptation. Otherwise, your screen becomes too cluttered, and your application becomes too complicated to use.

Fast and Simple

The Palm Powered handheld is fast to use and easy to learn. Because the handheld's main purpose is to make it easy to organize and manage your life, it requires a minimal learning curve. Users must be able to pick up a Palm Powered handheld and, with no training or instruction, navigate between applications (without getting stuck) and execute basic commands within five minutes.

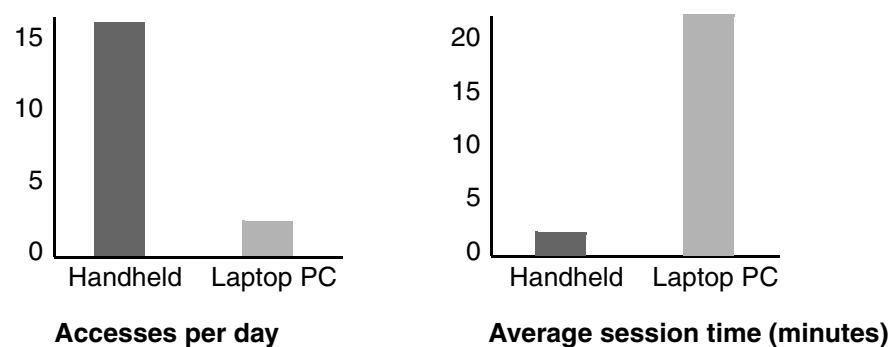
This contrasts with desktop systems. Although desktop systems are also designed to be easy to use, there is a desktop paradigm that new users must spend a day or two learning.

Because the Palm Powered handheld must be fast and easy to use, it has the following effects on application design.

Perceived Speed Is Important

On a desktop, users don't mind waiting a few seconds while an application loads because they plan to use the application for an extended amount of time. On a handheld, users want to quickly look something up and then go on about their lives, and they do this several times a day (see [Figure 1.2](#)).

Figure 1.2 Opposite usage patterns



Source: Palm, Inc. user surveys.

Optimize your application to these short bursts of user activity. Remember that requiring a user to spend an extra 30 seconds to find necessary information is excusable when they are sitting down for 3 hours at a desktop computer but cumbersome on a handheld when that is the only thing they are going to do before they turn it off.

As a rule of thumb, the user should be able to keep up with someone on the telephone when setting up appointments, looking up phone numbers, and so on. Priorities include the ability to:

- Execute key commands quickly
- Navigate to key screens quickly
- Find key data quickly (for example, phone numbers)

Minimize Required Steps

Minimize the number of steps a user must perform to see vital information. Display the most essential information on the first screen of the application. For example, Date Book always displays today's calendar or agenda when it starts up. It does so because

most users access the Date Book to see their current schedule or agenda 80% of the time.

Reduce clutter so that users will find the information they need quickly. Strive for a balance between providing enough information and overcrowding the screen.

Place command buttons on the first screen that perform the tasks the user will want to perform most often. Accomplishing common tasks should be fast and easy.

Choose the number of buttons on the screen carefully:

- The fewer buttons on the screen, the less time it takes to learn how to use the product. If there are too many buttons, users are forced to hunt and peck to find what they need until they eventually learn the placement of each button.
- On the other hand, keeping a few frequently used buttons on screen helps reduce the time spent learning basic functionality.

[Chapter 4, “Executing Commands,”](#) on page 71 covers the topic of choosing the number of command buttons in more detail.

Minimize Taps

Most information about that data should be accessible in a minimal number of taps of the stylus — one or two.

Desktop user interfaces are typically designed to display commands as if they were used equally. In reality, some commands are used very frequently while most are used only rarely. Similarly, some settings are more likely to be used than others. On Palm Powered handhelds, more frequently used commands and settings should be easier to find and faster to execute.

- Frequently executed software commands should be accessible by one tap.
- Infrequently used or dangerous commands may require more user action.

[Table 1.1](#) shows how the frequency of an action maps to its accessibility in the Date Book application.

Table 1.1 Frequency of actions

Frequency	Example	Accessibility
Several times per day	Checking today's schedule or to-do items	One tap
Several times per week	Scheduling a one hour meeting starting at the top of the hour	One tap, write in place
A few times a month	Setting a weekly meeting (repeating event)	Several taps, second dialog box

This goal of minimizing taps can be taken too far. It must be balanced with other guidelines. For example, using a command button generally minimizes the taps to perform a command, but if you have too many buttons, you overcrowd the screen and introduce confusion into the interface.

Some designers break the behavior guidelines for a particular element for the sole purpose of minimizing taps. User interface elements should behave the way users expect them to behave unless there is good reason for them not to. Minimizing taps is not always a good enough reason to break guidelines.

Be Consistent

Consistency reduces the time needed to learn an application by limiting the number of things that people need to keep in their heads at once. The user should not have to memorize an entire set of rules to use the handheld easily. For example, the up arrow key should not do different things on different screens.

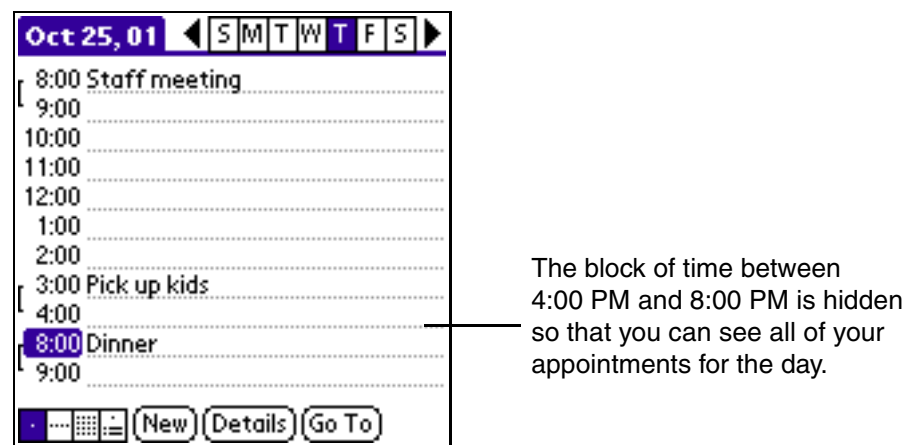
If possible, make your application consistent with the handheld's built-in applications; users know how to interact with them and will quickly learn your application if the user interface is similar to applications they already know.

Optimize Frequent Tasks

Most users launch Date Book to see today's schedule most of the time. Date Book helps with this task. Not only does it open to

today's schedule, it also always tries to show you all of today's appointments, even if it means some of the hour blocks disappear. For example, if you have appointments at 8:00 AM, 3:00 PM, and 8:00 PM, the Date Book shows you all three appointments without forcing you to scroll to see that last appointment. To do so, it hides the unused hour blocks between 4:00 PM and 8:00 PM. (See [Figure 1.3](#).)

Figure 1.3 Datebook calendar optimization



Similarly, the Address Book is optimized to display a large number of contacts because most users will have long lists of contacts. It allows the user to divide contacts into different categories and display only the contacts in the selected category. It also has a look-up field on the main screen that allows the user to easily navigate to the contact he or she wants by writing a character.

Consider providing power user features for people who use your application often. These advanced features should be easily accessible but not get in the way. For example:

- If you start to write a character or number while in Date Book, Date Book assumes you are trying to schedule a new appointment and helps you do so. However, the Date Book still shows a New command button so that novice users will know how to create an appointment.
- The Address Book beams your business card if you hold down the hard key. This is also a power user feature that newcomers can discover eventually.

Low Cost, Long Battery Life, and High Value

Previous handheld products failed in part because they cost too much and devoured batteries. For this reason, two key design goals of Palm Powered handhelds have been to have the handheld fall into a specific price range and have readily available batteries that last a long time. Many Palm Powered handhelds run on readily available, cheap AAA batteries, and a battery life of one month is not uncommon.

Keeping costs down and battery life high are key factors in determining which hardware components are selected. Those component choices, in turn, affect the decisions you make when designing your applications.

- Some Palm Powered handhelds have lower quality screens than competing handhelds because the low quality screen uses less power, requiring the user to replace the batteries less frequently.
- The processor was also chosen for its low power usage rather than for its speed. Most existing Palm Powered handhelds have processors with clock speeds ranging from 16 MHz to 33 MHz, although devices with faster ARM-based processors are beginning to become available.

The designers of the first Palm-Powered handheld felt that “perceived speed” was more important than actual processing speed, therefore a relatively low speed processor could be used to maximize battery life. Perceived speed is the speed of the overall user experience. If a user can start up the Palm Powered handheld and see a list of today’s appointments faster than he or she can on a competitor’s handheld, then processing speed does not matter.

- Graffiti power writing software was another design decision affected by the battery selection. During the design of the first Palm handhelds, users were clamoring for natural handwriting recognition. However, natural handwriting recognition would require a more powerful processor and more memory, which together required bigger batteries. Adding all these things to a handheld would have weighed it down and made it cost too much for the market. Instead, the Palm designers bet that users would settle for good-enough handwriting recognition if the result was long battery life.

As the handheld evolves, some of these decisions made by the designers of the initial Palm handheld are no longer as relevant as they once were. For example, some handhelds come with lithium ion rechargeable batteries built in rather than using the AAA batteries. Users do not worry about battery life as much if they can simply place the handheld in the recharger at the end of the day. Because maximizing battery life is not as important on these handhelds, they can use high resolution screens and faster processors.

Keep User Costs Down

Your application should help keep user costs down by maximizing battery life. To maximize battery life, consider your actions carefully before performing tasks that consume a lot of power. Serial communications, IR communications, playing sounds, disabling the auto-off feature, and extended animation are among the tasks that consume a lot of power. When your application must perform these tasks, it should do so in a way that consumes as little power as possible. If you open the serial port, for example, you should close it immediately after you are done using it.

Don't make the mistake of deciding that battery life is not ever important to users of handhelds with rechargeable batteries. The lithium ion batteries often do not last as long as two AAA batteries. Many users appreciate the convenience of being able to go for several days without charging. Others often travel on business and may be away from their chargers for days at a time. You should respect the experience of those users.

You affect battery life more with your programming choices than you do with your user interface design choices. See the *Palm OS Programmer's Companion* for programming tips on reducing the amount of power consumption.

Seamless Connection with Desktops

Desktop connectivity is an integral component of the Palm OS platform. The handheld comes with a cradle that connects to a desktop computer and with software for the desktop that provides "one-button" backup and synchronization of all data on the device with the user's desktop.

Many Palm OS applications have a corresponding application on the desktop. To share data between the handheld's application and the desktop's application, you must write a **conduit**. A conduit is a plug-in to the HotSync technology that runs when you press the HotSync button. A conduit synchronizes data between the application on the desktop and the application on the handheld. The conduit that you write must handle several different environments:

- One handheld may synchronize with several desktop computers.
- Several handhelds may synchronize with different users on a single desktop computer.
- Data can be entered either on a desktop or on the handheld.

The conduit must run without user intervention. The user is not required to watch the desktop screen or be anywhere near the desktop screen while the device synchronizes. The conduit must determine which data records are the most relevant, considering all factors involved, and then decide on its own which records are copied between the desktop computer and the handheld.

The Design Process

In the previous section, you learned the key characteristics of the Palm Powered handheld and how these characteristics should affect your application's user interface. The handheld's characteristics were determined by a design process that focused on the user: the user wanted a fast, easy-to-use, inexpensive device with a long battery life. All design decisions were made to give users just that.

This section introduces a user-centered design process used by successful designers both inside of and outside of PalmSource, Inc. You can ensure that your application is successful if you follow this process. A design process that focuses on the user is nothing particularly new or unique; successful interface designers for any platform follow a similar process.

Even though there's nothing earth-shattering here, all too often the user-centered design process is not followed. The results are poor user interfaces. For this reason, this section begins by describing the

wrong approach to interface design, an approach all too commonly followed. Then it describes the recommended approach.

The Usual Approach

As soon as you know what application you're working on, it is very tempting to jump right in and start coding features, thinking you'll fix any design problems later in the development cycle. However, this is almost always the wrong way to design an application, particularly a Palm OS application.

To an extent, you can get away with a poor user interface when you have a large, colorful screen on a computer with almost unlimited processing power. Design flaws on a Palm Powered handheld are more visible.

Because application developers are comfortable with technology, their first instinct often is to focus on technological solutions. The actual user problem that they are trying to solve gets lost in the rush to add cool features.

Suppose you have been assigned to design an application for book collectors. These users want to keep track of the books they own, to whom they have lent the books out, and when they are due back. A typical technology-focused approach to this problem goes something like this:

"This application needs at least two screens. The main screen lists the books, and the second screen is a detail screen showing if the book is lent and to whom. We need a New button to create a book, so we might as well add an Edit button and a Delete button just in case. We can sort the books by title and author. For data entry, we'll allow users to scan in the ISBN number for newly acquired books in case they have a Symbol bar code scanner. That's much faster than Graffiti writing. We can allow the users to beam books back and forth to each other. We'll also link the application to Web Clipping applications in case they have a Palm VII series handheld and want to buy a book online with it..." and so on.

Shortly after this initial thought process, the initial sketches are drawn (see [Figure 1.4](#)).

Figure 1.4 Initial design sketches

The figure consists of two side-by-side sketches of Palm OS application screens. The left screen is titled 'Books' and features a list of books with two columns: the book title and the author's name. The list includes 'Fellowship of the Ring' (Tolkein), 'Foundation' (Asimov), 'Foundation and Empire' (Asimov), 'Hitchiker's Guide to th...' (Adams), 'Hobbit' (Tolkein), 'I, Robot' (Asimov), 'Restaurant at the End...' (Adams), 'Return of the King' (Tolkein), 'Second Foundation' (Asimov), and 'Two Towers' (Tolkein). A dropdown menu 'Sort by Title' is in the top right. At the bottom is a bar with five buttons: 'New', 'Edit', 'Delete', 'Scan', and 'Beam'. The right screen is titled 'Book Detail' and contains several labeled fields: 'Title: The Hobbit', 'Author: JRR Tolkein', 'Loaned To: Gandalf', and 'Due Back: 12/19/01'. A 'Notes' field contains the text 'We have room for this, so we might as well have miscellaneous notes field.' A 'Done' button is at the bottom.

Books		▼ Sort by Title
Fellowship of the Ring	Tolkein	
Foundation	Asimov	
Foundation and Empire	Asimov	
Hitchiker's Guide to th...	Adams	
Hobbit	Tolkein	
I, Robot	Asimov	
Restaurant at the End...	Adams	
Return of the King	Tolkein	
Second Foundation	Asimov	
Two Towers	Tolkein	

New Edit Delete Scan Beam

Book Detail	
Title:	The Hobbit
Author:	JRR Tolkein
Loaned To:	Gandalf
Due Back:	12/19/01
Notes:	We have room for this, so we might as well have miscellaneous notes field.

Done

Is this good design? Does the user need all of the command buttons on the main form?

As you can see, this method of application design more closely resembles a brainstorming session than a reliable process.

Such a process may or may not lead you to an acceptable application. Some problems with this initial design:

- Scanning the ISBN number might be nice, but that only gives you the ISBN number. The user must then translate that into a book title by looking up the number on the web. What if the user does not have a bar code scanner on the device or owns a lot of old books that do not have an ISBN number?
- Beaming the book title has little utility and requires a lot of processing power for the little utility it provides. What would it mean to beam a book title to someone else?
- Forcing the user to navigate to a detail screen to show who has borrowed the book might become tedious if what the user really wants is to scan through his or her list of books and know at a glance which books have been loaned out.

The Recommended Approach

Now that we've learned a little bit about the usual approach to interface design, let's discuss a better approach. The better approach is a multi-step process that is more likely to give you the desired

results: lots of happy users willing to part with their money. The steps to the right approach are:

- [Decide on Design Goals](#)
- [Know Your Users](#)
- [Develop User Scenarios](#)
- [Propose an Implementation](#)
- [Develop the Initial Design Concept](#)
- [Complete the Design](#)

You might not follow all of this design process exactly as it is described here, particularly if you work in a small company or by yourself. Creating software is a complicated task affected by a lot of variables. Every project is unique. Still, it's useful to know about this recommended user interface design process and to try to follow it as closely as possible.

Decide on Design Goals

The Palm designers made decisions to achieve these design goals: a device that fits in a shirt pocket, is fast and simple to use, requires no learning curve, is low cost, and has a worry-free battery life. Just as the Palm designers started with their design goals, you should start with a design goal for your application. A generic set of design goals for all Palm OS applications is the following:

- Easy to learn and use
- Convenient
- Provides access to what most people want and need most of the time
- Helps users quickly achieve their goals

Your application may have additional goals, but to be successful on the Palm OS platform, it should achieve the design goals listed above.

In a small company, you'll no doubt come up with the application idea and its design goals on your own. In a large company, the design goals should come from the Marketing department. Consider the Books application example from the previous section. The statement of what the application should do may have come

from the Marketing department of a large company or it may have been an independent idea. “Create an application that allows book collectors to keep track of their books, to whom they have loaned them out, and when.” To this, the traditional Palm OS design goals simply adds adverbs: “Create an application that allows book lovers to *quickly and easily* keep track of their books and to whom they have loaned them.”

Know Your Users

Users come in all shapes and sizes. If you are like most computer professionals, then you are *not* the typical user of a Palm Powered handheld or of your future product.

Handhelds attract many people outside the computer industry due to their ease of use. Because you work in the computer industry, you are comfortable with the differences between “hardware,” “software,” and “applications.” Typical users outside of the industry see the handheld as one integrated device. They have no notion of what indicates a hardware problem, a software problem, or an application bug. They only know that they have a task to perform, and they want their Palm Powered handhelds to help them with that task.

Examine the user base your application is likely to attract. If possible, set up a focus group to learn more about your target users and why they want to use your application. A wine database application is likely to attract a wide non-technical audience. A utility to examine internal memory is going to have a mostly technical audience. A computer game might attract both the technical and non-technical members of the Palm economy.

For broad audiences, you’ll want to follow the built-in applications as much as possible. Users already know how to use those, so they will implicitly know how to use your application as well. Don’t “improve” the features, such as supporting the ability to search on suffixes as well as prefixes in the global Find facility.

If you have a narrow, highly technical audience, you can probably get away with a lot more in your user interface design; however, it never hurts to have an excellent interface. Your users will appreciate your application more if you do.

Like the design goals from the previous section, a description of the typical user is another piece of information that often comes from a Marketing department. A small company may not have the luxury of a large Marketing department with expertise in market research; however, you can still identify target users among your friends and family and talk to them about their needs.

For our Books application, we have discovered that people who are likely to buy our application have the following characteristics:

- They love to read.
- They typically own hundreds of books.
- They spend much free time browsing book stores.
- They often can't remember which books they own, which they don't, and which they own but still haven't read.
- Many freely loan books to friends, which becomes a problem when trying to determine which books they own.
- Many are not very comfortable with technology.

Because a large part of our target audience is uncomfortable with technology, we'll need to make our application as easy to use as possible, and we'll need to focus on making it work like the built-in Palm OS applications with which our user base is already familiar.

Develop User Scenarios

After gathering a picture of your typical user, begin your design by considering that user. What problem is the user trying to solve by using your application? Under what circumstances will they be using the application? Will they be at the office, the airport, at home, or in the car?

Develop some user scenarios. **User scenarios** are statements of what the user will do with the application and when. "The user needs to access email five times a day while riding on the subway" is a good scenario statement. It describes what the user is doing, where he or she is doing it, and how often. The best way to develop user scenarios is, of course, to talk to potential users, whether in focus groups or through other means.

To determine user scenarios, you must know these things:

- The likelihood that the user will perform a task
- The frequency with which the user will perform the task

For example, starting up the Date Book to access today's schedule is both likely and frequent—it is often done several times throughout the day. Tasks that are likely and frequent should be the easiest to perform.

Scheduling an appointment is another task that is both likely and frequent; however, scheduling a appointment that begins at other than a half hour or hour (for example, at 3:05) is unlikely and infrequent. We might allow people to schedule such appointments, but we can require more taps to do so because we know it is infrequently done.

A task may be likely but infrequent. For example, users are likely to run the Welcome application because it is always the first application run when they start up a new Palm Powered handheld. However, it usually is only run that one time, so its use is infrequent.

Develop as many scenarios as possible, including those you imagine to be unlikely and infrequent. This is a tricky stage in the design process. It's easy to come up with only scenarios that support preconceived notions of the application development team. If you focus only on those scenarios, you'll end up with an application that meets the needs of your team but not necessarily the needs of your users. An exhaustive user scenario effort keeps you focused on the needs of the users.

Remember our discussion of the wrong approach to designing the Books application? We came up with a design that involved bar code scanning, beaming, and buying books using a Palm VII wireless Internet connection. These were not bad ideas, but we've since learned that our user base is likely to include many people uncomfortable with technology. Such people are typically attracted to the lower-end handhelds such as m100s. A few may even own Palm Professionals. Our users are not likely to have Palm VII handhelds, and they most certainly don't have Symbol bar code scanners. So we'll have to abandon our ideas of scanning ISBN numbers and buying books from the Internet. Beaming may also be discarded if we discover that there are, in fact, a large number of Palm Professional users out there. Any reliance on color in our

application is most certainly not a good idea. Color handhelds tend to be more expensive. We can't guarantee that people have them.

Since the technology-focused design has led us down the wrong path, let's consider doing some user scenarios for the application:

- Users want to see a list of all of the books they own, which may number in the hundreds. (likely, frequent)
- Users are at a bookstore and want to see if they own a particular book. (likely, frequent)
- Users are at a bookstore and want to add a newly purchased book to the list. (likely, frequent)
- Users are away from their desktop computers and want to buy a book on the Internet (unlikely, infrequent)
- Users want to lend a book to a friend (likely, infrequent).
- Users are at home, looking for a particular book, and want to see if it is loaned out. (likely, frequent)
- Users want to see all books they have loaned out. (likely, infrequent)
- Users want to see all books they own by a particular author or about a particular subject. (likely, infrequent)
- Users want to see all books they have loaned to a given person and when they are due back. (unlikely, infrequent)
- Users have a bar code scanner and want to scan in the ISBN number of a new book. (unlikely, infrequent)

Notice that we have not yet focused on *how* to solve the users problems in these first three design steps (decide on design goals, know the user, and develop user scenarios). Instead, we have focused on *understanding* our users so that we can solve their problems.

Propose an Implementation

Now that we have an idea of who the user is and how the user will use our application, the next step is to describe the proposed implementation of the application. This description should focus on the *feel* of the application rather than its look. In this way, the proposed implementation flows naturally from the user scenarios.

The idea behind this step is twofold. First, once everybody involved in designing and developing the application agree upon the proposed implementation, engineers can begin to architect the features at the same time that human interface designers are designing the look. Second, by focusing on the feel of the application, you postpone the traditional arguments about the application look, which may unnecessarily delay software development at this point.

If you are working alone, of course, these two reasons for proposing an implementation before developing the look are not relevant. You may find it helpful to write the software without focusing much on the look until a little later, or you may not. Either way, you still should develop an implementation that flows from the user scenarios and the description of the typical users you created in the previous steps.

The proposed implementation for our Books application is:

- The initial screen will look very similar to the Address Book application. It will be a list of all owned books, sorted alphabetically by title. We've decided to mimic Address Book's main screen so that users will instantly feel familiar with our application.
- Because users may own hundreds of books, we will allow them to quickly navigate through this list of books by writing letters in the same way they navigate through the Address Book.
- To allow the user to quickly determine whether he or she owns a particular book, we will provide filters that allow special displays: All books by a particular author, all books on a particular subject, all books currently on loan.
- The initial screen will provide ways to enter a new book or to see more details about a book.
- The Details screen will have fields for title, author, and category and will have a way to loan a book to a friend.
- The main screen will have a way to indicate which books are out on loan so that the user can quickly scroll through the list and see which books are out on loan.
- To ease data entry, we'll provide a conduit that integrates with a desktop application so users can enter most of their

books on the desktop and synchronize them down to the handheld. People often buy several books by the same author, so we'll provide name completion for the author name based on existing authors in the database.

- We'll also integrate with other built-in applications. Loaning a book to a friend is linked to the Address Book as the friend's name and contact information is likely to be included there. We might integrate with Date Book by entering an event on the date that a friend is to return a book; however, because many people do not assign due dates when they lend to friends, we would make this an optional feature enabled by a user preference rather than a feature that is always available.

Recall that in our technology-focused design approach, we immediately came up with an Edit and Delete button in addition to the New button (plus scanning, beaming, and buying books through the wireless Internet—ideas we all but eliminated in the previous steps). In focusing on what tasks the user will perform most often, the Edit and Delete buttons did not come up in the conversation. Information about a book does not change, so the Edit button isn't as important as it is in, say, the Address Book. Book lovers rarely throw away books either. However, we should provide some way for users to correct mistakes in data entry, so we might provide Edit and Delete menu items or command buttons on a subscreen.

Develop the Initial Design Concept

After you have described the feel of the application, it is time to develop the look. Begin by focusing on the initial design concept. The initial design focuses on the main screens. At this point, selection of user interface elements comes into play. However, the exact details of the user interface may not be pinned down and subscreens, such as alerts, may be ignored until later on.

It's important to consider the usage frequency and likelihood of your user scenarios. More frequently used commands and settings should be easier to find and faster to execute.

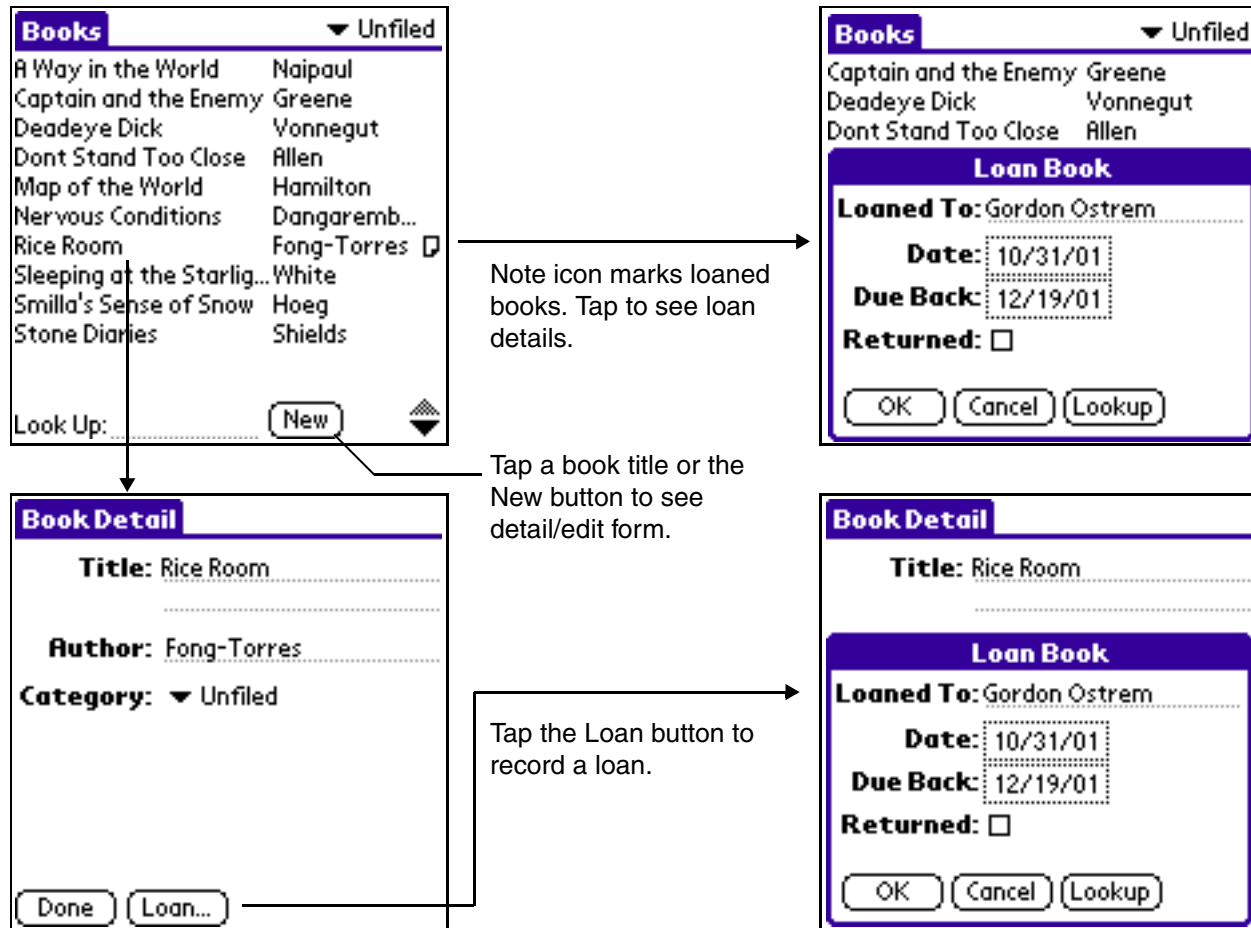
To make your application's important features easily accessible, choose the appropriate user interface element. Element selection is covered in detail in the remaining chapters of this book.

Palm OS Application Design

The Design Process

[Figure 1.5](#) shows the initial design concept for the Books application. Notice how it differs from the initial design shown for the wrong approach ([Figure 1.4](#) on page 12).

Figure 1.5 Initial design concept



Because we took the time to study our users' wants and needs, we know that the major problem they are having is managing the hundreds of books in their collections. The likely and frequent tasks from our user scenarios involve the user wanting to see what books they own and where those books are. Because the list of books can grow to be lengthy, we've decided that the most important tasks are the presentation of the list of books and the ability to enter new books. The main screen helps the user navigate the list of books by allowing them to filter the display according to categories of the

user's own choosing and by writing in the Look Up field to navigate to a particular book. It uses the note icon so that the user can easily scroll through the list of books and see which ones are out on loan. Tapping the note icon shows more information about the loan. The loan information shows to whom the book is loaned, when it was loaned, when it is due back, and provides a Lookup button so that the user can quickly go to the contact information for this person in Address Book if the person needs reminding that the book is due.

We will also provide other screens: a Preferences dialog that allows the user to sort by author name instead of title and a Search screen that allows them to search for all books loaned to a particular person. Because these tasks were described as likely but infrequent in our user scenarios, they are accessed through menu items instead of through command buttons.

You may only have sketches at this point, but it is appropriate to begin usability testing your interface at this stage to ensure that what you've designed meets the user's needs. You can mock up a user interface using technology as simple as note cards and have the user walk through common tasks. After you've performed a set of usability tests, refine your design concept and retest as you complete the next stage.

Complete the Design

After the main concepts have been agreed upon by everyone involved in the project (which might include human interface designers, marketing, and engineering), the design can be solidified. At this point, you should design every screen for the application, know how to get to every screen, have all user interface elements properly aligned and spaced on the screen, and design the menus. All error states are also defined.

This phase should coincide with the alpha build of your software. Thus, after the alpha release, you can have more user testing and feedback, which will refine your design for the beta build and then for the final build.

Notice that we complete the design during alpha development. Many developers are tempted to leave user interface design until much later in the product development cycle, but doing so is a mistake. If you wait until the end, you have solidified your initial

Palm OS Application Design

The Design Process

design in the code you have written. The longer you wait to change the design, the harder it becomes to do so, and the easier it becomes to rationalize leaving the interface as-is.

Fitting In

In the previous chapter, you learned what makes the Palm OS® platform unique, and you learned a process by which you can design a successful user interface for the Palm Powered™ handheld. This chapter documents the next step: making sure that your application integrates well with the Palm Powered handheld environment.

When users work with a Palm OS application, they expect to be able to switch to other applications, have access to Graffiti® writing or Graffiti 2 writing and the onscreen keyboard, access information with the global Find, receive alarms, and so on. Your application will integrate well with others if you follow the guidelines in this chapter. This chapter covers:

- [User Interaction with Palm Powered Handhelds](#)
- [Integrating with the Application Launcher](#)
- [General Application Layout Guidelines](#)
- [General Application Behavior Guidelines](#)
- [Becoming Compatible Worldwide](#)

User Interaction with Palm Powered Handhelds

Before you begin designing an application, you need to understand the different ways that users interact with their Palm Powered handhelds:

- Graffiti or Graffiti 2 writing
- Onscreen keyboard
- HotSync® operations
- External keyboards
- Hard keys on the handheld
- Icons in the input area

Fitting In

User Interaction with Palm Powered Handhelds

- Controls in the application

This section covers each of these methods in more detail.

Graffiti or Graffiti 2 Writing

Graffiti characters or, on newer models, Graffiti 2 characters are written in the input area on the digitizer (see [Figure 2.1](#)) and appear on the screen at the cursor location. The user specifies the cursor location by tapping directly on the screen with the stylus.

Figure 2.1 Input area

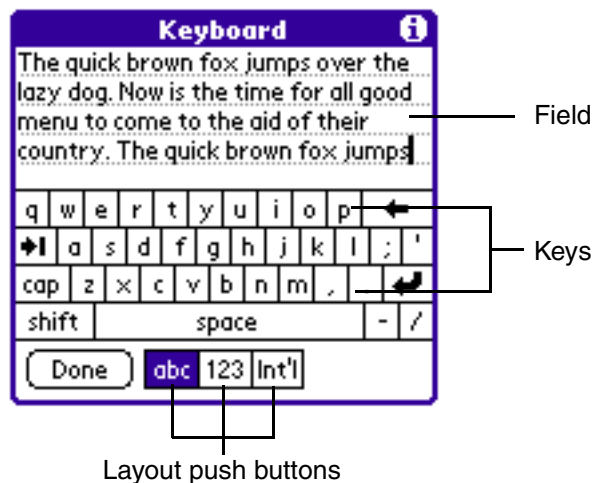


Users enter text and numbers by writing them in the input area.

Onscreen Keyboard

When the insertion point is in a text field, the user can open the onscreen keyboard by tapping on the letters “abc” or “123” in the lower corners of the input area. The keyboard dialog appears (see [Figure 2.2](#)).

Figure 2.2 The keyboard dialog



Field

Keys

Layout push buttons

The dialog displays any text currently in the field that contains the cursor. The user can then add to or modify the text as necessary.

HotSync Operation

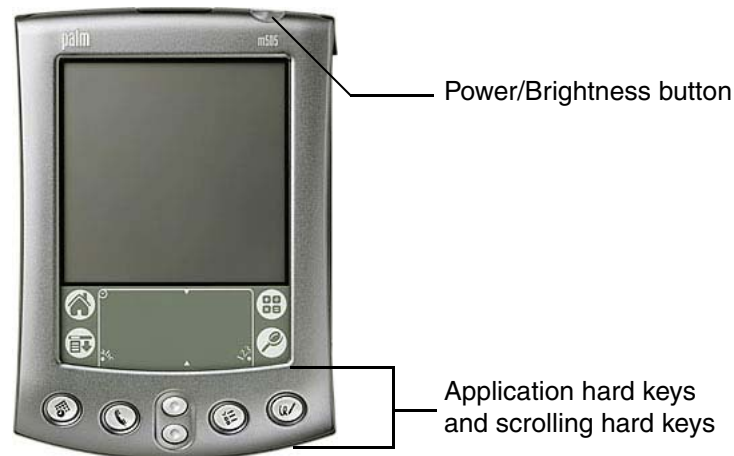
To enter data into a Palm OS application, users can enter data into the corresponding application on the desktop computer and then perform a HotSync operation. The data is loaded into the appropriate application on the Palm Powered handheld.

Hard Keys

A Palm Powered handheld usually contains four hardware application buttons (known as **hard keys**) plus two scroll buttons and a power button (see [Figure 2.3](#)). Some handhelds contain other hardware controls, such as a brightness or contrast adjust button or a jog wheel.

The four application buttons launch built-in applications such as Date Book, Address Book, To Do List, and Memo Pad. The exact list of applications launched by hard keys depends on the model.

Figure 2.3 Hard keys



Icons in the Input Area

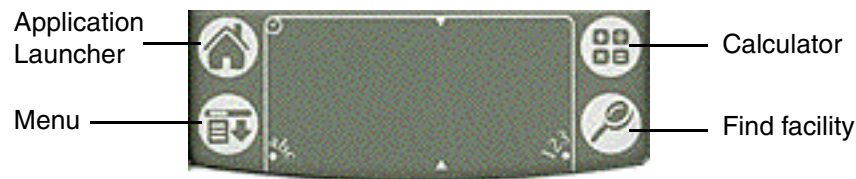
The four icons on either side of the input area (see [Figure 2.4](#)) usually display the Application Launcher, the current application's

Fitting In

Integrating with the Application Launcher

menu, the Calculator application, and the global Find facility. Some Palm Powered handhelds, such as handhelds sold in Japan, may have additional icons in this area. Others use a star icon in place of the Calculator icon and let the user choose the application that it opens.

Figure 2.4 Icons



External Keyboard

Many Palm Powered handheld models can accommodate an external keyboard. The keyboard attaches to the serial port. When it is attached and the appropriate software is installed on the handheld, users can type on the keyboard to enter text in any text field. They can also perform keyboard navigation such as tabbing to the next and previous fields.

Because external keyboards are an optional item priced separately, do not rely on your users having one. Instead, allow all lengthy data entry to be performed on a desktop computer.

Application Controls

The final way that users interact with their Palm Powered handhelds is by tapping the pen on controls in an application. Application controls are introduced in the section “[Controls](#)” on page 32 and described in more detail in the rest of this book.

Integrating with the Application Launcher

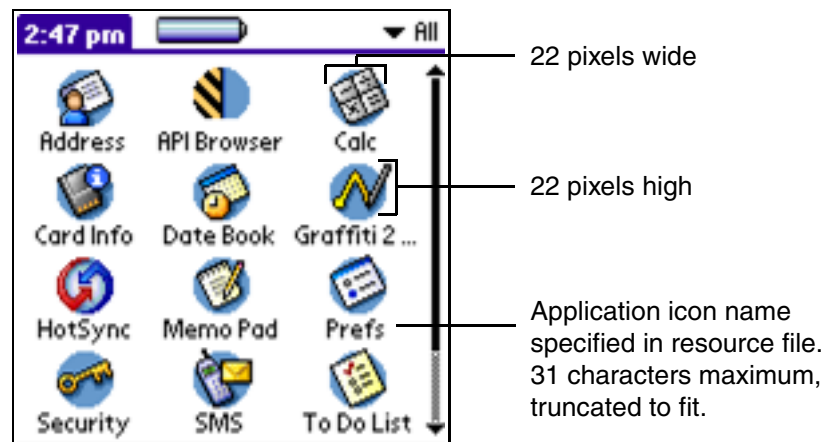
The Application Launcher (see [Figure 2.5](#) on page 27) is the screen from which most applications are launched. Users navigate to the Launcher by tapping the Applications icon in the input area. They then launch a specific application by tapping its icon.

To integrate well with the Application Launcher, you must provide application icons and a version string as described in the following sections. In rare cases, you might need to provide a default application category as well.

Application Icons

Provide two icons for the Launcher: a large icon for the icon view (see [Figure 2.5](#)) and a small icon for the list view (see [Figure 2.6](#)). Choose a short application name and an icon that's easy to recognize.

Figure 2.5 Application Launcher icon view



Make sure that the large icon has an identifiable center so that the user knows where to tap. Your icons should blend well with the icons for the built-in applications in addition to presenting the marketing message your company wants to project. The icons for the built-in applications follow these guidelines:

- The large icon has a blue circle in the background. On low-resolution screens, the circle is solid and 22 pixels wide by 22 pixels high. On high-resolution screens, the blue circle has a one-pixel anti-aliased blue border with a two-pixel white circle inside and then a solid blue circle in the center. The icon is the same physical size as the low-resolution large icon.
- On top of the circle, the main element is skewed 30 degrees. For example, the Date Book application shows a calendar

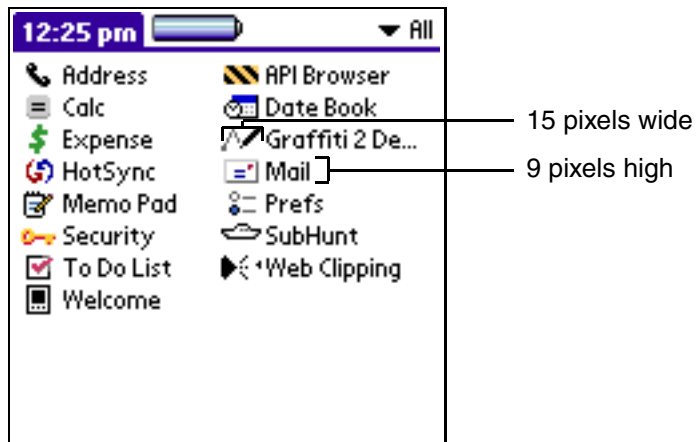
Fitting In

Integrating with the Application Launcher

skewed 30 degrees in addition to a clock at the normal angle.

- If the icon has a drop shadow, it should be shown as if there is a light coming from the top-left corner of the screen.

Figure 2.6 Application Launcher list view



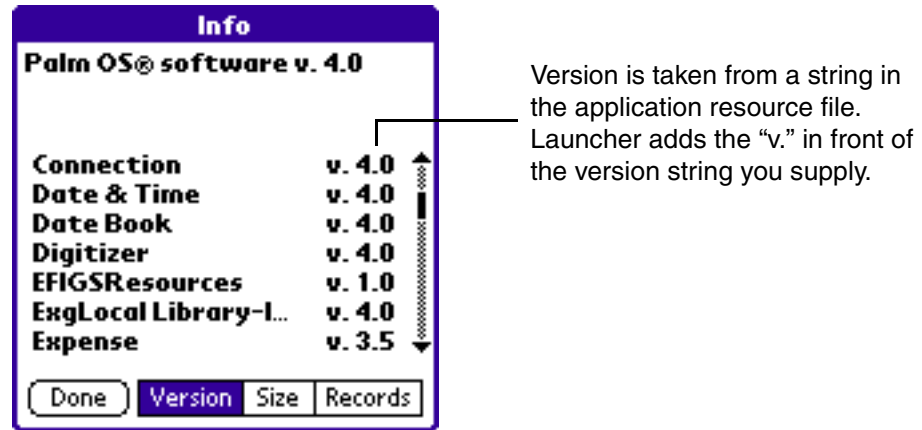
The small icon does not contain a circular background. The built-in applications typically take the skewed element from the large icon and show it at a normal angle.

For further guidelines on presenting graphics on Palm Powered handhelds, see [Chapter 8, "Color and Graphics,"](#) on page 159.

Version String

Provide a short (usually 3 to 7 character) string that gives your application's version number. This version string is contained in the application's resource file. It is *not* the version you provide to the PalmRez post linker. This string is displayed in the Info dialog (see [Figure 2.7](#)).

Figure 2.7 Launcher info



A version string should have the format:

major.minor.[stage.change]

where *major* is the major version number, *minor* is a minor version number, *stage* is a letter denoting the development stage (a for alpha, b for beta, or d for developer release) and *change* is the build number. Remove the *stage* and *change* numbers for the final release.

Default Application Category

Launcher divides applications into categories. If you specify a default application category in the application resource file, the Launcher places your application into that category when it is installed. If no application category is specified, the application is installed into the Unfiled category, and each user chooses where to file the application.

Most applications should *not* specify a default application category. Only specify one in these instances:

- Your application is intended for consumers and clearly belongs to one of the Launcher predefined categories (see [Table 2.1](#) on page 30).

Fitting In

General Application Layout Guidelines

- Your application is intended for a vertical market or you've created a suite of custom applications that work together to provide a complete custom solution.

In this case, you might define a custom category name.

Launcher creates the category if it doesn't already exist in the Launcher database.

Table 2.1 Launcher predefined categories

Default Launcher Category	Description
Games	Any game
Main	Applications that would be used on a daily basis, such as Date Book or Address Book
System	Applications that control how the system behaves, such as the Preferences, HotSync, and Security applications
Utilities	Applications that help the user with system management
Unfiled	The default category

Do not treat the default application category as something analogous to the Microsoft Windows Start menu category. On a Palm Powered handheld, the user is limited to 16 categories including Unfiled. Obviously, that limit would be quickly reached if each application defined its own category. Besides, placing your application in its own category only makes it harder to find and launch. Only assign a default category where it is a clear benefit to your users.

General Application Layout Guidelines

This section provides a general outline of what a Palm OS application looks like. The remaining chapters in this book provide more specific guidelines for the look of individual controls.

Main Application Forms

On Palm Powered handhelds, each screen of information is called a **form**. In general, the first form in your application is a **base form** that offers an overview of all available information and command buttons for the most frequently used commands. For example, the Address Book shows a list of all contacts (see [Figure 2.8](#)).

Figure 2.8 First form in Address Book

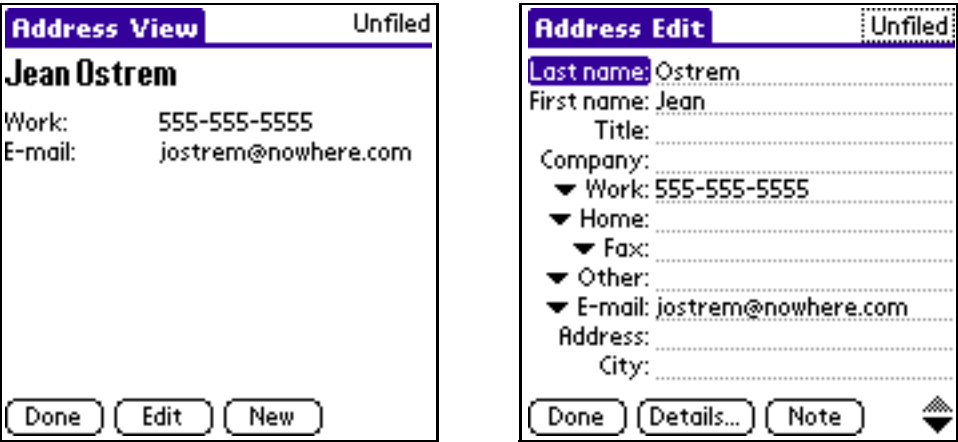


Address ▼ All	
Dugger, Mark	555-555-5555 W
Gossett, Brent	555-555-5555 W
Liu, Clif	555-555-5555 W
Maas, Brian	555-555-5555 W
Ostrem, Jean	555-555-5555 W
Parker, Dennis	555-555-5555 W
Parrett, JB	555-555-5555 W
Rathjens, Lisa	555-555-5555 W
Schaller, Anna	555-555-5555 W
Schneider, Susan	555-555-5555 W
Wilson, Greg	555-555-5555 W

Look Up: 🔍

Applications with a base form tend to also need one or more details forms to display more information about a record and to allow the user to edit the record (see [Figure 2.9](#)). Many applications use a single form for both purposes. Address Book uses two different forms for display and edit because many of the available fields in the Edit form are often left blank. Using a separate form to display a contact means users are more likely to see the information they need without having to scroll.

Figure 2.9 Detail forms in Address Book



For more detailed guidelines on creating all types of forms, see [Chapter 3, “Forms,”](#) on page 45.

Controls

Controls allow the user to perform commands, enter options, and edit data. [Table 2.2](#) shows the basic controls you might provide in your application and lists the chapter in this book that provides guidelines for the control.

Table 2.2 Controls

Control	Name	Main Purpose	For More Information
	Command button	Perform command	Chapter 4, “Executing Commands,” on page 71
<input type="checkbox"/> Show Due Dates <input checked="" type="checkbox"/> Show Priorities	Check boxes	Toggle state	Chapter 5, “Presenting Options,” on page 99
		Provide list of options that are not mutually exclusive	
Text.....	Field	Display or edit text	Chapter 6, “Displaying Data,” on page 127

Table 2.2 Controls (*continued*)





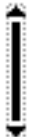

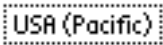


Control	Name	Main Purpose	For More Information
	List	Display text	Chapter 6, “Displaying Data,” on page 127
	Menu	Perform command	Chapter 4, “Executing Commands,” on page 71
	Pop-up list	Display a list of mutually exclusive options	Chapter 5, “Presenting Options,” on page 99
	Push buttons	Display a list of mutually exclusive options	Chapter 5, “Presenting Options,” on page 99
	Scroll bar	Navigate to new view Scroll the display	Chapter 7, “Scrolling,” on page 147
	Scroll buttons	Scroll the display	Chapter 7, “Scrolling,” on page 147
	Selector trigger	Display setting that user can change from modal form	Chapter 5, “Presenting Options,” on page 99

Table 2.2 Controls (continued)

Control	Name	Main Purpose	For More Information
	Slider	Adjust a setting	Chapter 5, “Presenting Options,” on page 99
	Table	Display data	Chapter 6, “Displaying Data,” on page 127

Control Placement

Place the most frequently accessed controls near the bottom of the form. The user interacts most often with the input area, with the icons in the input area, and with the hard keys on the handheld. Placing controls at the bottom of the form puts them as close as possible to the input area, making them quicker and easier to access.

People from Western cultures tend to read the screen from top to bottom and left to right. Therefore, anything important for users to read (rather than interact with) should be near the top of the form. Horizontally, you should arrange controls so that the leftmost control is the most important.

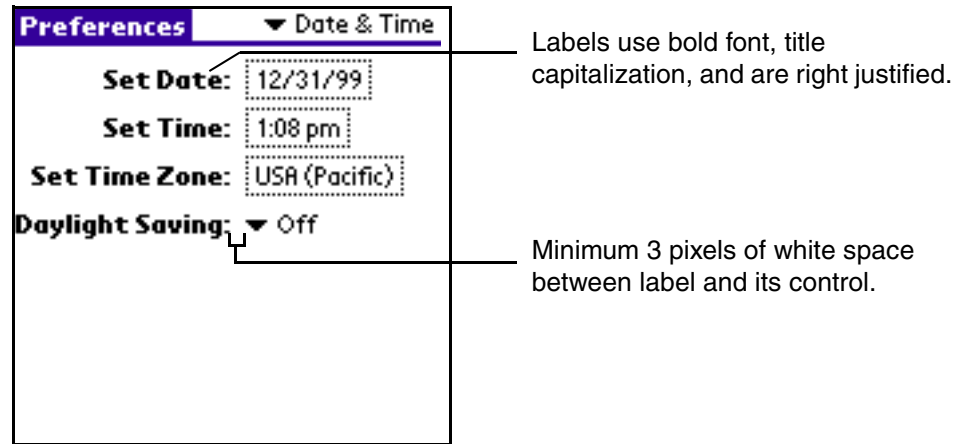
Do not clutter the screen. Running out of space is a usually a sign that simplification is needed. Squashing a lot of controls on the form by reducing white space is usually the wrong answer.

In most cases, use spacing instead of lines and boxes to separate user interface elements into logical groups. Lines and boxes add to screen clutter and actually make the small screen harder to read.

Labels

Provide a label for any control or option that requires further explanation. Right justify the labels and left justify the fields (see [Figure 2.10](#)).

Figure 2.10 Label guidelines



Use bold font and title capitalization for labels. That is, capitalize the first letter of each important word in the label in the same way you would capitalize the title of a book. For example, use “Set Date” as a label, not “Set date.” Never use all lowercase (“set date”) or all uppercase (“SET DATE”) for labels.

NOTE: This guide uses the term **label** to refer only to a textual label that appears outside of the user interface element. Sometimes, the text appearing inside the user interface element is also called a label. This guide refers to the text inside the user interface element as its **contents**. For example, “Daylight Saving:” is the label for the pop-up list in [Figure 2.10](#). The word “Off” is the pop-up list’s contents.

Fonts

Palm OS supports four different fonts: 9 point regular, 9 point bold, 12 point regular, and 12 point bold.

Use 9 point regular for the textual content of most controls, such as the text that appears inside of command buttons, pop-up triggers, and selector triggers.

Labels like those shown in [Figure 2.10](#) should be shown in 9 point bold font.

Fitting In

General Application Layout Guidelines

When displaying textual data, consider allowing users to set the font they would like to use. The standard font picker dialog resource included in the system allows the user to choose between 9 point regular, 9 point bold and 12 point bold.

You can also create your own custom font and include it in the system. Use a custom font only for displaying textual data. Don't use a custom font for labels or text inside of controls. For example, a document reader might show the document text in a custom font, but it should still use the system fonts for control contents and labels.

When creating your own custom font, remember to support both low-density and high-density screens. The high-density fonts are the same size as the low-density fonts, but they use the extra pixels to draw more smoothly and thus are easier to read. See the *Palm OS Programmer's Companion* for more information on creating fonts for multiple screen densities.

Graphical Controls

The command button and push button controls allow you to substitute a graphic for the button name.

Only use graphics on buttons if there is a common, clear precedent in similar desktop applications that you can leverage. For example:

- Web browsers use similar arrow buttons for browsing history.
- Document readers often use arrows to move to the next page.
- Most desktop applications use a clipboard to denote paste, scissors to denote cut, and so on.

If there is not a clear precedent, avoid creating your own graphical symbol for a command. It's difficult to make a graphic small enough to be useful while clearly conveying its meaning. Desktop applications often have several graphic buttons, but desktop applications often rely on tool tips to help the user learn what each graphic means. Palm OS does not provide tool tip support (see [Figure 2.11](#)).

Figure 2.11 Graphic buttons



What do these buttons mean?
There are no tool tips to help the user.

For more information on creating graphics for Palm OS, see [Chapter 8, “Color and Graphics,”](#) on page 159.

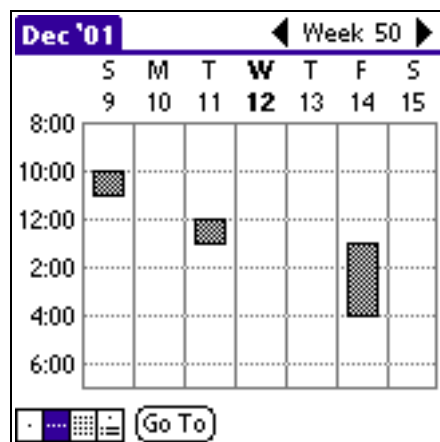
Custom Controls

Palm OS supports the creation of custom controls, called **gadgets**. You can create your own gadget if you have a need that the user interface guidelines don’t address. Your gadget must support the general design principles outlined in [Chapter 1](#), and your users must be able to intuit how to use it.

Some examples of gadgets include:

- Launcher uses a gadget to display the current battery level.
- Date Book uses a gadget for the main portions of its week view and month view forms (see [Figure 2.12](#)).
- A medical application might use a gadget that displays an outline of the human body so that users can tap to record where they are injured.

Figure 2.12 Date Book gadget



The main part of this form is a gadget. The gadget responds to taps by displaying the scheduled event or by allowing the user to create a new one.

Application Categories

Organize your application's database records into user-defined categories if that makes sense. Categories usually result in more efficient screen use. Users can switch between categories using a pop-up list or can display all records at once.

Categories are application-specific and are stored with the database. An application can have a total of 16 categories, including the Unfiled category. Users are allowed to create categories and are allowed to decide which records belong in which categories.

Your application can provide a set of predefined categories. Limit the number of categories you predefine so that your users are free to create categories that make sense to them. If you do provide predefined categories, allow your users to delete them so that they can add new ones. The Unfiled category must be one of your predefined categories. Do not allow users to delete the Unfiled category.

Provide a category pop-up list in the title bar of the base form to allow the user to change which category of records are displayed (see [Figure 2.13](#)).

Figure 2.13 Category pop-up list



Applications typically have a Details dialog that allows the user to change uncommonly accessed parts of a database record. The Details dialog is typically where you place controls to change the category. If your application has a form that edits a single record,

you can also supply a selector trigger on that form that displays a pop-up list from which the user can select a record's new category. See "[Selector Triggers](#)" on page 119 for more information.

If your application does not support categories but you still want to use the right side of a title bar for a pop-up list, that pop-up list must contain category-like information that filters the display. It should not perform some other operation. For example, a spreadsheet application might show a pop-up list in the title bar that allows the user to display a different worksheet if the spreadsheet has multiple worksheets.

General Application Behavior Guidelines

This section covers general guidelines for how applications should behave:

- [Launching the Application](#)
- [Exiting the Application](#)
- [Supporting Global Find](#)
- [Respecting User Preferences](#)
- [Allowing System Messages](#)

The remaining chapters in this book provide more specific guidelines for the behavior of individual controls.

Launching the Application

Your application must launch quickly. The typical user session with a Palm Powered handheld is one or two minutes long. Users do not want to add an extra several seconds to each session waiting for an application to launch.

During a normal launch, you should not display a splash screen. You might display a splash screen if this is the first time the user launched the application or if the application is running as a demo that will eventually expire and you want to show how much time is left.

You do not necessarily display the base form when users launch your application. Do so only if it makes sense. It's often better to

return to the place the user exited last. For example, Memo Pad always returns you to the memo you were last reading.

Displaying the location where the user exited is desired because it creates a seamless interface. If you make your application behave like the user never exited, users can think of all Palm OS applications as running at the same time.

Exiting the Application

Applications do not provide an exit command. On the Palm Powered handheld, users do not think in terms of exiting one application and then launching another. The paradigm is such that they consider all applications to be running at once and they can move between them at will. Users move between applications by pressing one of the four hard keys on the handheld or by tapping the Applications icon.

Allow the user to exit any form, including modal dialogs or alert dialogs, gracefully at any time. If the user has been editing data, save the data before the user exits the form. If the user is in the middle of editing a database record when the form is exited, your application should allow the user to exit the form and perform the least destructive operation on the data.

The built-in applications define a minimum set of data that must be present for the record to be saved. In most cases, it is one character of data. In the Expense application, the expense type must be set.

For example, suppose you are editing a contact in Address Book, and you need to look something up in another application. You can tap the Applications icon and launch another application without losing your data. Address Book saves the changes you have made even though you have not tapped the Done button at the bottom of the screen. When you return to Address Book, you see the main form, and the record you last edited contains your changes (see [Figure 2.14](#)).

Figure 2.14 Address Book with partial record

The figure consists of two side-by-side screenshots of a handheld application interface for an address book.

The left screenshot is titled "Address Edit" and has a status bar at the top right that says "Unfiled". It contains the following fields:

- Last name: **Ab**
- First name: _____
- Title: _____
- Company: _____
- ▼ Work: _____
- ▼ Home: _____
- ▼ Fax: _____
- ▼ Other: _____
- ▼ E-mail: _____
- Address: _____

At the bottom are three buttons: "Done", "Details...", and "Note", followed by a small downward-pointing arrow icon.

The right screenshot is titled "Address" and has a status bar at the top right with a dropdown menu showing "All". It displays a list of names and phone numbers:

Ab	
Dugger, Mark	555-555-5555 W
Gossett, Brent	555-555-5555 W
Liu, Clif	555-555-5555 W
Maas, Brian	555-555-5555 W
Ostrem, Jean	555-555-5555 W
Parker, Dennis	555-555-5555 W
Parrett, JB	555-555-5555 W
Rathjens, Lisa	555-555-5555 W
Schaller, Anna	555-555-5555 W
Schneider, Susan	555-555-5555 W

At the bottom is a "Look Up:" field followed by a "New" button and a small downward-pointing arrow icon.

If you are editing a record in Address Book and leave the application before tapping Done, Address Book saves your data. You see it listed in the main form when you return.

In certain vertical market applications, it is both possible and acceptable to write your application so that it never exits. This is an acceptable practice only as long as the application is preloaded onto the handheld for each user and the handheld will never be used as a personal digital assistant.

Supporting Global Find

Support the global Find facility (see [Figure 2.15](#)) if your application stores textual data in a database. The Find facility should perform a prefix-only case-insensitive search on each word in a database record. A prefix-only search finds matches only if a word begins with the string the user enters. For example, suppose an application contains two records, one with the text "And Then There Were None" and the other one "How To Use a Lathe." If the user enters the word "the," the Find facility will find matches in the words "Then" and "There" in the first record, but not in the word "Lathe" in the second record.

In many applications, a user can mark a record as private. A private record is stored in the database but may not be displayed. If your application supports private records, make sure that the records are unavailable to the global Find facility when the user has them hidden or masked.

Figure 2.15 Find



The Find facility matches on prefixes in displayed records only. Your application should support Find if it stores data in a database.

Respecting User Preferences

Palm OS has system preferences for the display of:

- Date formats
- Time formats
- Number formats
- First day of week (Sunday or Monday)

Be sure your application uses the system preferences for display of this type of information.

Palm OS also provides standard panels that can be used to select a date, a starting and ending time, or just a single point in time. If your application works with dates and times, use these standard panels. Users are familiar with their usage from the Date Book and Preferences applications.

Palm OS preferences also allow users to remap the hard keys and icons in the input area so that they launch different applications. Respect the user's preference for which application to launch; however, you can help the user override the preference. For example, suppose you provide a replacement for the Date Book application and you would like the Date Book hard key to display your application. You can display an alert dialog upon startup that asks the user if they want to remap the Date Book hard key so that it launches your application. If they tap the OK button, your

application can change the preference. If they tap Cancel, you should not ask the question again, but it is acceptable to provide a user preference that allows the user to remap the button at a later time.

Allowing System Messages

Allow the system to post these messages:

- Alarms
- Low-battery warnings
- System messages during synchronization

The normal event loop used by virtually all Palm OS applications allows ample time for the system to post messages and handle necessary events. You only need to take special care if your application performs a lengthy computational task. For example, if your application has a large database with greater than 20,000 records and it must search through each of these database records, you might want to check for system events every so often during this loop.

Becoming Compatible Worldwide

If you're planning to localize your application, it's best to plan for localization starting from the beginning. Keep in mind the issues listed below:

- Try to leave extra room for translation. Many languages simply take up more space than English does. German and French words, for example, are on average 25% longer than their English counterparts. Try to allow extra space for strings to display and larger modal forms than the English version requires.
- Be aware that items such as colors and symbols also need to be localized. For example, red is a warning color in English, but that is not true in all cultures. Avoid colors and symbols that may be offensive in other cultures.
- Dates and numbers are represented differently in other parts of the world. Always respect the user preferences that have been set for date, number, and time displays.

Fitting In

Becoming Compatible Worldwide

- Abbreviations may be the best way to accommodate the particularly scarce screen real estate on the Palm Powered handheld.
- Consider using string templates. For example, the Memo Pad application uses the template: Memo # of %. The application can replace # and % to change the text. Use as many parameters as possible to give localizers greater flexibility. Avoid building sentences by concatenating substrings together, as this often causes translation problems.
- Remember that user interface elements such as lists, fields, and tips dialogs scroll if you need more space.
- Write your code so that the application is easily localizable. For coding guidelines, see the chapter on localization in the *Palm OS Programmer's Companion*.

Forms

This chapter describes the different types of forms available. **Forms** are analogous to windows in a desktop application. They are containers for control and data objects. There are two basic types of forms:

- [Modeless Forms](#)
- [Modal Forms](#)

There are also several special cases of modal forms:

- [Alert Dialogs](#)
- [Progress Dialogs](#)
- [About Dialogs](#)
- [Tips Dialogs](#)

This chapter tells you how to choose which type of form to use and then describes behavior and appearance guidelines for each type of form. The form types are covered in order from most commonly used to least commonly used.

Choosing between Types of Forms

Most applications should use more modeless forms than modal forms. **Modeless forms** are full-screen forms that are the primary screens in your application. **Modal forms** are sub-forms used for a specific purpose.

Each application must have at least one modeless form, and most applications have more than one. For example the Address Book offers a main form that shows a list of contacts, an Address View form that shows the details for a single contact, and an Address Edit form that allows you to edit the contact information (see [Figure 3.1](#)).

Forms

Choosing between Types of Forms

Figure 3.1 Modeless forms in the Address Book

The figure displays three screenshots of modeless forms from the Address Book application. The first screenshot, titled 'Address', shows a list of names and phone numbers, with 'Ostrem, Jean' selected. The second screenshot, titled 'Address View', shows the details for 'Jean Ostrem', including work and email addresses. The third screenshot, titled 'Address Edit', shows the fields for editing the contact information, including last name, first name, title, company, and various phone and email fields.

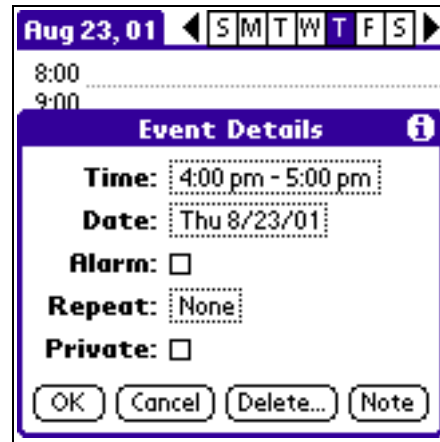
Address		▼ All
Dugger, Mark	555-555-5555 W	
Gossett, Brent	555-555-5555 W	
Liu, Clif	555-555-5555 W	
Maas, Brian	555-555-5555 W	
Ostrem, Jean	555-555-5555 W	
Parker, Dennis	555-555-5555 W	
Parrett, JB	555-555-5555 W	
Rathjens, Lisa	555-555-5555 W	
Schaller, Anna	555-555-5555 W	
Schneider, Susan	555-555-5555 W	
Wilson, Greg	555-555-5555 W	

Look Up:

Address View		Unfiled
Jean Ostrem		
Work:	555-555-5555	
E-mail:	jostrem@nowhere.com	

Address Edit		Unfiled
Last name:	Ostrem	
First name:	Jean	
Title:		
Company:		
▼ Work:	555-555-5555	
▼ Home:		
▼ Fax:		
▼ Other:		
▼ E-mail:	jostrem@nowhere.com	
Address:		
City:		

Modal forms (see [Figure 3.2](#)) are often shorter than the screen. They typically display only a few controls.

Figure 3.2 Modal form

Modal forms are shorter than modeless forms and align at the bottom. They are used only in specific instances.

Use Modal Forms Sparingly

Reserve the use of modal forms for instances where a user must acknowledge a certain state before they can continue. Some acceptable uses of modal forms are:

- Setting options, such as application preferences
- Editing data that is changed rarely. For example, marking a record as private is an infrequent operation. The built-in applications place the control for this on a modal form.
- Editing commonly used parts of a database record if the rest of the record does not consist of discrete fields.

For example, a Date Book appointment consists mainly of text, the name of the appointment. Setting an alarm for that appointment is common, but the alarm appears in the Details modal form with the uncommon fields because there is no room for it elsewhere. If your data consists of discrete fields similar to Address Book, it's better to edit all commonly used parts of your records in a single modeless form.

Avoid Modal Forms for Lengthy Data Entry

Avoid the use of modal forms for lengthy data entry. Sometimes, applications that perform data validation display a series of modal forms, one stacked on top of the other. Avoid creating such a user interface. If, for example, you require a user to create a record by

Forms

Modeless Forms

displaying three different modal forms, this requires a minimum of six taps: one tap to display each of the three forms, and one tap to dismiss each of the three forms. It is better to allow users to navigate between data entry forms at will, and it is better to have as few forms as possible.

If your application needs to perform data validation, it is tempting to use a modal form and not allow the user to leave the form until the record is complete and validated. Doing so is a poor design decision that may result in user frustration. The user may be distracted in the middle of entering the data. Suppose the user receives a phone call and needs to schedule an appointment with the caller. If you don't allow the user to leave your application without validating the data, he or she must cancel the data entry operation to schedule the appointment and re-enter all of the data later.

Instead, consider having your application behave in a similar manner as Memo Pad. Define a minimum set of data that must be present. If the user leaves the application in the middle of editing a record, save the record without validating it as long as that minimum data has been entered. When the user returns to your application, restore the edit form and restore the record that was previously being edited. Validate the data only when the user taps the Done button at the bottom of your form. If you have a corresponding desktop application, your conduit will have to check for and handle the case where a data record has not been validated. Your conduit can skip that database record and log the reason why it has not been synchronized.

Modeless Forms

A modeless form is the main GUI area of your application. Most forms are composed of a title bar at the top of the screen, an area for viewing or entering data, and one or more command buttons at the bottom that navigate to other forms (see [Figure 3.3](#)).

Figure 3.3 Modeless form parts

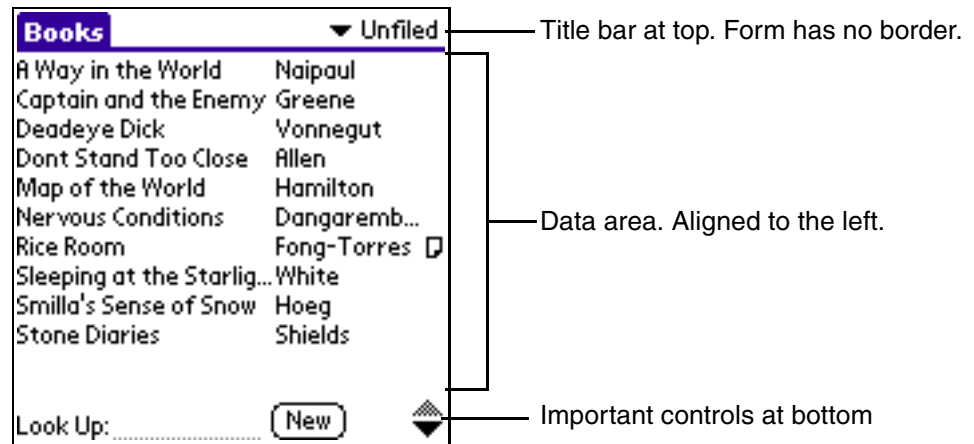


Table 3.1 Modeless form details

Dimension	Value
Width	160
Height	160
Top	0
Left	0
Border	None

System Supplied Behavior

Palm OS® supplies the following behavior for every modal form.

Always Full Screen

The currently displayed form takes up the entire screen and remains on the screen at the same location until the user chooses a new form. It's not possible for the user to resize, move, or collapse a form.

Keep in mind that screen sizes of future Palm Powered™ handhelds may vary. Newer handhelds may have high resolution screens (320 X 320 pixels), rotatable displays, and collapsible input areas. If the user collapses the input area, the application should move command buttons to the bottom of the screen and use the extra screen space to display additional data or an additional view of the

Forms

Modeless Forms

same data (such as a Details view). Using the portion of the screen previously covered by the input area to display additional command buttons or any other additional utility is discouraged.

Only Topmost Form Is Active

Only one form has the input focus at a time. This form is called the **active form**. If one form is drawn on top of the other (as is the case with modal forms), the topmost form is active until it is closed. Then the previous form becomes active.

Form Navigation

To navigate to another form, the user does one of the following:

- Taps one of the controls in the current form.
- Taps an icon in the input area that displays another application.
- Presses a hard key that navigates to another application.

Look and Feel

Follow these guidelines for the behavior and appearance of a modeless form.

Limit Form to a Screenful of Information

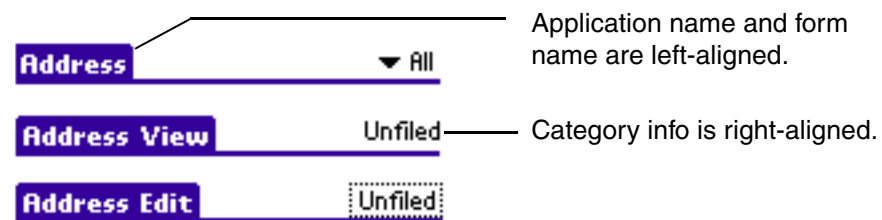
Try to ensure that a form contains only one screenful of information. However, if it does not make sense to break the information into multiple forms, a form can be scrollable. For example, the Launcher form is scrollable so that it can show all of the installed applications. The Address Edit form is scrollable so that you can edit all fields in a contact entry on a single form.

Use Title Bar to Orient the User

The title bar shows the title of the form flush left with the screen. If the form displays a database that uses categories, information about the category is flush right with the screen. Forms that display multiple records show a category pop-up list that can be used to filter the display. Forms that allow editing a single record display a selector trigger that allows you to change the category for that record. (See [Chapter 5](#), “[Presenting Options](#),” on page 99 for more

information about selector triggers.) The Palm OS draws a 2-pixel line on the border between the title bar and the data area. [Figure 3.4](#) shows example title bars.

Figure 3.4 Title bars



The purpose of the title bar is to orient the user. From a quick glance at the handheld, the user should be able to recognize the current application and current form within that application. Applications often look alike, so most should display a title bar that gives the name of the application followed by the name of the form. The Address Book follows this guideline, as shown in [Figure 3.4](#).

The title bar can also provide extra information. The Mail application shows how many messages there are in the current mailbox. (See [Figure 3.5](#).) The Address Book shows an icon that indicates when the user's business card is being displayed or edited.

Figure 3.5 Extra information on title bar



Title Bar Controls Are Primarily Informational

You can place controls in the title bar. The Date Book application shows a control that allows the user to select a day of the week. It is best if controls in the title bar are primarily informational in nature, that is, if they are the type of control that the user seldom changes. Keep in mind that users interact mainly with the input area of the handheld, which is below the bottom of the form. If your controls are in the title bar, users must move their hands as far away from the input area as possible. It is better to place the more commonly accessed controls near the bottom of the screen.

Forms

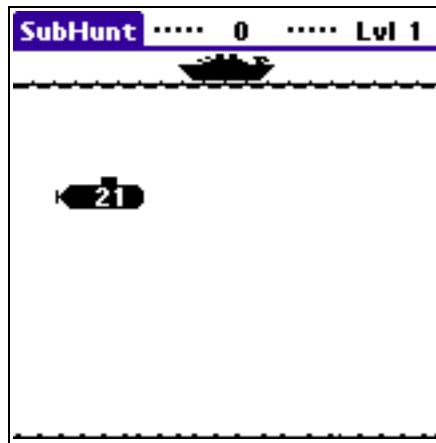
Modeless Forms

If your application does not support categories but your title bar has a pop-up list on the right side of the screen, that pop-up list must display category-like information that filters the display. Do not use this space for a pop-up list that has a purpose other than filtering the display.

Games Should Have Title Bars

On Palm Powered handhelds, even games have title bars (see [Figure 3.6](#)). The title bar makes an excellent location to show the game status: the score, current level, and so on.

Figure 3.6 A title bar for a game



Games should have title bars. They can use them to show the game status.

Breaking the Rules

This section points out a few of the applications that break modeless form guidelines and tells you if doing so was appropriate.

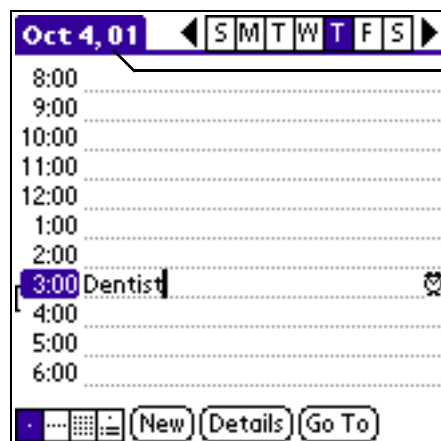
Nonstandard Title Bars

If the form's data area is so distinctive that it cannot be mistaken for another application, you may use the title bar to display something other than the application and form name. However, the information in the title bar still should orient the user.

For example, the Date Book main form (see [Figure 3.7](#)) uses its title bar to show the current date, the day of the week, and controls that allow users to navigate to a different date. It can do so because the main form is clearly an appointment calendar for a certain date. The

user would gain little if the Date Book showed a title that said “Date Book Day View” and had the current date and the day selection controls below the title bar.

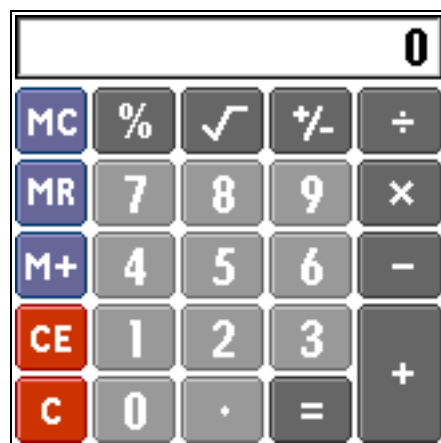
Figure 3.7 Other orienting information in title bar



In Date Book, the title bar shows the current date. Doing so conserves space without confusing the user because the form is distinctive.

In rare circumstances, the title bar may be skipped altogether, but only if the form cannot possibly be mistaken for some other application and the user could not be confused about its purpose. The Calculator application does not display a title bar (see [Figure 3.8](#)). This provides more room for the numeric display area and the buttons, allowing for buttons large enough to be tapped with fingers instead of the stylus.

Figure 3.8 Modeless form with no title bar



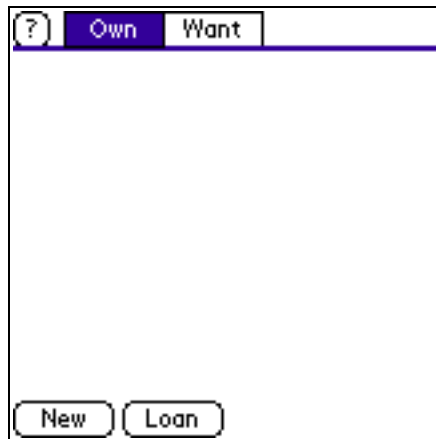
The Calculator omits the title bar to provide larger buttons. There is no confusion because the form is distinctive.

Forms

Modeless Forms

Note, however, that if users see a title bar, they expect to see some form of a title, whether it is the current date or time or the title of the form. Resist the temptation to include the title bar but omit the title in favor of adding other controls to your application (see [Figure 3.9](#)).

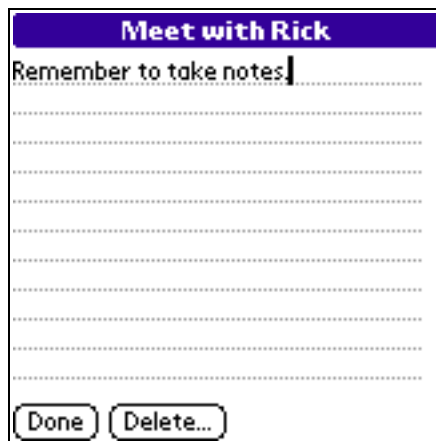
Figure 3.9 Bad example of title bar



If you have a title bar, you should have a title. This interface is confusing, especially to new users.

The Note view (see [Figure 3.10](#)) used in Address Book, Date Book, and To Do is another example of a modeless form that breaks the title bar rules. In this case, you should never follow its example. The Note view shows a dark background for the entire length of the title bar and centers its title rather than left-justifying it. This makes the Note view look like a modal form without a border.

Figure 3.10 Bad example of title bar



The Note view is not a modal form, but its title bar makes it look like one.

The Note feature was designed early in the development of the Palm OS and its built-in applications. It uses the name of the appointment, contact, or task as its title. At the time, it was thought that left-justifying the title made the form look strange. In retrospect, there were several better ways to solve this problem without breaking the guidelines. It would be better to make the Note view a modal form to underscore that this form simply changes one field in a database's record, or to use a modeless form whose title began with the word "Note."

Modal Forms

A **modal form**, or **dialog**, is a sub-form that is used for a specific purpose. Modal forms are displayed on top of the current form. Modal forms have a different title bar style than modeless forms, and they have a border (see [Figure 3.11](#)).

Modal forms should be far less common in your application than modeless forms. Their main use is for setting options, such as application preferences.

Figure 3.11 Modal form

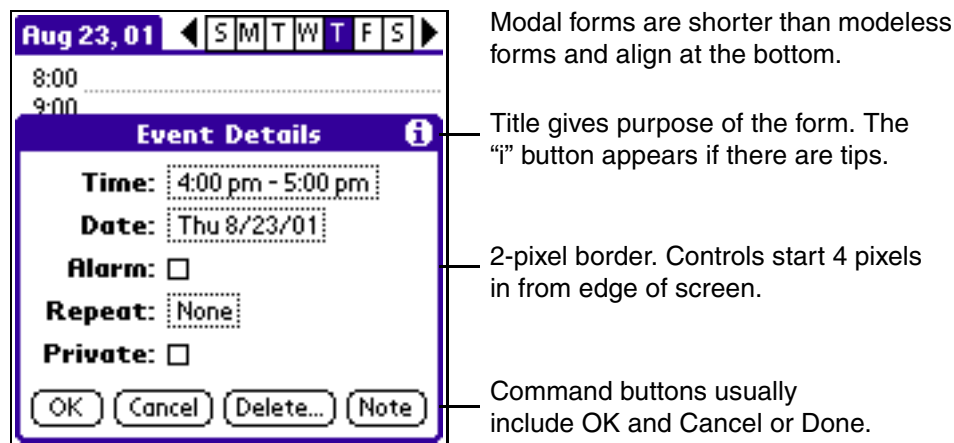


Table 3.2 Modal form details

Dimension	Value
Width	156 (to allow for the border)
Height	20 to 141 or 156 if it must be > 141 Always make as small as possible
Top	158 – <i>height</i>
Left	2
Border	2 pixels on each side

System Supplied Behavior

The Palm OS ensures that the topmost modal form is the only form that receives pen down events or keystrokes. The user cannot, for example, click the main form's title bar to dismiss a modal form. Also, the global Find facility is not available while a modal form is displayed.

Look and Feel

Follow these guidelines for the behavior and appearance of a modal form.

The User Can Exit at Any Time

You must allow the user to exit the application by tapping an icon in the input area or pressing a hard key even while a modal form is displayed. When you create the modal form, you specify a default button. If the user exits the application without tapping a button on the modal form, the application behaves as if this button were tapped. The default button should be the one that results in the least destructive action. Typically, this is the Cancel button, but in some cases the OK button is least destructive.

Make Modal Forms as Short as Possible

Although modal forms are always the width of the screen, they are not always the height of the screen. It's best to make the modal form

shorter than the screen so that the main form, appearing below the modal form, provides context. Try to make the modal form as short as possible. Allow at least 3 pixels of space between the main form's title bar and the top of the modal form. If the modal form obscures any portion of the main form's title bar, make the modal form the full size of the screen. (See [Figure 3.12](#).)

Figure 3.12 Full screen modal form



There should be 3 pixels of space between the two title bars. Because there is not, Select Business Card should be full screen.

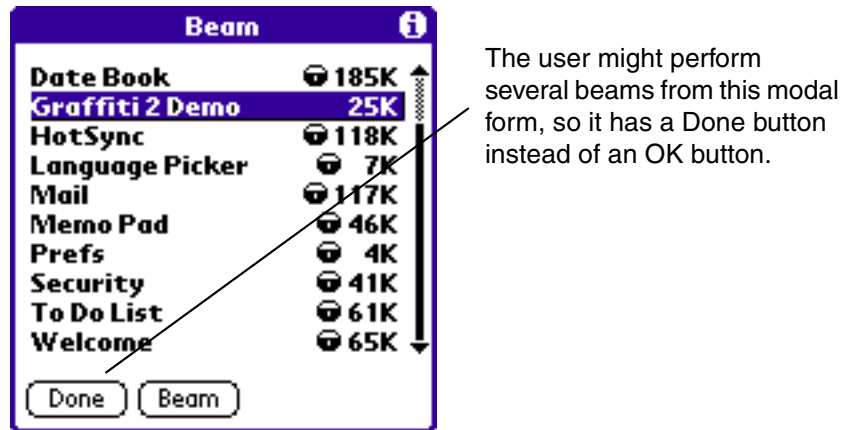
Always align the modal form with the bottom of the screen. Modal forms align with the bottom of the screen for two reasons. First, it allows the main form's title to be visible, providing more context for the user. Second, it places the modal form's controls closer to the input area.

Buttons on Modal Form

Most modal forms have an OK button in the bottom left and a Cancel button immediately to its right. They may have other buttons aligned with these buttons as necessary (see [Figure 3.11](#) on page 55).

If the modal form performs more than one action, use a Done button to dismiss the form rather than OK and Cancel. For example, you might beam more than one application from the Launcher's Beam form (see [Figure 3.13](#)). In this case, a Cancel button does not make sense because you can't cancel a beam after it has finished. The OK button is renamed Done in this case because users tap the button to indicate that they are done beaming applications.

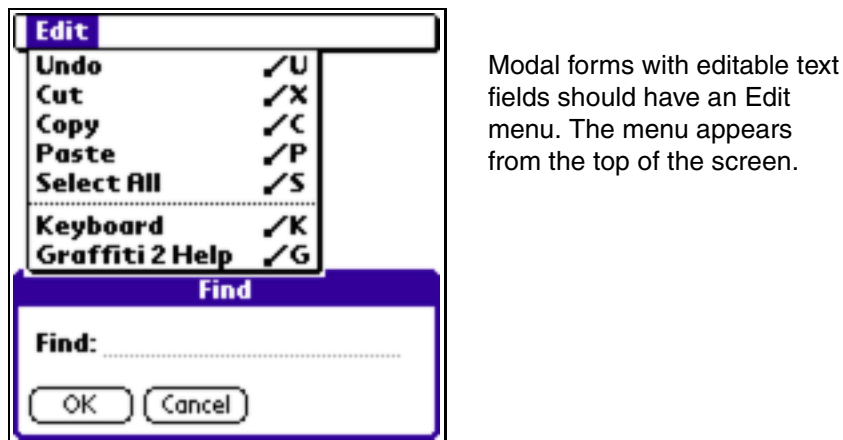
Figure 3.13 Modal form with Done button



Edit Menu Required if Form Has a Text Field

If the modal form has an editable text field, you must associate a menu bar with it, which should contain the Edit menu at minimum. This allows support for the copy and paste shortcut keys. The menu always displays from the top of the screen (see [Figure 3.14](#)).

Figure 3.14 Menu on modal form



A shift indicator is required as well as an Edit menu for forms with editable text fields. See "[Fields](#)" on page 129 for more information.

Breaking the Rules

This section points out a few of the applications that break modal form guidelines and tells you if doing so was appropriate.

Modal Forms Without OK Button

Your modal form may exclude the OK button if it helps minimize the number of required taps. Most modal forms have more than one control or have a text field. The OK button is required to let the application know when the user is done entering information. However, consider the Go To Date form in Date Book (see [Figure 3.15](#)). This form is dismissed as soon as the user has tapped one of the dates. Because choosing a date is the only purpose of the form, the OK button has been removed.

Figure 3.15 Modal form with no OK button

This modal form omits the OK button to minimize the number of taps for date selection.

Modal Forms that Don't Exit

It's possible for a user to miss a modal form entirely. Suppose the user becomes distracted immediately before the form is displayed, the handheld powers off, and then, to turn it back on again, the user presses one of the hard keys. The hard key dismisses the modal form by simulating a default button tap.

In most cases, the default button should protect the user from any unintended consequences of not seeing the modal form. For example, if the form was a Delete confirmation dialog, the default

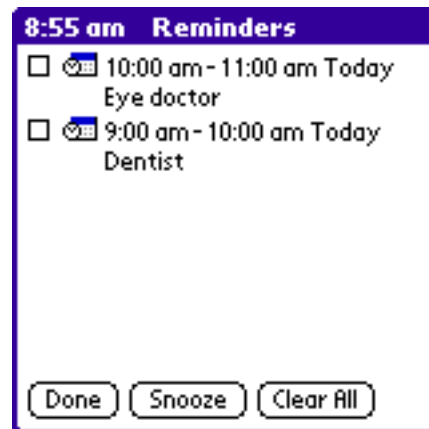
Forms

Alert Dialogs

button would be the Cancel button, so not seeing the modal form is not destructive.

If a default button is not enough protection for the user, you can create the modal form such that it is not dismissed until one of its buttons is explicitly tapped. Usually, this is only the case when the form presents vital information to the user. The low battery alert dialog, for example, must be explicitly dismissed because its intent is to inform the user of impending data loss. The Date Book alarm dialog and the Attention Manager dialog (see [Figure 3.16](#)) also must be explicitly dismissed because they present information that the user has requested to see at a specific time. If the user were to miss this information, it would be more annoying than having a dialog that must be explicitly dismissed.

Figure 3.16 Attention Manager dialog



This dialog must be explicitly dismissed because the user would not want to miss the information it contains.

Alert Dialogs

An alert dialog is a special case of a modal form. Alert dialogs are used in specific instances and have more rigid rules about appearance. Alert dialogs have no other controls on them besides the command buttons used to dismiss the alert (see [Figure 3.17](#)). They display an alert icon and a message.

Figure 3.17 Alert dialog

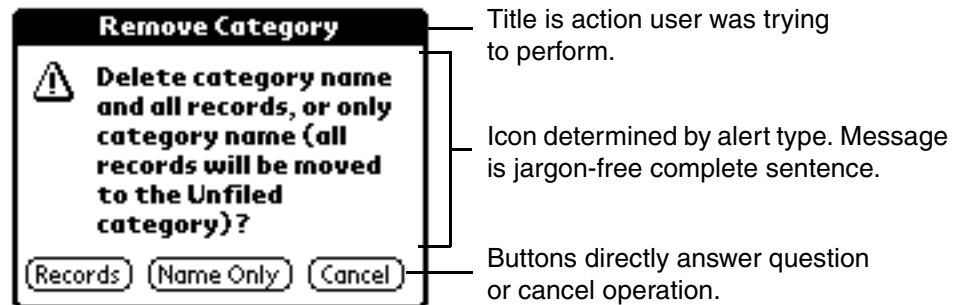


Table 3.3 Alert dialog details

Dimension	Value
Width	System determined
Height	System determined
Top	System determined
Left	System determined
Border	2 pixels on each side

Types of Alerts





Use an alert dialog to display an error condition, to prompt the user for a response, to inform the user of an event, or to warn the user of an upcoming event (such as low battery level).

[Table 3.4](#) describes the four possible types of alerts, when to use them, the icon to display on each, and an example message.

Forms

Alert Dialogs

Table 3.4 Alert dialog types

Type	Icon	Definition	Example Message
Information		Lowest-level warning. Action shouldn't or can't be completed but doesn't generate an error or risk data loss.	An alarm setting must be between 1 and 99.
Confirmation		Confirm an action or suggest options.	Your option to receive beamed data is turned off. Do you want to turn it on now?
Warning		Confirm a serious or potentially dangerous action.	Are you sure you want to delete this entry?
Error		Attempted action generated error and/or cannot be completed.	The battery is too low for this operation.

Look and Feel

Follow these guidelines for the behavior and appearance of an alert dialog.

The User Still Can Exit at Any Time

The rules for an alert dialog's behavior are the same as those described in "[Modal Forms](#)" on page 55. Specifically, you must allow the user to press a hard key or tap an icon in the input area to exit the application while an alert is displayed. If so, your application behaves as if the default button on the alert were tapped. You specify which button is the default when you create the alert.

Title Describes Operation That Caused Alert

The title of the alert dialog should describe the operation that the user was trying to perform. For example, alert dialogs about the beaming operation have "Beam" in the title bar. An alert dialog about deleting an address from the Address Book is titled "Delete Address."

Message Is a Complete Jargon-Free Sentence

The message you display should be a complete sentence. It should describe the problem in plain language and, where possible, describe how to correct the problem. It should not contain computer jargon. “Disk full” is an example of a poor alert message. Palm Powered handhelds do not have disks, and users may not know what a disk is. A better message would be “The storage area is full.”

Buttons Directly Answer Question

If the dialog simply displays a message, it should have a single button named “OK”. If the dialog asks a question about how to proceed, avoid using generic buttons such as “Yes” and “No” or “OK” and “Cancel” wherever possible. When users are provided a message such as “Save data?” with buttons “Yes,” “No,” and “Cancel,” they are often confused about which button to choose. Instead, it is better to make the affirmative answer to the question more explicit. Repeat the verb of the message (see [Figure 3.18](#)). If the message is “Delete this record?” the buttons should be “Delete” and “Cancel,” making it clear that choosing “Delete” deletes the data. Resort to “Yes” or “No” only if the alternative is cumbersome. The confirmation alert “Make this address your business card?” uses “Yes” and “No” buttons for this reason.

Figure 3.18 Alert dialog buttons



The buttons on an alert should directly answer the question. Avoid the use of generic names such as “OK” or “Yes.”

Use Attention Manager for Reminders

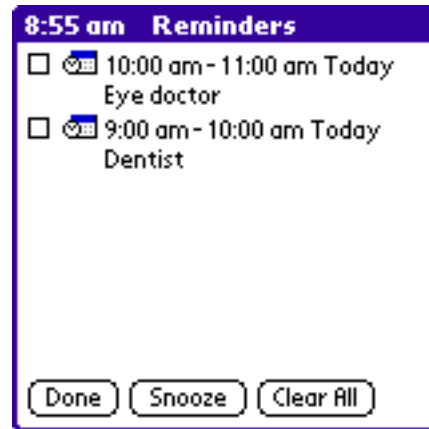
If your application must remind the user of a particular event similar to the way the Date Book reminds users of appointments, use the Attention Manager dialog (see [Figure 3.19](#)) on Palm OS 4.0 and later releases rather than displaying an alert dialog. To use the

Forms

Alert Dialogs

Attention Manager, you don't create an alert dialog resource. You write code that tells the Attention Manager what text to write in its dialog. The Attention Manager displays all current and past due reminders in a single dialog.

Figure 3.19 Attention Manager dialog



Use the Attention Manager instead of a separate alert dialog if all you want to do is remind the user of an upcoming event.

In earlier releases, each application that reminded the user displayed a separate alert dialog for each reminder. Most users have run into the situation where they have had their handheld turned off all day, turn it on, and are presented with a series of alert dialogs about appointments that have passed. The Attention Manager displays a single dialog so that the user can dismiss all past reminders at once. It also has the ability to perform non-visual reminders such as playing sounds or vibrating the handheld.

The Attention Manager is only designed for attempts to get attention that can be effectively suspended. It is *not* suitable for the following:

- Anything requiring an immediate response, such as the “put away” dialog that is used during beaming or a request to connect to another user
- Error messages
- To Do List items or incoming email messages

Breaking the Rules

If you want any other control on the alert dialog, you must create a modal form rather than an alert dialog. Use the alert icons on the modal form to make it look more like an alert dialog. [Figure 3.20](#) shows an example of a modal form that behaves as an alert dialog. In this way, the dialog can include a check box that allows the user to archive the deleted record.

Figure 3.20 Modal dialog disguised as alert



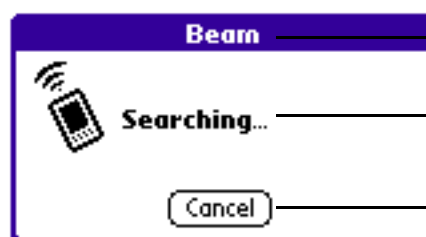
If it makes sense to have a control on an alert, make a modal dialog and include an alert icon on it.

If your application runs only on Palm OS 3.5 and later, you can create an alert dialog that has a text field on it. You can use this, for example, to have a user enter a password to confirm the operation.

Progress Dialogs

Despite your best efforts, you may find that some lengthy operations cannot be avoided. For example, if you need to connect to a network or use the serial or infrared ports, there is no way to make a connection happen more quickly. In these cases, use a progress dialog (see [Figure 3.21](#)) to indicate that an action is taking place.

Figure 3.21 Progress dialog



Title is action being performed.

Message and icon that update periodically.

Only contains a Cancel button.

Table 3.5 Progress dialog details

Dimension	Value
Width	System determined
Height	System determined
Top	System determined
Left	System determined
Border	2 pixels on each side

The progress dialog has a bitmap and a message that are updated periodically to indicate that the operation is progressing. For example, when you beam a business card to another Palm Powered handheld, the Beam progress dialog shown in [Figure 3.21](#) toggles between two beaming bitmaps and updates the message in the following order: “Initializing,” “Starting,” “Searching,” “Connected,” and “Sending.”

Progress dialogs dismiss themselves when the operation completes, so they do not include an OK or Done button. They do have a Cancel button so that the user can cancel the operation.

The purpose of using a progress dialog is to placate the impatient or unknowing user who may believe that the handheld has crashed. The progress dialog informs the user of what the system or application is doing. If your task does not have multiple steps and if it takes more than one second but less than three seconds to complete, you may instead use a simple Please Wait form centered in the screen. The Launcher uses such a form while it gathers database information when you choose the Info menu item (see [Figure 3.22](#)). However, as with progress dialogs, the Please Wait form is to be avoided if at all possible. Use it rarely if at all.

Figure 3.22 Please Wait form



For “quick” long tasks, a simple Please Wait form may do, but be careful not to display it for too long.

NOTE: Do not display the Please Wait form for more than a few seconds. If you do, the user will still believe that the handheld has frozen.

About Dialogs

Use the about dialog (see [Figure 3.23](#)) to display the version number and copyright notice for your application. All applications should have an about dialog. The about dialog must be accessible from the Options menu on the main form. Its title is “About *application*” where *application* is the name of your application. It has an OK button centered on the form and no Cancel button.

Forms

About Dialogs

Figure 3.23 About dialog

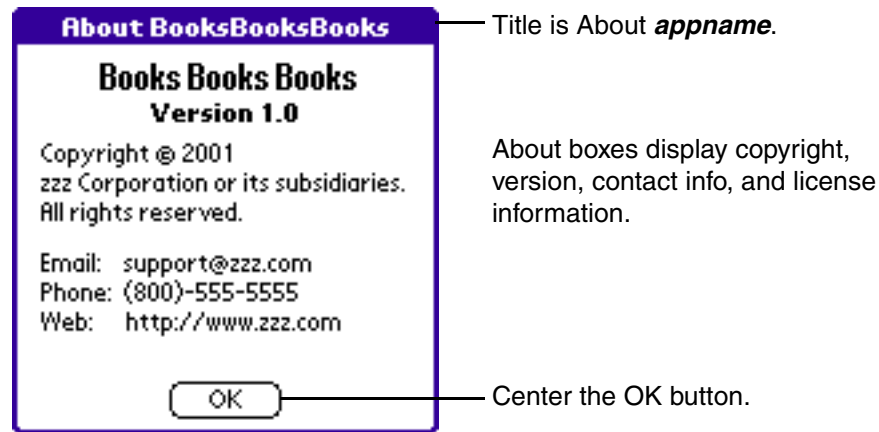


Table 3.6 About dialog details

Dimension	Value
Width	156
Height	156
Top	2
Left	2
Border	2 pixels on each side

The about dialog should contain the following types of information:

- Copyright notice
- Version number
- Terms of usage: Can your program be copied? Is it freeware, shareware, or commercial software?
- Registration information, such as the user's license number if they have paid or the amount of time remaining on the demo if they have not paid
- A phone number or web site where they can go for support

Tips Dialogs

Tips dialogs (see [Figure 3.24](#)) are displayed by modal forms when the user taps the “i” button in the corner of the modal form.

Figure 3.24 Tips dialog



The Tips dialog is created by the system; you just supply the text.

Table 3.7 Tips dialog details

Dimension	Value
Width	System determined
Height	System determined
Top	System determined
Left	System determined
Border	2 pixels on each side

The tips dialog provides further assistance to a user who is wondering how to use a particular modal form. It is a good idea to associate a tips dialog with most, if not all, modal forms in your application other than progress or about dialogs.

The tips dialog should briefly describe each field on the form, the purpose of each field, and any special symbols on the form. Some tips dialogs also tell the user about shortcuts that are otherwise hidden. For example, in Address Book a user can attach a note to a contact by tapping to the right of the contact’s information in the list

Forms

Tips Dialogs

view. The user learns this if he or she reads the tips in the Address Book Details dialog.

There is little or no formatting to the string that you supply for the tips dialog. You cannot, for example, change fonts or add a graphic to the tips. You also cannot change the dialog's title.

TIP: It's common to show a bulleted list in the tips dialog. To type a bullet in Constructor for Palm OS, set the string so that it shows all characters as hexadecimal, type "95", and then deselect the "View as Hex" option.

Tips dialogs should only contain necessary and useful information. Because of the compact nature of the text presented in the dialog, this dialog is not a good venue for providing users with conceptual background material they might need to know to use your application effectively. Such information is best presented in an external user manual, not on the handheld itself. Tips dialogs also are not a place for extra information, such as programmer credits that you might see in an about dialog.

Executing Commands

This chapter discusses user interface elements that, when selected, perform an action in your application. In other words, these elements execute commands.

There are two main methods of executing commands on Palm OS®:

- [Command Buttons](#)
- [Menus](#)

The first section tells you how to choose which of the elements listed above is best for your application. The rest of the chapter describes behavior and appearance guidelines for each element.

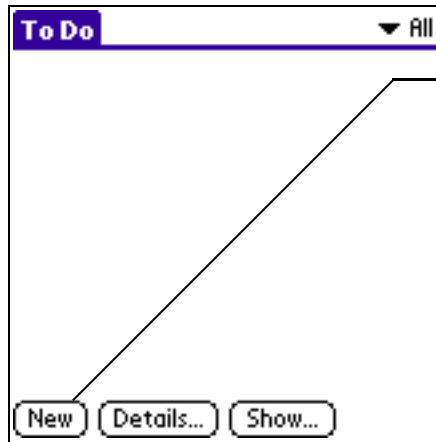
Choosing between Buttons and Menus

Choosing which application features should have command buttons and which should have menu items is a fine art. Command buttons provide instant access but take up valuable screen space (see [Figure 4.1](#)).

Executing Commands

Choosing between Buttons and Menus

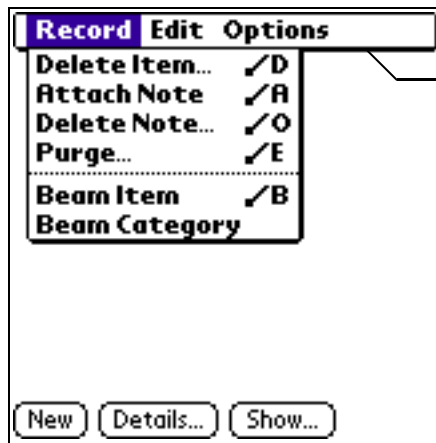
Figure 4.1 Command buttons



Command buttons provide instant access to important commands.

Menu items do not take up screen space, but they are hidden and require more taps to execute (see [Figure 4.2](#)).

Figure 4.2 Menus



Menus are hidden until the user taps the title bar or the menu icon.

In general, you use command buttons for common commands and menus for uncommon commands. This section provides further guidelines to help you choose.

Limit the Total Number of Commands

Remember that a new user must be able to pick up a Palm Powered™ handheld and successfully navigate the system within five minutes. If you have too many buttons, you may overcrowd the screen and ultimately confuse the user. If you have too few buttons,

you may end up with an interface that is not instantly understandable or easy to learn. Likewise, providing too many menu commands confuses the user.

Instead of focusing on how many command buttons and menu items you can squeeze in, focus on providing the minimum number required for your application. Running out of space is usually a sign that simplification is needed. Remember that users use 20% of a desktop application's features 80% of the time. You should concentrate on finding the top 20% of the features required by your users and provide commands that implement only those features.

Remember also that you are limited by the size of the screen. Realistically, you can fit only five or six buttons on a form. You can fit three or four menus on each form, and each menu can have at most 13 commands.

Use Buttons for Important Tasks

Use command buttons for important tasks, tasks that are essential to your application and that are frequently performed. Reserve the use of menu items for the less important, less frequently accessed commands.

Using command buttons for essential tasks serves two purposes: They make the commands visible to all users, and they minimize the number of taps required to execute the commands.

For example, all of the built-in applications have a New command button on the main form so that even new users can see how to create a record. Navigating between the main forms of the application is another important task that is nearly always worthy of a command button.

Many users do not even know that menus are available in Palm OS applications. If you place an essential command like the New command in a menu, those people won't find it.

Note that context plays a role in deciding which commands are important and which are not. A command may be important enough to deserve a command button on one form of your application but not on another or a command may be important in one application but not another. For example, most games use a

Executing Commands

Choosing between Buttons and Menus

menu item for the New Game operation because it is more important to devote the entire screen to the playing of the game.

Use Menus for Destructive Commands

Just as essential tasks should have a command button so that new users know where to find them, destructive commands should be relatively hidden so that a user does not execute the command by mistake. Menu items are good for this purpose because menus are a somewhat hidden interface.

An alternative is to maximize the number of taps for a destructive operation. For example, deleting a contact from the Address Book is performed by a command button, but it requires three taps to reach the Delete button.

Just as context plays a role in determining which commands are important, it also plays a role in determining how difficult it should be to execute destructive commands. Some applications may want to elevate the importance of the Delete command by placing its button at the same level as, for example, the New or Edit button. Consider an application that loads digital images from a camera into the handheld. In such an application, you may want the Delete button to have the same accessibility as the Edit button so that the user can quickly and easily free up space on the handheld.

All destructive operations should display a confirmation alert before actually performing the task.

Don't Duplicate Commands

Don't provide both a command button and a menu item for the same operation in the same form.

In desktop applications, it's common for there to be both a New menu item and a New command button on the toolbar, for example. On Palm Powered handhelds, screen space is so limited that such an interface is discouraged. Use command buttons for the important, common commands only. Use menu items for infrequent commands only.

Keep in mind, however, that each form in an application can have a different menu. It's acceptable for one form's menu items to

duplicate command buttons that appear on a different form. For example, in Memo Pad the main form has a New button to create a new memo. The Memo Edit form has a menu item that performs the same function. This way, the user does not have to return to the main form to create a second memo.

For further discussion on duplicating commands, see “[Duplicating Menu Items](#)” on page 95.

Remember the Goal: Minimize Taps

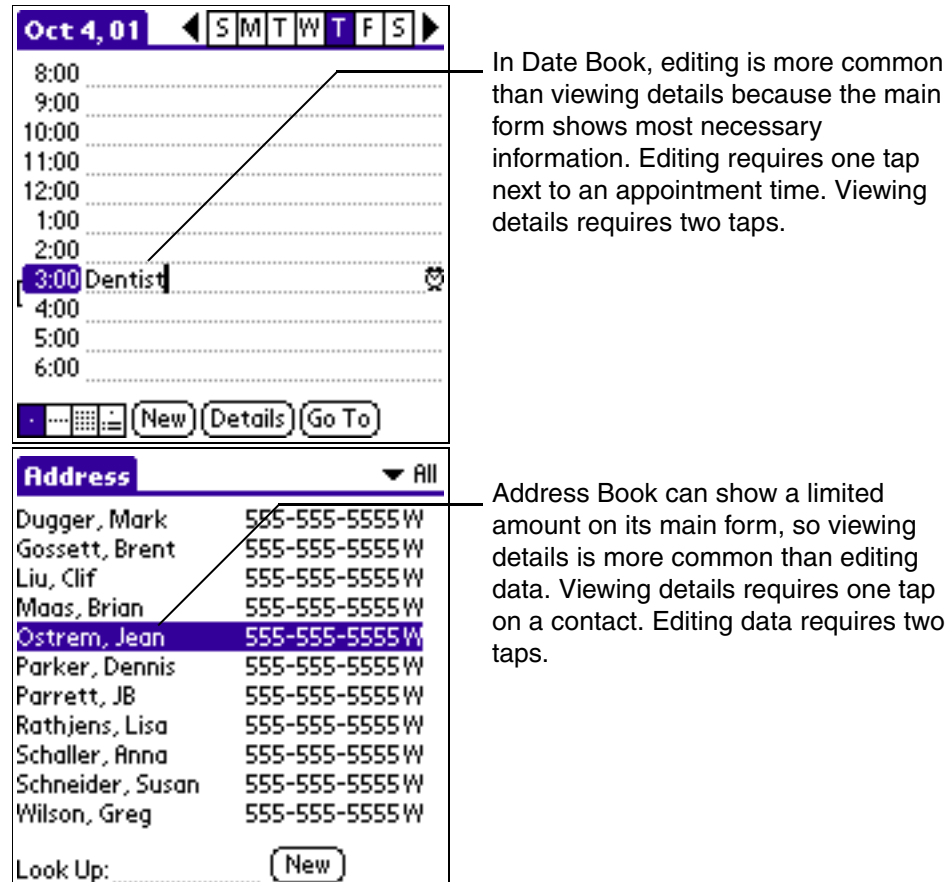
Don’t use a command button or a menu when some easier method of performing the task is available.

Each of the built-in applications allows you to edit an existing record and to display a record’s details. They use different means for performing these tasks depending on what fits that application best. The Date Book and To Do List applications allow you to edit a single record in the main form simply by tapping it. For viewing the record’s details they provide a command button, so this action requires one more tap than editing the record does. Address Book reverses this functionality: viewing the record’s details requires a single tap on the record. Editing the record requires the user to tap the record and then tap the Edit button. (See [Figure 4.3.](#))

Executing Commands

Choosing between Buttons and Menus

Figure 4.3 Minimizing taps in built-in applications



Even though these applications use different means for editing an existing record and displaying a record's details, each uses the best means for its data. In Date Book and To Do List, you rarely need to see more information about a record than is shown in the main form. Therefore, editing the record is more common than viewing the details. In Address Book, viewing the details is more common than editing because Address Book's main form cannot display all relevant information about a contact (such as the contact's address). Each of these applications minimizes the number of taps for the most frequently performed operation.

Use Buttons for Commands Executed by New Users

Provide command buttons for tasks commonly executed by novice users if they are otherwise hidden.

In Address Book, tapping anywhere in the View form takes you to the Edit form. This is a power user feature; two quick taps on a contact listed in the main form takes you to the Edit form. Newcomers may not discover this feature right away, so Address Book still provides an Edit command button on the View form.

You do not have to provide a command button if the functionality is easily discovered, however. Tapping a contact name in the main form of Address Book has no command button counterpart. Newcomers can easily figure out this interface.

Don't Provide Save or Exit Commands

Most desktop applications have menu commands to save changes and to exit the application. These commands are so ubiquitous on desktop applications, novice Palm OS application designers try to fit them into their handheld applications. This is almost never a good thing to do.

You find no Save command in the built-in applications. Palm Powered handheld users are not used to saving changes before they switch to another application. Changes are always saved as they are made. Undo is available if the user makes a mistake in an editable text field, and destructive operations have a confirmation.

Palm Powered handheld users are also not used to exiting applications. On the Palm Powered handheld, users do not think in terms of exiting one application and then launching another. The paradigm is such that they consider all applications to be running at once and they can move between them at will. (In reality, the system only runs one application at a time, and the applications are written in a way that moving between them looks seamless. This is an implementation detail that the end user does not know about.)

Command Buttons

A command button (see [Figure 4.4](#)) is a button that has a rounded rectangular border and that performs a command. Place a single row of command buttons at the bottom of the form.

Figure 4.4 Command button dimensions

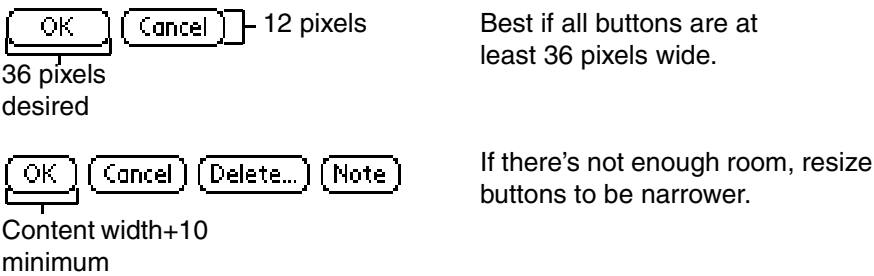


Table 4.1 Command button details

Dimension	Value
Width	36 or <i>content width + 10</i>
Height	12 or <i>font height + 3</i> if not using standard font
Top	147 (modeless forms) or $160 - \text{button height} - 1$ $\text{form height} - \text{button height} - 5$ (modal forms)
First Button Left ¹	1 (modeless forms) 5 (modal forms)
Button(<i>n</i>) Left ¹	$\text{button}(n-1) \text{ left} + \text{button}(n-1) \text{ width} + 6$
Border	1 pixel rounded rectangle
Content	Text or graphic that succinctly describes command

1. See “[Left-Align Buttons at the Bottom of the Form](#)” on page 79.

System Supplied Behavior

The user executes the button's command by tapping the button once with the pen or with a finger. When the user does so, Palm OS highlights the button while it is being pressed. Palm OS ensures that the command is not executed if the user drags the pen or finger outside of the bounds of the button before releasing it.

Look and Feel

Follow these guidelines for the behavior and appearance of a command button.

Never Require a Double Tap

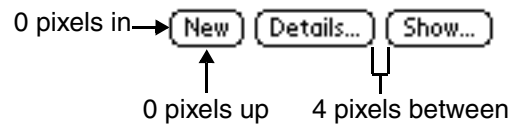
Never require a double tap on a command button, even as a power user feature. Reserve double taps for text fields.

Left-Align Buttons at the Bottom of the Form

Command buttons belong at the bottom of the form. Remember that the user is working most often in the input area. Because the user will tap these command buttons often, it is best to place them at the bottom so that they are nearest the user's pen.

On a modeless form, buttons and all other user interface controls begin at the left edge of the screen (see [Figure 4.5](#)).

Figure 4.5 Command buttons on modeless form

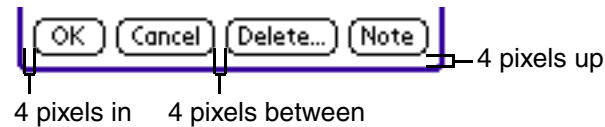


On a modal form, you must leave space between the edges of the form and the buttons to make sure that the button borders don't blend with the form borders (see [Figure 4.6](#)).

Executing Commands

Command Buttons

Figure 4.6 Command buttons on modal form



Borders are drawn outside of an object's boundaries. That means to align the button with the left edge of the form, the button must begin at pixel 1. Its border is then drawn at pixel 0. Similarly, to place 4 pixels between buttons, you must actually add 6 to the width and left origin of the button on the left. Adding 6 allows for the left button's 1-pixel border on the right and the right button's 1-pixel border on the left. See [Table 4.1](#) on page 78.

Use Succinct Names

The names of command buttons should be succinct and should clearly convey what the command button does. Remember, the user should be able to learn your basic interface in five minutes. That does not leave much time for learning by experimentation. It is best to use standard command names wherever possible so that the user can instantly recognize the button and know its purpose. [Table 4.2](#) lists some of the command buttons found in the built-in applications.

Table 4.2 Common command button names

Name	Minimum Width	Description
OK	23	Confirm action and return to previous form
Cancel	36	Revert the current form to the previous settings and return to the previous form
Delete	37	Remove record
Details...	44	Display Details dialog
Done	31	Return to previous form
Edit	25	Edit an existing record

Table 4.2 Common command button names (*continued*)

Name	Minimum Width	Description
New	27	Create a new record
Note	30	Add a note to a record

Use an ellipsis (...) in the button name if the user must provide more information before the operation is complete. In general, if the button displays a dialog other than an alert, about, or progress dialog, its name should have an ellipsis. The ellipsis can be omitted if you need more space.

TIP: Type three periods instead of using an actual ellipsis character. The character is not available in all fonts.

You can use graphics in the button instead of text (see [Figure 4.7](#)). If you keep the graphics small, you can generally fit more graphic buttons on the screen than you can text buttons. However, as with text, the graphic must instantly convey what the button does. On a desktop, application designers often rely on the tool tips feature to teach users what the buttons do. (The user can hover the cursor over the button, and a text box pops up that describes the button.) No such feature exists in Palm OS, so users will have to learn your icons by trial and error.

Figure 4.7 Graphic buttons



What do these buttons do? There are no tool tips to help the user.

Never Dim a Disabled Button

Never dim or gray out a button to show that it does not apply to the current situation. If the button depends on a certain user context, display an alert dialog that explains why the button does not apply. For example, To Do List has buttons at the bottom of the main form that apply to the currently selected task. If the user has not selected

Executing Commands

Command Buttons

an item, tapping one of the buttons results in an alert dialog explaining how to select an item (see [Figure 4.8](#)).

Figure 4.8 Alert dialog when button does not apply



To Do List displays an alert dialog when the Details button is tapped and no item is selected.

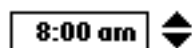
If the button depends on certain hardware capabilities, such as the ability to connect to a network, you can test for those capabilities before the form opens and hide the button if it does not apply.

Repeating Buttons

If you want a button to repeatedly perform an action while the user holds down the pen on it, use a repeating button instead of a command button. Use of repeating buttons is relatively uncommon. Two of the more common uses of repeating buttons are:

- As scroll buttons that scroll the display (see “[Scroll Buttons](#)” on page 153)
- To increment or decrement a value (see [Figure 4.9](#))

Figure 4.9 Repeating buttons



The up arrow and down arrow are repeating buttons. The user may press and hold to repeatedly increment or decrement the time.

Repeating buttons may look exactly like command buttons, but usually the border is removed from a repeating button.

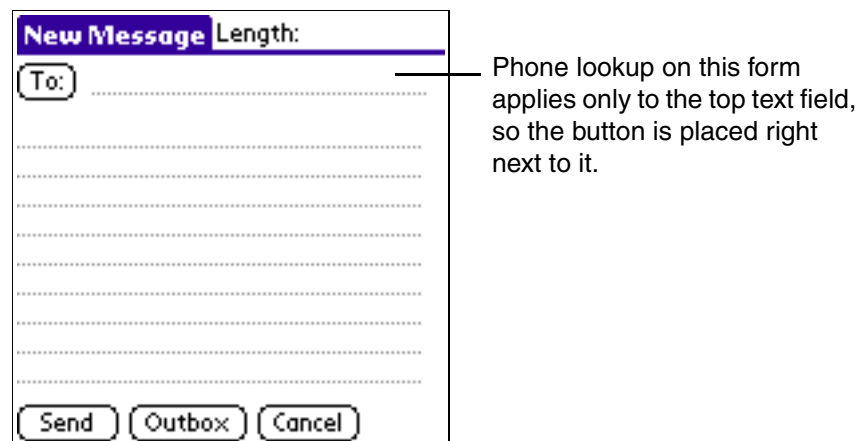
Breaking the Rules

This section points out a few of the applications that break command button guidelines and tells you if doing so was appropriate.

Placing Command Buttons Elsewhere on the Form

If a command button applies only to a particular field on a form, you can place that button next to the field instead of at the bottom of the form. [Figure 4.10](#) shows a form in the SMS Messenger application. In this form, the To command button looks up a phone number in the Address Book. Placing this button next to the field it modifies helps the user understand what the button does. If this button were placed at the bottom of the form, the word “To” would not be a descriptive name for the button. Instead, it would have to be named “Phone Lookup,” which still is not as clear as naming the button “To” and placing it next to the field.

Figure 4.10 Command button elsewhere on form



Before breaking the command button guidelines in this manner, consider using a selector trigger in combination with a modal form. The selector trigger could replace the command button and text field. It could display a modal form containing a text field and a list of contacts. Users could either choose a phone number from the list of contacts or write the phone number directly in the text field.

For the SMS Messenger application, a selector trigger is less desirable because it would force people to always use the phone

Executing Commands

Command Buttons

lookup feature. Many people might want to bypass the phone lookup feature if they know a phone number by heart. A selector trigger would not allow for this behavior.

See “[Implementing a Combo Box](#)” on page 104 for further discussion of this style of interface.

Centering Command Buttons

You may notice that every form in Palm OS and its built-in applications has left-aligned command buttons with the exception of about dialogs and progress dialogs (see [Figure 4.11](#)). About and progress dialogs each have a single button that is centered on the screen. Do not model any other form after about or progress dialogs. Consistently left justifying buttons is best for the user; the user expects all buttons to appear in a certain location. Note that single-button alert dialogs always left justify the button.

Figure 4.11 Bad example of button placement



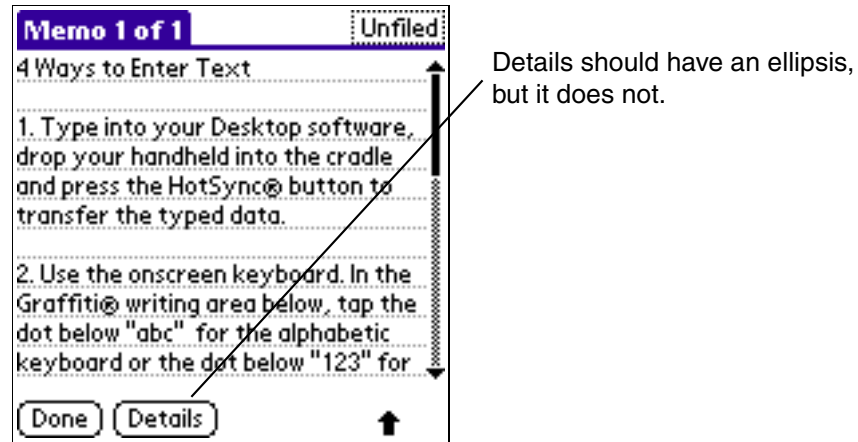
Do not follow the example of progress dialogs. Do not center a button on a form.

Omitting the Ellipsis

The ellipsis character is used inconsistently within the built-in applications. Don't follow what the built-in applications do. Try to follow the ellipsis guideline as it is presented in this chapter: Any button that displays a modal dialog other than an alert, about, or progress dialog should have an ellipsis in its name. The ellipsis should only be omitted if you need to make space.

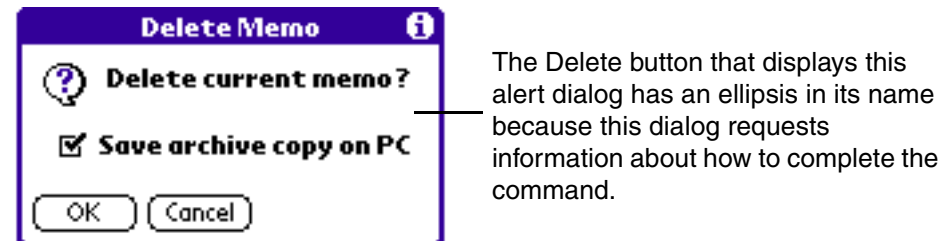
You may notice that in the built-in applications sometimes omit the ellipsis even when there is room for it (see [Figure 4.12](#)).

Figure 4.12 Button that requires ellipsis



The use of an ellipsis on the Delete button also can confuse newcomers to the platform. In general, a Delete command would not have an ellipsis because it displays only a confirmation dialog in response. Many of the PalmSource built-in applications, however, correctly use the ellipsis on Delete because they do require extra information to complete the operation (see [Figure 4.13](#)).

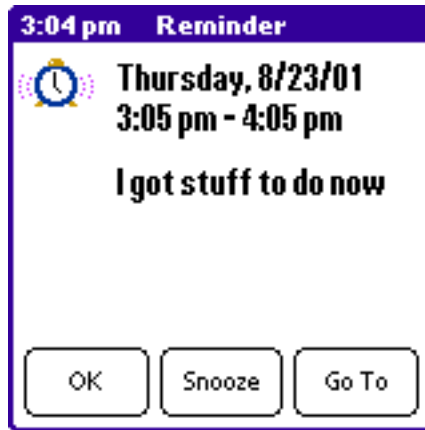
Figure 4.13 Delete confirmation with extra information



Buttons with Nonstandard Height

The Date Book alarm dialog uses buttons that are taller than normal so that users can dismiss the dialog as quickly as possible (see [Figure 4.14](#)).

Figure 4.14 Date Book alarm buttons



The Date Book alarm dialog uses buttons large enough that users can press them with their fingers.

It's quite common for this dialog to appear and the alarm to sound when the user has the handheld turned off. Users want to dismiss the dialog as quickly as possible and finish what they were doing. The large buttons allow for "finger navigation," enabling virtually all users (not just those with small fingers) to dismiss the dialog without having to get out the stylus.

Use large buttons such as these only if your application has a dialog that is likely to display when the user does not have the stylus out. Once the user has the stylus in his or her hand for some other task, the advantage of the large buttons is lost.

Menus

A menu displays a set of commands that the user can perform (see [Figure 4.15](#)). The user either taps the form's title or the Menu icon to display the menu bar, then chooses an item from one of the menus. That command is executed.

Figure 4.15 Menu bar and menu



Table 4.3 Menu details

Dimension	Value
Width	System determined
Height	System determined
Top	System determined
Left	System determined
Border	Bold rectangle
Content	Adjective or verb that succinctly describes command

System Supplied Behavior

The menu bar is displayed when the user taps either the menu icon or the form's title.

Palm OS ensures the behavior of menus as outlined in [Table 4.4](#).

Table 4.4 Default behavior of menus

When...	Then...
User drags the pen through the menu	Command under the pen is highlighted
Pen is released over a menu item	That item is selected and the menu bar and menu disappear

Executing Commands

Menus

Table 4.4 Default behavior of menus (*continued*)

When...	Then...
Pen is released outside both the menu bar and the menu	Both menu and menu bar disappear and no selection is made
Pen is released in a menu title (Palm OS 3.5 and later only)	Menu bar and menu remain displayed until a selection is made from the menu
Pen is tapped outside menu and menu bar	Both menu and menu bar are dismissed but no event is posted

Menus are fairly simple. Palm OS provides no concept of having a submenu because of the limited screen space. Menus also are not scrollable.

System Displays Last Command Executed

To make it easier for users to perform commonly used commands, the system remembers the last menu item that the user selected (menu items selected using the command stroke followed by a shortcut character do not count). The next time the user displays the menu bar, the menu for the previously selected item is displayed and that item is highlighted.

The system only remembers the last menu item as long as the current form is displayed. If a new form is displayed, the current form's menu memory is erased. This means that if the result of the menu item is to display a new form, such as an about dialog, the command is not remembered.

Look and Feel

Follow these guidelines for the behavior and appearance of a menu.

Note that you have no control over the height and width of a menu. The system always determines a menu's boundaries at run time.

Each Form Can Have a Different Menu

Each form can have a menu bar associated with it. (The menu bar is optional; a form does not have to provide menus.) You can provide a different menu bar with a different set of menus for each form in the application.

Provide Options Menu with About Command

Provide an Options menu with an About menu item to display your about dialog. This menu must at least be present on the main form.

If your application has a Preferences dialog, the Preferences command also belongs on the Options menu.

Edit Menu Required for Text Fields

Provide the standard system Edit menu if the form has an editable text field. The Edit menu must look as shown in [Figure 4.16](#). You can add your own items below the last item.

Figure 4.16 Standard Edit menu



All forms with text fields, even modal forms, must have this Edit menu. It must contain these commands in this order and with these shortcuts.

NOTE: Newer handhelds use Graffiti® 2 writing and have “Graffiti 2 Help” as the last item in the Edit menu.

Menus Are Static

Keep menus and menu items static. Menu items should not gray or dim when inapplicable. Never remove a menu item nor change its name based on user context. If a menu presents commands that are dependent on certain hardware capabilities and the handheld does not contain those capabilities, hide the menu items before the menu bar is displayed the first time.

Naming Menus and Menu Items

The rules for naming menus and items within menus are similar to those used in desktop applications.

For the menu name, choose a single word that describes the type of items in that menu. Some common menu names:

- **Record**—The items are actions performed on the selected database record and can also create and delete records.
- **Edit**—The items change the text displayed on the form.
- **Options**—The items allow the user to control how the application behaves.

For the menu item name, use either verbs or adjectives. If the item performs an action, use a verb. If the item changes an attribute of an object on the screen, use an adjective. In some cases, it's appropriate to use a noun. Typically, nouns display a dialog with the same name, such as "Preferences." It is particularly important on Palm OS to keep the menu names short. You can manage around 20 characters per item name, but try to keep them much shorter than that. Twenty characters takes up the entire width of the screen, obscuring the entire form.

Use title capitalization in menu items and in all user interface items. That is, capitalize all important words in the item just as you would the title of a book. The menu item for attaching a note, for example, is "Attach Note," not "Attach note."

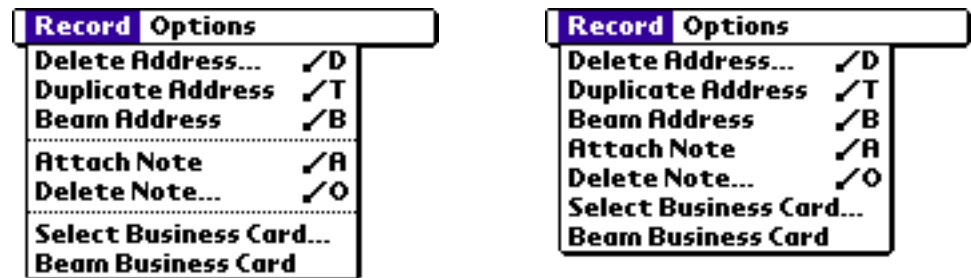
Use an ellipsis (...) in the menu item if the user must provide more information before the operation is complete. In general, if the item displays a dialog other than an alert, about, or progress dialog, it should have an ellipsis. As with command buttons, type three periods instead of using an actual ellipsis character.

Divider Lines Group Menu Items

If your menu contains a long series of commands (greater than five as a rule of thumb), try to group related commands together with a divider line. In [Figure 4.16](#) on page 89, you can see the Edit menu uses a divider to separate the usual cut, copy, paste, and undo operations from the menu commands that display the keyboard dialog and that display the Graffiti or Graffiti 2 Help dialog. By

providing dividers to group related menu items, you help the user learn and understand the interface. See [Figure 4.17](#).

Figure 4.17 Menu with dividers and without



The menu on the left is easier to grasp than the menu on the right.

Menu Shortcuts

A menu shortcut is an alternative to selecting a menu item. Users draw the command stroke (a diagonal line drawn upward from left to right) followed by a single character.

This support is present in all versions of Palm OS. In Palm OS 3.5 and later, a command toolbar is displayed. (On earlier releases, just the word “Command” was displayed.) The shortcut toolbar (see [Figure 4.18](#)) displays a status message on the left and buttons on the right. After entering the command character, the user has the choice of writing a character or of tapping one of the buttons on the shortcut toolbar. Both of these actions cause the status message to be displayed briefly and then cause the corresponding menu item to be executed.

Figure 4.18 Menu shortcut toolbar



There is no requirement to give every menu item a shortcut. Writing characters is difficult for many users, and mistakes are common. If the command is destructive (such as Delete), it is better not to provide a shortcut to avoid having the user inadvertently select it. If the destructive command is followed by a confirmation alert, you can assign a shortcut to it.

Executing Commands

Menus

NOTE: If you assign a shortcut, it must be unique through the entire application, not just the current menu system.

Avoid assigning letters with similar strokes to two different menu items. For example, the shortcut for Paste is P rather than the V used on most systems. One reason for this is that it is easy to confuse the stroke for V with the stroke for U, which is the shortcut for Undo. It could be detrimental to the users intending to Paste if they instead performed Undo by mistake. Another reason is that most Palm Powered handheld users do not use keyboards with their handhelds, and many may not even know how to type. For keyboard users, the V shortcut for Paste is easy to remember because the V key is right next to the C key. The V shortcut for Paste would not make sense to users who do not know how to type.

Consider menu shortcuts already assigned by other applications when assigning your own menu shortcuts. Make sure that you use the same shortcut if you include the same menu item. This makes the entire system easier for users to learn. [Table 4.5](#) shows the common shortcuts used by the built-in applications.

Table 4.5 Common menu shortcuts

Command	Shortcut
New Record	N
Delete Record	D
Beam	B
Undo	U
Cut	X
Copy	C
Paste	P
Select All	S
Keyboard	K

Table 4.5 Common menu shortcuts (*continued*)

Command	Shortcut
Graffiti Help/Graffiti 2 Help	G
Preferences	R

Buttons on the Shortcut Toolbar

You can include your own buttons on the toolbar for the more commonly executed menu shortcuts (see [Figure 4.19](#)). The buttons displayed on the toolbar depend on the user context. If the focus is in an editable field, Palm OS displays buttons for cut, copy, and paste on the shortcut toolbar. If there is an action to undo, the system also displays a button for undo.

Figure 4.19 Toolbar buttons



Only add a button to the toolbar if you define a shortcut for that command. Don't arbitrarily add extra buttons to the toolbar.

Limit the buttons displayed on the shortcut toolbar to four or five. There are two reasons to limit the number of buttons. First, you must leave room for the status message to be displayed before the action is performed. Second, consider that the toolbar is displayed only briefly. Users must be able to instantly understand the meaning of each of the buttons on the toolbar. If there are too many buttons, it reduces the chance that users can find what they need.

As described previously, the system already potentially displays four buttons by itself (for cut, copy, paste, and undo). You can suppress this behavior if you feel your users would benefit from seeing other buttons instead.









The system contains bitmaps that represent such commands as beaming and deleting records. If your application performs any of these actions, it should use the system bitmap. [Table 4.6](#) shows the system bitmaps and the commands they represent. If you use any of these, you should use them in the order in which they are listed in [Table 4.6](#), from right to left. That is, BarDeleteBitmap should

Executing Commands

Menus

always be the rightmost of these bitmaps, and BarInfoBitmap should always be the leftmost.

Table 4.6 System shortcut toolbar bitmaps

Name	Bitmap	Command
BarDeleteBitmap		Delete record
BarPasteBitmap		Paste clipboard contents at insertion point
BarCopyBitmap		Copy selection
BarCutBitmap		Cut selection
BarUndoBitmap		Undo previous action
BarSecureBitmap		Show Security dialog
BarBeamBitmap		Beam current record
BarInfoBitmap		Show Info dialog (Launcher)

Breaking the Rules

This section points out a few applications that break the menu guidelines and tells you if doing so was appropriate.

Including an Exit Command

If you are writing an enterprise application to be used only within a particular company, your users may insist on having an Exit command. These users are used to their desktops, and they feel uncertain about moving from application to application on a Palm Powered handheld.

In this case, it may be best to give your users either an Exit menu command or an Exit command button. Before you do so, make sure they know that when an application is exited (that is, when the system receives an application exit event), the system returns to the last screen displayed before the user launched that application. For most applications, the previous screen is the Launcher because it is used to launch most applications. If your application is launched by

one of the hard keys on the handheld, then exiting the application would not always take the user to the Launcher.

Even if your users insist on an Exit command, you should always design your application so that it can be exited in the normal means for a Palm OS application—by pressing one of the hard keys or by tapping the Applications icon—at any time. Consider this a power feature for your users. Your users may use the Exit command at first but will eventually become used to using the other buttons.

Never include an Exit command on an application intended for broad third party distribution.

Nonstandard Menu Names

You do not have to name the first menu “Record” even if it contains items that manipulate a single record in your database. Instead, it is probably better to name it something more familiar to your users. Users are typically not expected to know the terms “record” or “database” as they are used in the Palm OS paradigm. For example, it would probably be better for the “Record” menu in Date Book to be named “Event” and for it to be named “Contact” in Address Book.

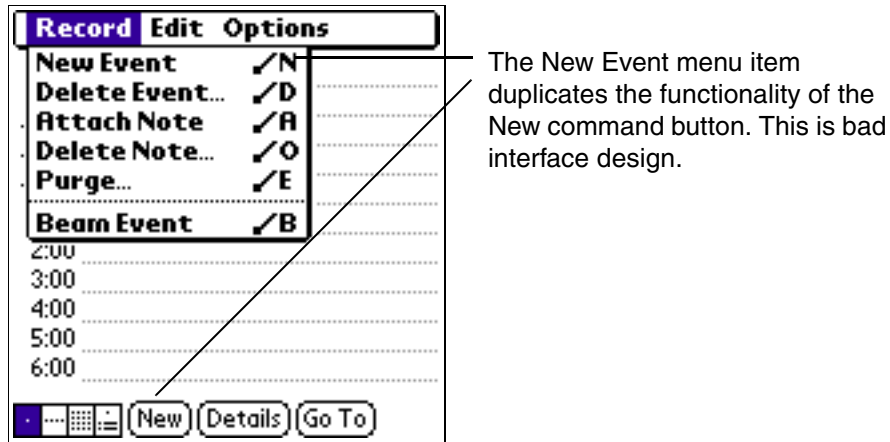
Duplicating Menu Items

You may have noticed that the Date Book application violates the rule of not duplicating a command button with a menu item (see [Figure 4.20](#)).

Executing Commands

Menus

Figure 4.20 Menu item duplication



This gives the user several ways to create a new appointment:

- Tapping the field next to a time and entering the name of the appointment
- Writing in the input area (creates an appointment at the specified time if a number is written first or an untimed appointment if a letter is entered first)
- Tapping the New button, selecting the time, and then entering the name of the appointment
- Tapping the title bar, choosing the New Event menu item from the Record menu, choosing the time, and then entering the name
- Writing the shortcut for the New Event menu item, choosing the time, and then entering the name

The first two methods of creating a new appointment are somewhat hidden interfaces. Most users catch on that they can simply start writing to create an appointment, but not all users discover this right away. Therefore, some duplication is necessary. Because novice users are the ones most likely to not understand how to enter a new appointment, the New command button (rather than a menu item) is necessary. You cannot provide a menu item and expect novice users to find it because many novice users don't know about the menu. Therefore, the New Event menu item and shortcut are simply redundancies that are probably never used.

Some application designers duplicate commands in the menu so that they can provide a shortcut for the command. This is a common practice in desktop applications; however, it is unnecessary for most users of the Palm Powered handheld. Command buttons are placed at the bottom of the form immediately above the input area. If a command has a button, it is actually more convenient and quicker to tap the button than it is to enter the menu command stroke followed by a letter. Plus, the more menu shortcuts you provide, the more you risk duplication of shortcut letters.

Do keep in mind, however, that external keyboards have a key that simulates the menu command stroke. This means that users with external keyboards will find a shortcut more convenient than a command button. Therefore, if you have room in your menu, you may want to duplicate a command if it is not one of the standard commands. The external keyboards usually have preassigned keys that simulate taps on the common command buttons, such as New, Delete, Details, OK, and Done, which makes providing a shortcut for these commands unnecessary.

Executing Commands

Menus

Presenting Options

This chapter describes user interface elements that allow the user to select discrete options or values. These elements are:

- [Check Boxes](#)
- [Pop-Up Lists](#)
- [Push Buttons](#)
- [Selector Triggers](#)
- [Sliders](#)

The first section tells you how to choose which of the elements listed above is best for your application. The rest of the chapter describes behavior and appearance guidelines for each element.

Choosing Which Element to Use

[Table 5.1](#) provides some general tips on user interface element selection. More details are provided after the table.

Table 5.1 Choosing an element to present options

Desired Action	UI Choices
Choose several from many	Check Boxes
Choose one from small number of options	Push Buttons Pop-Up Lists
Choose one from moderate number of options	Pop-Up Lists
Choose one from large number of options	Selector Triggers

Table 5.1 Choosing an element to present options (*continued*)

Desired Action	UI Choices
Choose from a range	Sliders
Toggle state	Check Boxes Can be simulated with Command Buttons if desired

Choosing Several of Many Options

Check boxes (see [Figure 5.1](#)) are the only user interface element that allow the user to select several options at the same time. The other elements allow only one of the options to be selected. Therefore, if you have several options that are not mutually exclusive, check boxes are the element to use.

Figure 5.1 Check boxes



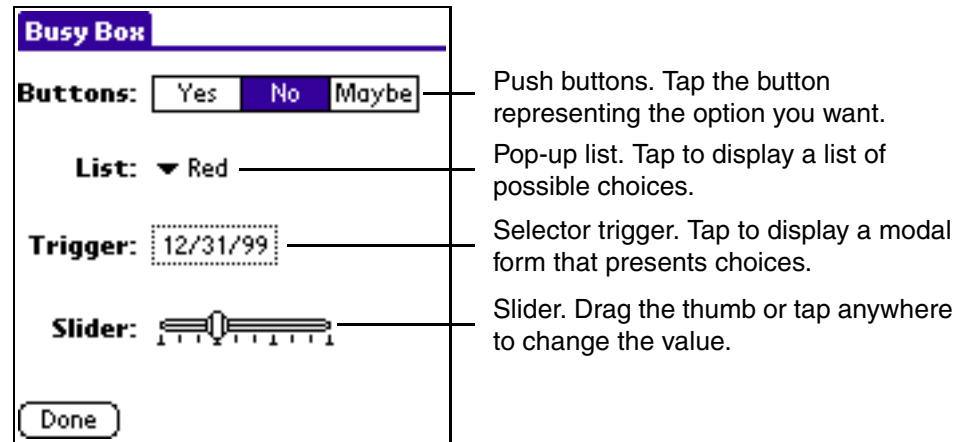
Use check boxes for options that are not mutually exclusive.

Choosing One from Many Options

If you want the user to choose only one from a set of options, there are several user interface elements you can use: push buttons, pop-up lists, selector triggers, or sliders (see [Figure 5.2](#)). Pop-up lists and sliders are very similar to their counterparts in desktop computers. Push buttons are analogous to radio buttons on a desktop computer, but they have a different look. Selector triggers are unique to Palm OS®. A **selector trigger** looks like text surrounded by a dotted

rectangle. The text displays the current selection. When the user taps the selector trigger, it displays a modal form that allows the user to change the selection.

Figure 5.2 Controls to choose one of many



The following sections discuss the pros and cons of each element.

Push Buttons

- In general, push buttons are the easiest of these elements to understand because they present all options on the screen at one time. Sliders can be equally easy to understand, but the use of sliders is limited to a few specific instances.

Push buttons are *not* easy to understand when there are only two choices (see [Figure 5.3](#)). Some users think that the highlighted (dark) button is the option that is not selected. They only realize which is the selected button when there are at least two deselected buttons to compare it with.

Figure 5.3 Two-choice push button



- In general, push buttons are the easiest element for a user to use when compared to pop-up lists and selector triggers.

Presenting Options

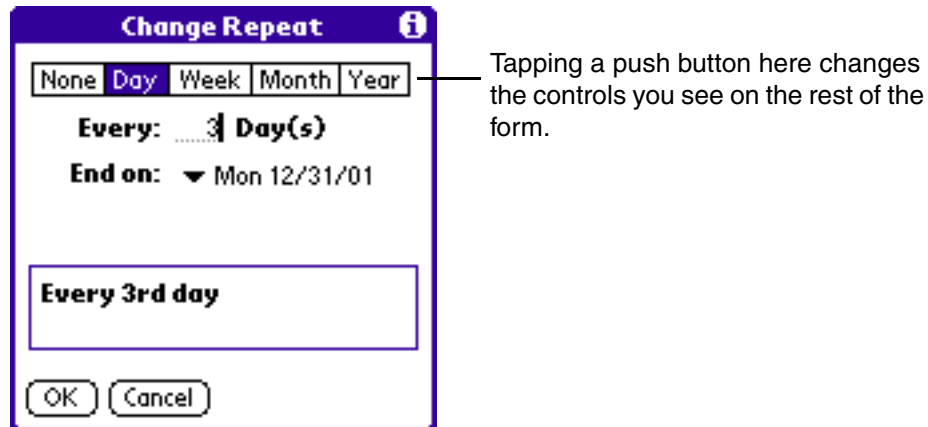
Choosing Which Element to Use

- Use push buttons if there are between three and seven options that have short names. If there are more options or the names become long, push buttons become hard to read.
- Don't use push buttons if your application is going to be localized or if there's a possibility you'll add another push button in a future version of the application.

Because of the severe space limitations, push buttons don't scale well. If you can't leave room for button growth due to localization or a new feature, then it is best to use a pop-up list.

- Use push buttons to select among application views. See [Figure 5.4](#).

Figure 5.4 Push buttons that select view



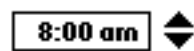
Pop-Up Lists

- Use pop-up lists when there are too many choices to display in a series of push buttons or you simply don't have room for push buttons.
- Use pop-up lists if the application is going to be localized or if there might be more options added in future versions.
- The more options a pop-up list presents, the more difficult and confusing it becomes to navigate. Try to limit yourself to one screenful of items in the pop-up list. Avoid creating pop-up lists that require the user to scroll more than one or two times. When users must scroll more than a couple of times, they tend to forget where they are in the list.

Sliders

- Use a slider to select one value from a continuous range of possible values. Sliders are best used when the setting is non-specific. For example, when setting the brightness or contrast on a screen, users do not know the specific value they want. They only know that they want the screen brighter or darker. Volume settings are also a good fit for sliders. Users do not know the numeric value of the decibel level they want; they only know that they want the sound louder or quieter.
- Sliders are only supported on Palm OS 3.5 and later. If you want to support older versions of Palm OS, do not use sliders. An alternative if you're using a slider for numeric data is to create a spin box style of interface element. Use repeating buttons to increment or decrement the numeric value. The Date Book Preferences dialog uses such an element (see [Figure 5.5](#)).

Figure 5.5 An alternative to sliders



Consider using increment and decrement arrows instead of sliders if you want to run on Palm OS 3.3 and earlier.

Selector Triggers

- Use a selector trigger and a modal form when you have so many options that a pop-up list would be too cumbersome. For example, there are well over fifty possible time zones, which would make a very large pop-up list, so the Preferences application uses a selector trigger for the time zone setting. Tapping the trigger displays a modal form containing a list of time zones. Tapping OK on the form changes the time zone shown in the selector trigger to the time zone selected in the form's list.
- Use selector triggers when you want to limit user input to a particular format. Setting a date or a time of day, for example, requires a set format.
- Use selector triggers when the data has multiple elements that must be set at the same time. A date has a month, a day, and a year. A time has the hours, minutes, and an AM or PM designation.

Presenting Options

Choosing Which Element to Use

Check Boxes

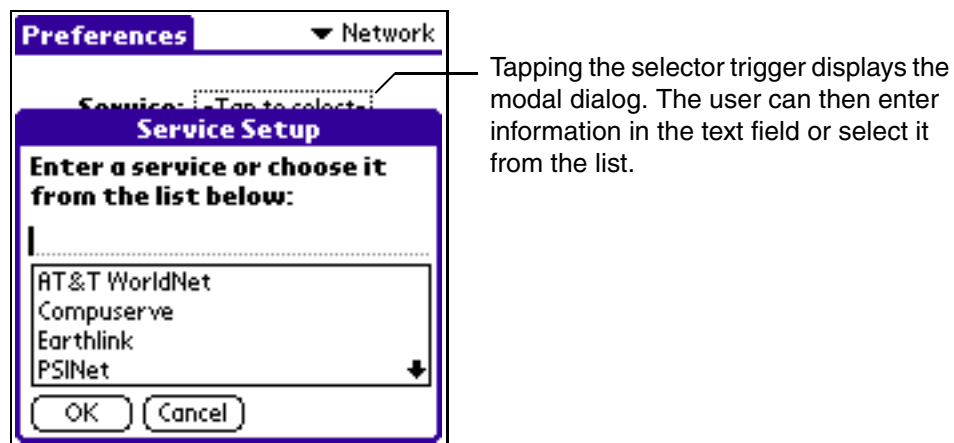
- Don't use check boxes to choose one of many items. Constructor for Palm OS allows you to create a group of check boxes that behave like push buttons, where selecting one check box deselects another, but such an interface is strongly discouraged. Users expect to be able to select more than one check box in a series.

Implementing a Combo Box

Desktop interfaces often support combo box elements, where the user can either select from an existing list of options or type in a new one. Combo boxes on the desktop often look like a pop-up list combined with a text field. Palm OS does not have a combo box element, but it supports several ways of implementing similar behavior.

It's usually best to use a selector trigger where you might use a combo box on a desktop application. Tapping the selector trigger displays a modal form that presents both the list and a text field. To fill in the text field, the user can either select from the list or write characters in the input area (see [Figure 5.6](#)).

Figure 5.6 Selector trigger as combo box



The selector trigger is a good choice when the following are true:

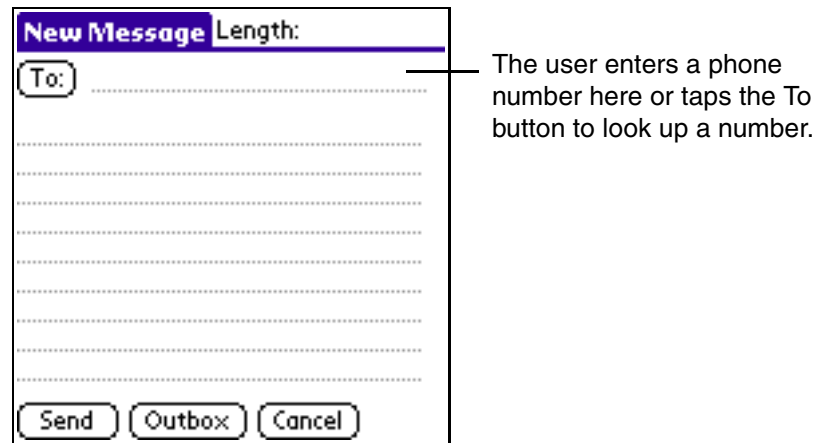
- Users do not access the form containing the selector trigger often.

- Users are more likely or just as likely to choose an option from the list of available choices than to write in a new one.

If you expect the form to be used frequently and you believe users will want to write in a new choice often, the selector trigger is not the best solution. The selector trigger implementation makes writing a new choice less convenient than a simple text field would. For example, consider implementing the expense type in the Expense application with a selector trigger and modal dialog. When users enter expenses, they often enter several in a row. If they have to tap a selector trigger, navigate a list, and tap OK to enter each expense, using the application can become tedious.

Another option is to use a command button in combination with a text field as shown in [Figure 5.7](#). This style of user interface allows the user to enter text in the text field just as easily as selecting from the list.

Figure 5.7 Command button as combo box



The command button implementation of a combo box is a good choice when the list of available options is potentially quite large. In [Figure 5.7](#), the To command button does phone lookup, and there may be hundreds of phone numbers from which to choose.

A third choice for implementing a combo box element is to use a pop-up list. Using a pop-up list element is a good idea when the following are true:

- The form is used so frequently that the extra taps required by the selector trigger make its use tedious.

Presenting Options

Check Boxes

- The available choices are far fewer than with the phone lookup example shown in [Figure 5.7](#).

To allow the user to enter new items for the list, include an Edit command in the list, in the same way that the Categories pop-up list includes an Edit Categories command. See the section “[Pop-Up Lists](#)” on page 108 for more information on implementing pop-up lists.

A pop-up list allows the user to select from the list more conveniently than a selector trigger does. Entering text is still slightly inconvenient. If you anticipate that users will want to add new items to the list often, you also may want to add a command to your interface that would display a modal dialog in which the user could add several items to the list at once.

Check Boxes

Check boxes (see [Figure 5.8](#)) display an on/off setting. Tapping a check box or its text content toggles the setting.

Use check boxes for two-state data or options, that is, for values that are either on or off, enabled or disabled.

Figure 5.8 Check box dimensions

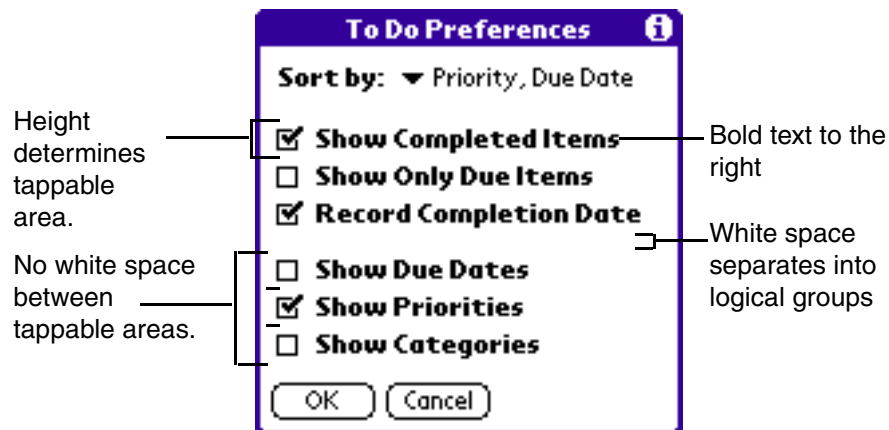


Table 5.2 Check box details

Dimension	Value
Width	System determined
Height	12 pixels minimum Height determines tappable area Height does not affect size of box or text
Top	Varies
Left	Varies
Border	None
Content	Text describing option. Can be lengthy. Use title capitalization and bold font.

Most check boxes have bold text to the right that describes the option. (This book refers to this text as the check box's **text** or **text content**.) The text should describe the "positive" state of the check box, that is, what happens when the check box is checked. The text content is part of the active area of the check box. The user can tap the text to toggle the check box.

Check boxes are not required to have text content. For example, the check boxes in each To Do List record do not.

Notice that some check boxes omit the text content to the right of the check box in favor of a label to the left. For example, the Memo Pad Details form in [Figure 5.9](#) has a checkbox that sets whether the record is private. The label for this check box is to the left of the box to make it match the only other user interface element on this form. Placing the check box on the left is done at a cost: the check box's label is no longer part of the check box's tappable area. For this reason, it's unwise to arbitrarily choose to place check box labels to the left. Do so only if there is good reason.

Presenting Options

Pop-Up Lists

Figure 5.9 A check box with a label to the left



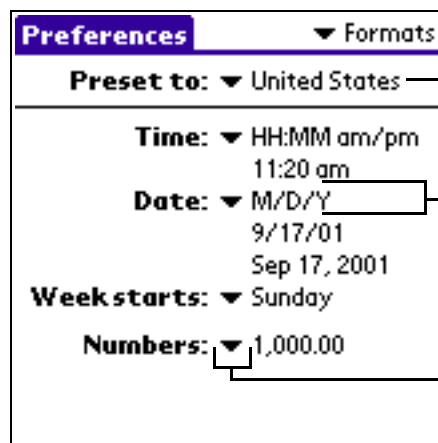
It's better to have labels for all UI elements on the same side, so this check box has no text to the right. The label on the left is not active.

Pop-Up Lists

A pop-up list allows a user to select one item from a series of items. It displays the currently selected item. A pop-up list is actually made of two user interface elements: a pop-up trigger (see [Figure 5.10](#)) and a list (see [Figure 5.11](#)).

Both push buttons and pop-up lists allow a user to select one item from a series of items. Pop-up lists are a little more difficult for users to navigate than push buttons, but they are often ideal for situations where there are several possible options from which the user can choose.

Figure 5.10 Pop-up triggers



Pop-up trigger has an arrow and shows the current selection.

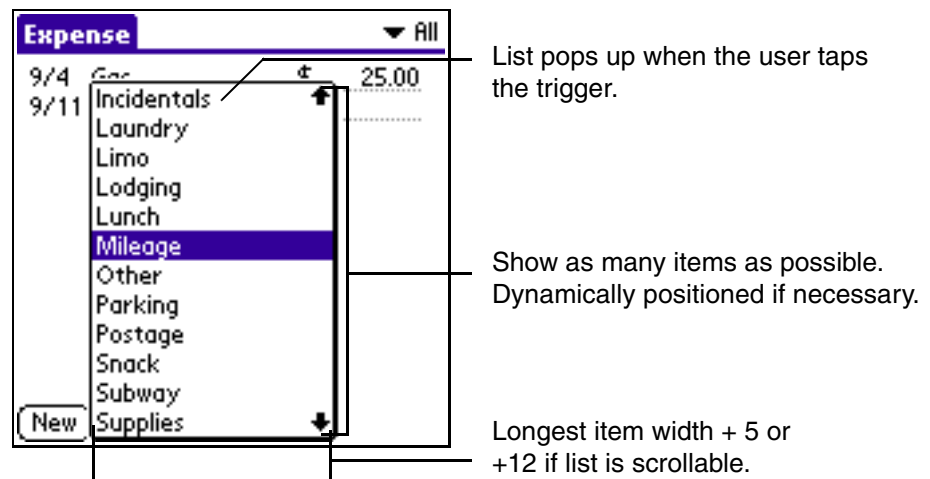
12 pixels high. Dynamic width.

Trigger takes up 15 pixels

Table 5.3 Pop-up trigger details

Dimension	Value
Width	System determined
Height	12 or <i>font height</i> + 3
Top	Varies
Left	Varies Use 160 to align trigger with right edge of screen if it resizes to the left
Border	None
Content	Current list selection

Figure 5.11 List dimensions



Presenting Options

Pop-Up Lists

Table 5.4 Pop-up list details

Dimension	Value
Width	5 + width of longest item 12 + width of longest item if list is scrollable
Height	13 items or number of items in the list if < 13 items Use 0 for the Categories list. Its height is determined at run time.
Left	<i>trigger left</i> if trigger is left-aligned <i>trigger left – list width</i> if trigger is right-aligned
Top	<i>trigger top</i> (Palm OS repositions if necessary)
Border	1 pixel square rectangle

System Supplied Behavior

The pop-up list displays the current selection. If the user taps a pop-up list's arrow or its content, the list is displayed.

Pen Interaction

Tapping a list item unhighlights the current selection and highlights the item under the pen. Dragging the pen through the list highlights the item under the pen. Dragging the pen above or below the list causes the list to scroll if it contains more choices than are visible. If the pen is otherwise dragged outside the list, the selection is unchanged.

Scrollable Pop-Up Lists

If there are more choices than can be displayed, the system draws small arrows (scroll indicators) in the right margin next to the first and last visible choice. When the pen comes down and up on a scroll indicator, the list is scrolled. When the user scrolls down, the last visible item becomes the first visible item if there are enough items to fill the list. If not, the list is scrolled so that the last item of the list appears at the bottom of the list. The reverse is true for

scrolling up. The scrolling hard keys on the handheld perform the same behavior. Scrolling doesn't change the current selection.

Look and Feel

Follow these guidelines for the behavior and appearance of a pop-up list.

Show Selection When List Pops Up

When the user taps the pop-up trigger, the list should display in such a way that the current selection is visible. If the list is large, try to adjust the scrolling so that the selection is in the center of the display. This way, the user has to do a minimum amount of scrolling to change the selection.

Present Items in a Logical Order

Position list items so that the most useful items are near the top. For example, if you are presenting a list of times of day, start the list with the most active part of the day, such as 8:00 AM. Do not start with midnight. If you enable incremental searching, as described below, present list items in alphabetical order.

Lists Do Not Execute Commands

On some desktop systems, pop-up lists can present a list of commands to the user. Such an interface is not allowed on Palm OS. Use command buttons or menus to execute commands.

Lists Are Not Hierarchical

Do not create a hierarchical pop-up list. Palm OS has no concept of a hierarchical structure for pop-up lists or menus because of the lack of screen space.

Use Incremental Search for Large Lists

Pop-up lists have an incremental search feature that the programmer can enable. This feature allows the user to select an item by writing characters matching the first few characters in that item. Particularly if your list is large, consider enabling this feature; however, it is not widely known, so you should not rely on your

Presenting Options

Pop-Up Lists

users knowing about it. Enabling this feature requires the list to be presented in alphabetical order.

First List Item Is Default Selection

The initial selection for the list by default is the first item in the list. You can display a non-selection in the trigger, if applicable. For example, the Expense application defines no default expense type. Before an expense type is selected, the words “-Expense Type-” appear in the pop-up trigger. Use dashes in the default trigger to show that the item is a pop-up list. For example, the default trigger content might be “-Select-”.

Provide a Label if Necessary

You can provide a label for the pop-up list if necessary, but doing so is not required. The content of the list trigger usually provides enough context for the user to understand the list’s purpose.

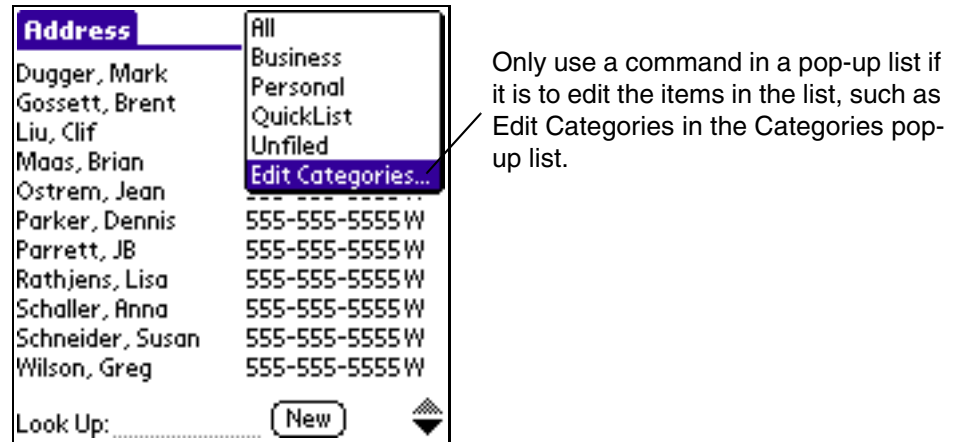
Breaking the Rules

This section points out a few of the applications that break pop-up list guidelines and tells you if doing so is appropriate.

Commands in a Pop-Up List

In general, a list should not execute commands, but this rule can be broken if you want the user to edit the list (see [Figure 5.12](#)). For example, the Categories pop-up list often includes an Edit Categories item to allow the user to delete or add categories. If you want to present a series of command choices to the user, use command buttons or a menu.

Figure 5.12 Command in a pop-up list



Pop-Up Lists with No Trigger Arrow

In some cases, you may wish to forgo drawing the pop-up trigger's arrow. The Expense application does not display a pop-up trigger for its pop-up list (see [Figure 5.13](#)), and the To Do List application does not display a trigger for its priority list. They do not use the pop-up trigger arrow for the following reasons:

- The list sets a data field in the record, as opposed to setting a display option.
- The data field is set once for each record and rarely changed.
- The trigger detracts from the display of data.
- The pop-up list is a power-user feature. Each application uses the Details dialog as an alternate, more obvious way to set the field so that new users can find it.

If you want to omit the trigger arrow because it detracts from your data display, you may do so. You must provide an obvious method of setting the field for new users.

Presenting Options

Pop-Up Lists

Figure 5.13 Pop-up trigger without arrow

The screenshot shows a window titled "Expense" with a header bar containing a dropdown menu set to "All". Below the header is a list of expenses:

Date	Description	Amount
9/4	Gas	\$ 25.00
9/11	Mileage	mi 24
9/11	-Expense type-	\$

At the bottom of the window are three buttons: "New", "Details...", and "Show...".

The Expense application does not use a trigger arrow for the expense type because it would otherwise detract from the display.

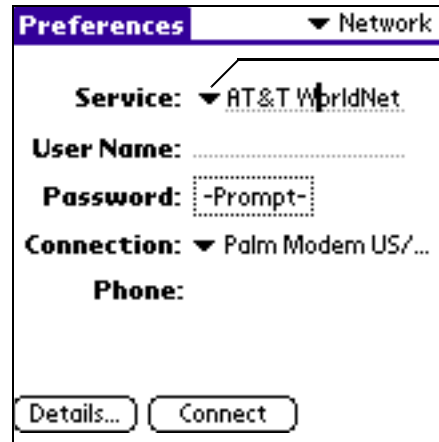
Nonstandard Pop-Up Triggers

The section "[Implementing a Combo Box](#)" on page 104 described three possible choices for implementing a combo box style element: a selector trigger with a modal form, a command button with a text field, and a pop-up list containing an Edit command.

To make it easy to add items to a pop-up list used as a combo box style element, some application designers have combined text fields with pop-up triggers that have nonstandard content. There are two competing styles of pop-up trigger used in this manner. Each style has its champions, but both have such deep design flaws that they are best avoided entirely.

Some designers use a pop-up trigger with no content (see [Figure 5.14](#)). If the user selects from the list that the trigger displays, the selection is copied into the text field appearing immediately to the right. If the user enters text in the field that does not match an item in the list, a new item is added to the list.

Figure 5.14 Pop-up list with no label



This list shows only the trigger so that it looks like a combo box. This gives the user only a very tiny target to tap to see the list.

The advantage to this implementation is that it looks like a desktop combo box and it makes it easy to add a new item to the list. However, this design has two problems:


- Only the trigger is tappable. In a standard pop-up list, the user can tap either the trigger or its contents to see the list. Many users find the trigger alone too small to tap accurately.
- The only visual difference to indicate that the combo box element's usage differs from the standard pop-up list is the use of the dotted underline in the editable text field. Some users may find this difference too subtle and may wonder why tapping the text field does not display the pop-up list.

Other application designers use the text field's label as part of the pop-up list. For example, in [Figure 5.15](#), the user can tap either the trigger to the left of the "Service" label or the "Service" label itself to select the network service displayed in the text field.

Presenting Options

Push Buttons

Figure 5.15 Pop-up list with static label

A screenshot of a 'Preferences' dialog box with a 'Network' tab. It contains several fields: 'Service' (a pop-up list showing 'AT&T WorldNet'), 'User Name' (text field with 'username@worldnet.att.net'), 'Password' (text field with '-Prompt-'), 'Connection' (a pop-up list showing 'Palm Modem US/...'), and 'Phone' (text field with '515-555-1212'). At the bottom are 'Details...' and 'Connect' buttons. A line points from the 'Service' label to the explanatory text on the right.

Tapping the Service label displays a list that modifies the text field. Users expect it to modify Service, not the field next to it.

This implementation is not a good choice because many users expect the pop-up list to behave the same way the pop-up lists in the Address Book Edit form do; they expect the list to modify the word “Service,” not the text field next to it.

Because both of these styles of combo box have usability problems, they are best avoided. For the particular form shown in [Figure 5.14](#) and [Figure 5.15](#), the selector trigger is the best fit because this form is used infrequently. If you have a form where the element is going to be used so frequently that you’d like to use a pop-up list instead, follow the recommendations in “[Implementing a Combo Box](#)” on page 104: use a standard pop-up list. Include an Edit command in the pop-up list to allow the user to add items. To make it easy for users to add several items at once, include an extra command in your interface that displays a modal form where the user can freely edit the list items.

Push Buttons

Push buttons (see [Figure 5.16](#)) are analogous to radio buttons in a desktop application. They are designed to take less space than radio buttons—they do not need room for both the round button and the label next to it.

Both push buttons and pop-up lists allow a user to select one item from a series of items. Push buttons are easier to understand because they present all options on the screen at one time. A pop-up

list hides all but the current selection. It is also easier to tap a push button than it is to navigate through a pop-up list.

Figure 5.16 Example push buttons

Priority: **1** 2 3 4 5

Automatically lock handheld:

Never
On power off
At a preset time
After a preset delay

Push buttons can be either horizontal or vertical. All buttons in the group are the same height and width. They are best used when there are between three and seven items.

Table 5.5 Push button details

Dimension	Value
Width	<i>content width + 2</i> minimum All push buttons in group should have same width
Height	12 All push button in group should have same height
Top	Varies All push buttons in group should have same top (if horizontal row)
First Button Left	Varies
Button(<i>n</i>) Left	<i>button(n-1) left + button(n-1) width + 1</i>
Border	1 pixel square rectangle
Content	Text label or graphic. As small as possible.
Label	Provide label if purpose of buttons not immediately apparent

System Supplied Behavior

As long as you assign all of the push buttons the same group ID, the system ensures that the user can select only one button. If the user taps one button in the group, that button is highlighted, and the

Presenting Options

Push Buttons

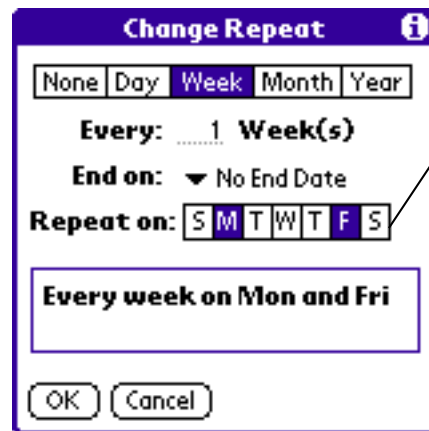
previously selected button is unhighlighted. Unlike a command button, a push button is not unhighlighted when the user releases the pen; a push button remains highlighted until another button in its group is selected.

Breaking the Rules

It's possible to create push buttons that aren't mutually exclusive by leaving the group ID unassigned. In fact, it's a common programming mistake. Because it is such a common mistake, such an interface is likely to look more like a bug than a feature to your users.

The Week view in Date Book's Change Repeat dialog uses non-exclusive push buttons to present the days of the week (see [Figure 5.17](#)). It's easy to see why this user interface element was chosen: it takes up far less screen space than a list of check boxes would. However, users expect items that look like push buttons to behave like push buttons. Many users will look at the dialog shown in [Figure 5.17](#) with no days selected and believe that they cannot choose more than one day.

Figure 5.17 Non-exclusive push buttons



You can select more than one from this set of push buttons. Avoid creating this type of interface.

Selector Triggers

A selector trigger (see [Figure 5.18](#)) provides a value for a single field or option. Tapping it displays a modal form in which the user can change the value. When the form is dismissed, the selector trigger updates to show the newly selected value.

Figure 5.18 Selector triggers

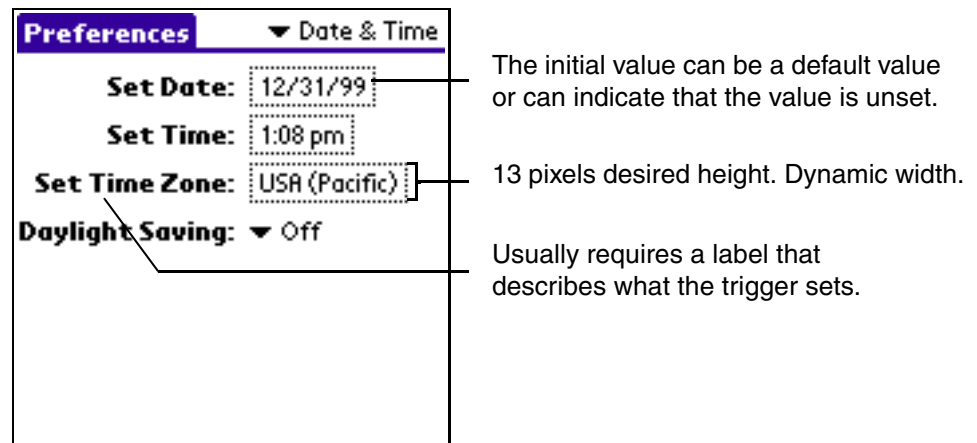


Table 5.6 Selector trigger details

Dimension	Value
Width	System determined
Height	13 pixels or odd number so that the corner pixels are black
Top	Varies
Left	Varies
Border	1 pixel square dotted rectangle
Content	Text providing current value
Label	Usually requires separate bold label to describe the value

System Supplied Behavior

The user taps the trigger once with the pen. When the user does so, Palm OS highlights the trigger momentarily while it is being pressed. Palm OS ensures that the trigger is not selected if the user drags the pen outside of the bounds of the trigger before releasing it.

Look and Feel

Follow these guidelines for the behavior and appearance of selector triggers.

Selector Triggers Display a Modal Form

Your application must ensure that the selector trigger displays a modal form and that upon return, the trigger's content changes to the value set using that modal form.

Do not use a selector trigger as a toggle button that toggles between states. Depending on the situation, a command button or a check box is more appropriate for this purpose.

Include a Label if Appropriate

It's usually appropriate to include a bold label that indicates what value the selector trigger sets. This label can be omitted if the selector trigger's placement or its setting make the trigger's purpose clear. The selector trigger in the Address Edit form's title bar is not labeled, for example, because its position and setting make it clear that it displays the category.

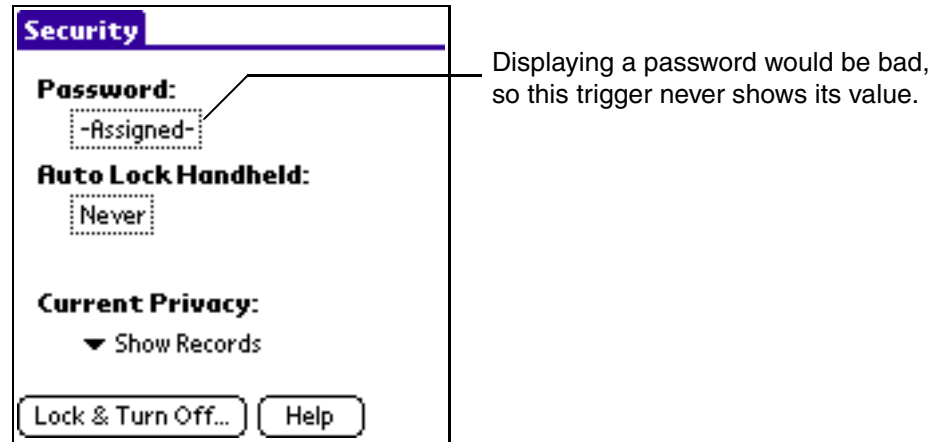
Breaking the Rules

This section points out a few of the applications that break selector trigger guidelines and tells you if doing so is appropriate.

Selector Triggers That Don't Update

The only triggers that are allowed not to update when the user has selected a value are triggers for password fields (see [Figure 5.19](#)). Password fields update from "Unassigned" to "Assigned" when the password is set the first time. From that point on, the selector trigger remains "Assigned" when the user changes the password.

Figure 5.19 Selector trigger for passwords



All other selector triggers must update to show the current selection or give some indication that the selection has changed. You can consider the password selector trigger's changing from "Unassigned" to "Assigned" as updating to show the current value at a much lower resolution than a normal selector trigger would. This is acceptable if you want the selector trigger to change some value that should remain hidden from view. If you want to create a trigger that doesn't update at all, consider using a command button instead.

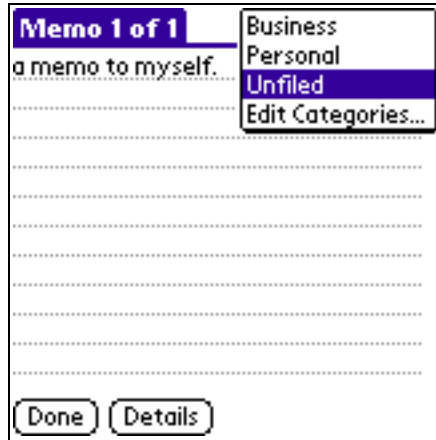
Selector Triggers with No Modal Form

Some selector triggers do not display a modal form when tapped. The Address Edit and Memo Edit forms each use a selector trigger to display that record's category (see [Figure 5.20](#)). If you tap the selector trigger, the Category list is displayed rather than a modal form.

Presenting Options

Selector Triggers

Figure 5.20 Selector trigger displaying a list



The selector trigger is used on the Memo Pad Edit form to differentiate its category control from the one on Memo Pad's main form because they two serve different purposes.

These two forms use a selector trigger for the category rather than a pop-up list to convey that the control performs a different function than the Category pop-up list the user sees on other forms. On other forms, the Category pop-up list simply filters the display of records. In contrast, the Category selector trigger on these forms changes the category in which the current record is filed, but does not change which record is displayed.

Why doesn't this Category selector trigger display a modal form that changes the record's category? Doing so would allow it to conform to the selector trigger interface guidelines, but it would increase the number of taps for this task from two taps to four. The application designers believe that minimizing the number of taps outweighs conforming to the guidelines in this instance. Keep in mind that novice users are more likely to set the record's category using the Details dialog.

That being said, the Category selector trigger on these two forms is widely considered to be an imperfect solution at best. Some users expect this control to be placed where it is and immediately understand its purpose. Other users confuse it with the Category pop-up list that controls the display on the main form.

Sliders

A slider (see [Figure 5.21](#)) represents a value that falls within a particular range. For example, a slider might represent a value that can be between zero and ten.

Figure 5.21 Slider

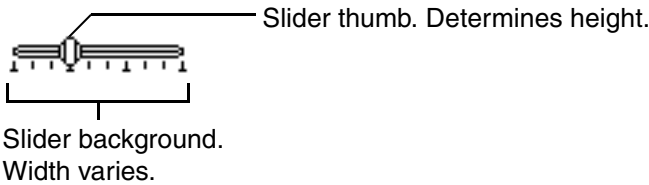


Table 5.7 Slider details

Dimension	Value
Width	Varies The default background bitmap looks best at these widths: 72, 93, 114, 135, or 156
Height	15 or height of thumb bitmap
Top	Varies
Left	Varies
Border	None
Content	N/A

System Supplied Behavior

The slider has four attributes that determine its behavior. You set these when you create the slider:

- The minimum value the slider can represent
- The maximum value the slider can represent
- The initial value

Presenting Options

Sliders

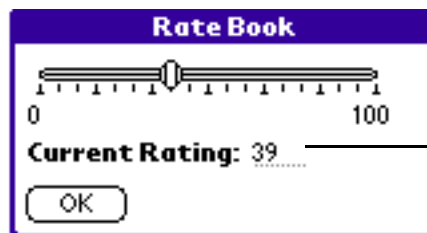
- The page jump value, or the amount by which the value is increased or decreased when the user clicks to the left or right of the slider thumb

Palm OS updates both the slider appearance and slider value. When the user drags the thumb, the slider value updates accordingly. When the user taps the slider background to the left of the thumb, the slider value decreases by the page jump value and the thumb moves to the left the same amount. When the user taps to the right of the thumb, the slider value increases by the page jump value and the thumb moves to the right.

Palm OS supports two types of sliders: regular and feedback sliders. The difference between the two is entirely behavioral. If the user drags the thumb or holds down the pen within the slider bounds, the feedback slider continually updates the slider value and sends events as this happens. The regular slider waits until the user lifts the pen and then updates the slider value.

In most cases, a regular slider suffices. You only need a feedback slider if you want to provide further visual feedback to the user apart from the slider thumb moving. For example, if you have a field that shows the current value of the slider, you probably want to use a feedback slider (see [Figure 5.22](#)).

Figure 5.22 Feedback slider



If you want this field to change while the user drags the thumb, use a feedback slider.

Look and Feel

Sliders are drawn using two bitmaps: one for the slider background, and the other for the thumb. If you use the default bitmaps, you should make the slider 15 pixels tall to allow room for the thumb, and you can make it as wide as you want.

The default bitmaps work well if the slider sets a numeric value. If your slider sets a more visual value, use a different background

bitmap to show the user what dragging the thumb does. The brightness and contrast adjust forms each use a non-default bitmap (see [Figure 5.23](#)). If possible, make the bitmap as wide as it will display on the screen.

Figure 5.23 Non-default slider bitmap



This slider uses a different background bitmap to show that moving the thumb to the right makes the screen brighter.

Presenting Options

Sliders

Displaying Data

This chapter describes user interface elements that display data. They often comprise the bulk of a user interface. These elements are:

- [Fields](#)
- [Lists](#)
- [Tables](#)

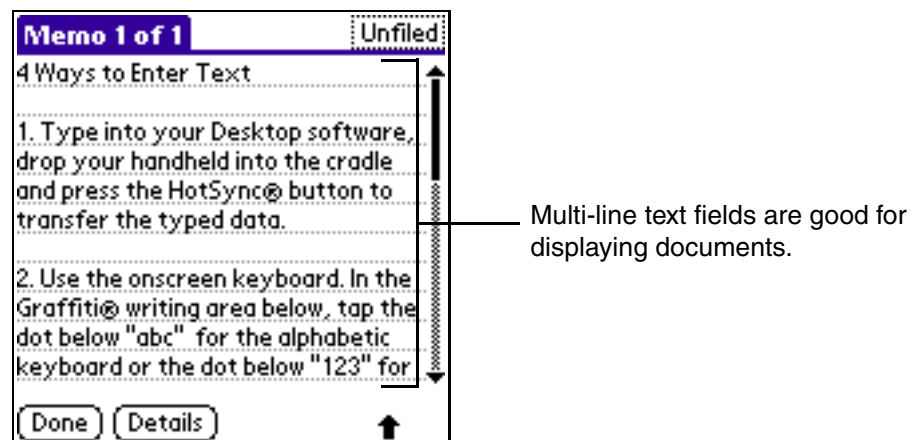
The first section tells you how to choose which of the elements listed above is best for your application. The rest of the chapter describes behavior and appearance guidelines for each element.

Choosing Which Element to Use

To display data, you can use fields, tables, or lists. Fields are good for allowing free-form entry of text and for displaying large documents. Tables and lists display columnar lists of data.

Text fields can display long documents or a single line of text. The Memo Pad Edit form shown in [Figure 6.1](#) is a good example of using a field to display and edit a document.

Figure 6.1 Multi-line text field

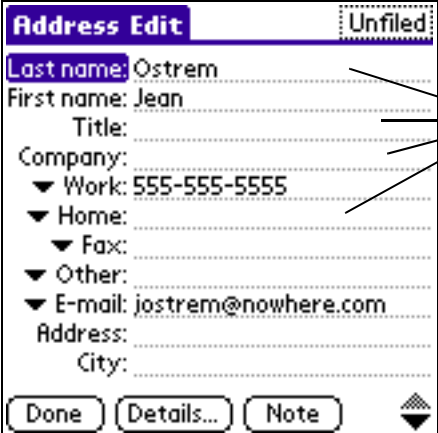


Displaying Data

Choosing Which Element to Use

The Address Book Edit form uses single-line fields for each part of a contact's information (see [Figure 6.2](#)).


Figure 6.2 Single line text fields



The screenshot shows a window titled "Address Edit" with a status bar "Unfiled". The form contains several single-line text fields for contact information: "Last name: Ostrem", "First name: Jean", "Title:", "Company:", "Work: 555-555-5555", "Home:", "Fax:", "Other:", "E-mail: jostrem@nowhere.com", "Address:", and "City:". At the bottom are buttons for "Done", "Details...", and "Note", along with a small icon. An arrow points from the text "Use single-line fields for simple data entry." to the form fields.

Use a table to display a list of items if those items represent the primary data of your application. Tables can show either a single column of text or multiple columns containing text and other user interface elements (see [Figure 6.3](#)).

Figure 6.3 Table



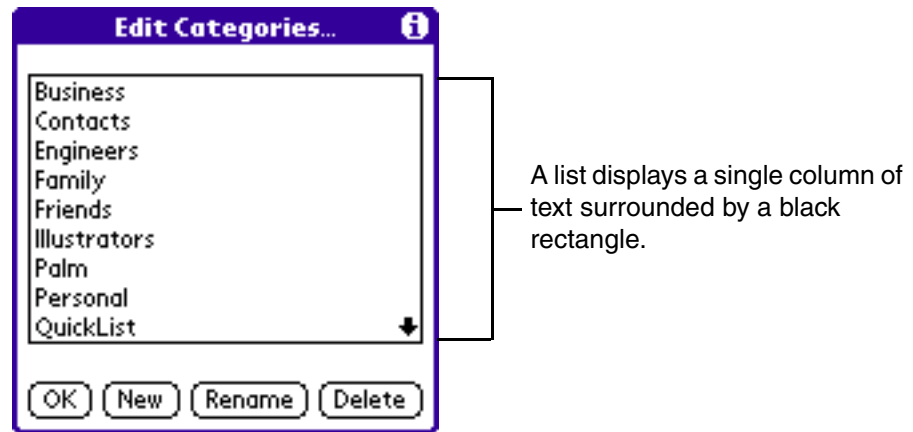
The screenshot shows a window titled "Books" with a status bar "Unfiled". It displays a table with two columns: book titles and authors. The table contains the following data:

Book Title	Author
A Way in the World	Naipaul
Captain and the Enemy	Greene
Deadeye Dick	Vonnegut
Dont Stand Too Close	Allen
Map of the World	Hamilton
Nervous Conditions	Dangaremb...
Rice Room	Fong-Torres
Sleeping at the Starlig...	White
Smilla's Sense of Snow	Hoeg
Stone Diaries	Shields

At the bottom, there is a "Look Up:" field, a "New" button, and a small icon. An arrow points from the text "Tables can display multiple columns of text or other elements and can allow direct editing of text." to the table.

A list user interface element is a box that generally displays a textual single-column list (see [Figure 6.4](#)) although it can be modified to display multiple columns or graphics. The user cannot directly edit the items presented in the list.

Figure 6.4 List



Use a list for information that is secondary in nature and that is primarily static. Some examples of places where a list element is appropriate are:

- The Edit Categories dialog uses a list to display categories because they are a secondary form of information.
- An application that shows the time of day in various cities could show the possible cities in a list element because the primary data of the application are the times of day.
- A document reader application could show the titles of available documents in a list element because the primary data are the contents of the documents themselves, not the list of titles.

For more information on choosing between a table and a list, see [“Choosing Lists Over Tables”](#) on page 140.

Fields

Fields (see [Figure 6.5](#)) can display one or more lines of text. Fields can be either editable or non-editable.

Figure 6.5 Multi-line text field

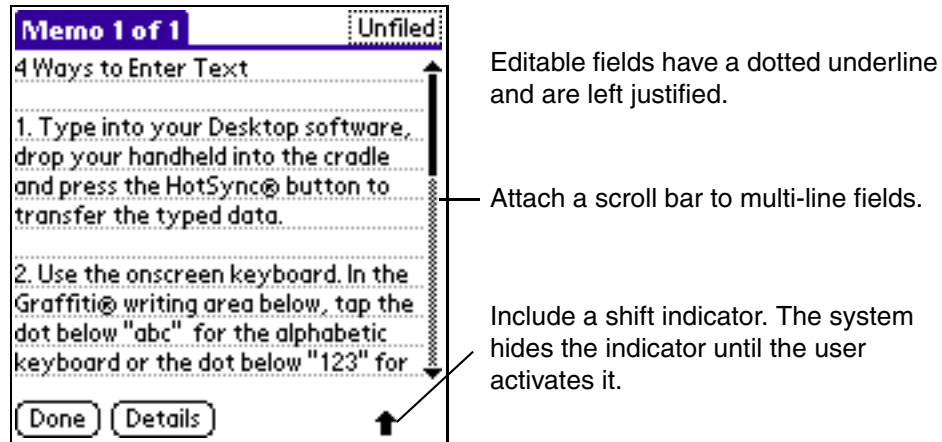


Table 6.1 Field details

Dimension	Value
Width	Varies Make as wide as possible
Height	11 or <i>font height</i> + 2 (single line) Make multi-line fields as tall as possible. Use a multiple of the line height (<i>font height</i> + 2).
Top	Varies
Left	Varies
Border	None. Dotted underline denotes editable field.
Label	Usually requires label to tell user what to write in the field

System Supplied Behavior

Palm OS® provides the following support for fields:

- Insertion point positioning. When the user taps in an editable text field, Palm OS displays a blinking cursor at that location.
- Text selection. The user can drag the stylus to select a range of text. On Palm OS 3.5 and later, the user can also double-

tap a word to select that word or triple-tap to select the entire line.

If the field is scrollable, the user can drag the stylus past the edge of the field, and the system scrolls the field in direction indicated.

- Display of text using a single font. You cannot, for example, display three words of text and have the second word be bold.
- Word wrapping for multi-line text fields.
- Maximum character limit. You can set a maximum number of characters for an editable text field. If the user enters more than the maximum number of characters, the field stops accepting input.

NOTE: The maximum characters attribute is actually the maximum number of bytes that the user can enter. In some encodings, characters are larger than one byte. Consider this when setting your field's maximum characters.

Palm OS does *not* support overstrike input mode (that is, overwriting text to the right of the cursor), horizontal scrolling, or numeric formatting for text fields.

Look and Feel

Follow these guidelines for the behavior and appearance of fields.

Editable Fields Have Dotted Underline, Are Left Justified

Fields that are editable should use a dotted underline to indicate that the field can be edited. Do not use a solid underline.

Virtually all editable fields should left-justify the text contained in them. Constructor for Palm OS allows right-justified fields, but you should only use right justification for numeric fields or for non-editable text fields that you use as labels.

You can place an editable field anywhere on a form except in the title bar area.

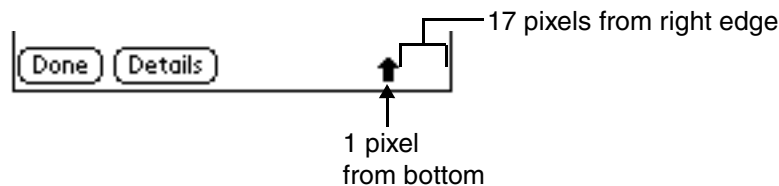
Edit Menu Required

Any form containing an editable text field must contain an Edit menu. See the section “[Edit Menu Required for Text Fields](#)” on page 89 for more information.

Shift Indicator Required

Any form containing an editable text field must contain a shift indicator. Most modeless forms have the shift indicator positioned as shown in [Figure 6.6](#).

Figure 6.6 Shift indicator placement (modeless form)



This placement leaves space for the scroll buttons if the form uses them. Even modeless forms that use a scroll bar instead of scroll buttons place the shift indicator at this location. If you have so many controls at the bottom of the form that the shift indicator cannot be placed 17 pixels in from the right, leave 4 pixels of white space between the shift indicator and the control to its left.

On modal forms, place the shift indicator as shown in [Figure 6.7](#). It's rare for modal forms to have both a shift indicator and scroll buttons, but if yours does, place the shift indicator 4 pixels to the left of the scroll buttons.

Figure 6.7 Shift indicator placement (modal form)

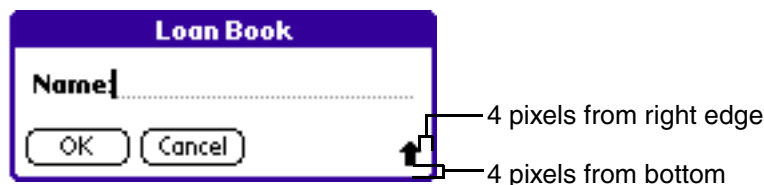


Table 6.2 Shift indicator details

Dimension	Value
Width	9 pixels
Height	9 pixels
Left	135 (modeless form) 142 (modal form)
Top	150 (modeless form) <i>form height</i> – 13 (modal form)

If you place a shift indicator on your form, Palm OS correctly displays all of the Graffiti® or Graffiti 2 shift states (punctuation, symbol, uppercase shift, and uppercase lock). For Japanese systems, the shift indicator also indicates whether the front end processor is on or off and whether it converts to Hiragana or Katakana characters.

Enable Auto-Shifting for Most Text Fields

Turn on the auto-shift attribute for all editable text fields unless the field should not accept any uppercase letters. The auto-shift attribute is a convenience for your users that automatically capitalizes the first word in the field and the first word after a sentence. Do not enable auto-shifting if the field contains information that is not routinely capitalized, such as an email address.

Note that the auto-shifting rules are language-specific, since capitalization differs depending on the region. These rules depend on the Palm OS version, the market into which the device is being sold, and so on.

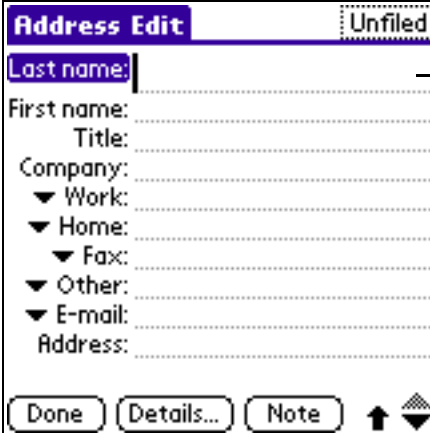
Set the Focus When the Form Is Displayed

When a form containing a text field is displayed, set the focus in the text field so that it shows the blinking cursor. Users can then start writing immediately without having to tap the text field first. If the form has more than one text field, set the focus in the first field. For

Displaying Data Fields

example, the Address Book Edit form places the focus in the Last Name field (see [Figure 6.8](#)).

Figure 6.8 Field with focus

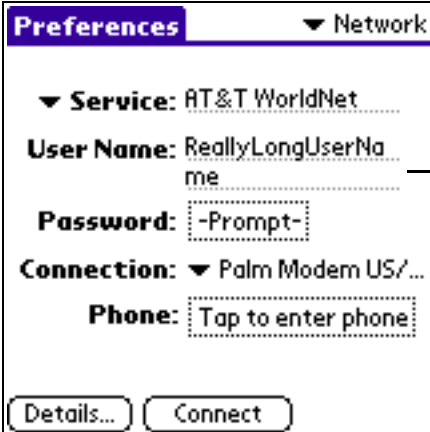


The first field has the focus when the form opens.

Decide if Single-Line Fields Should Grow

Wherever possible, you should make the text field wide enough to display its maximum number of characters. If it's not possible, consider growing the field as needed. For example, the User Name field in the Network Preferences panel grows if the user enters a very long name (see [Figure 6.9](#)). All other controls on the form are adjusted downward.

Figure 6.9 Field that resizes as text grows



The User Name field adds a second line if necessary.

If you absolutely cannot allow the user to enter more than a single line of information, turn on the single line attribute. If this attribute is not set, the user can accidentally enter a carriage return in the field and keep writing. If the field does not resize with the carriage return, the user may not know that they've entered extra information in the field.

Add a Scroll Bar to Multi-Line Text Fields

If you have a multi-line text field that accepts a large number of characters, add a scroll bar to support scrolling the text field. See [Chapter 7, "Scrolling,"](#) on page 147 for guidelines on scrolling.



Limit the Amount of Required Data Entry

When designing your form, remember that most users only have the Graffiti or Graffiti 2 power writing software and onscreen keyboard available to enter data. For this reason, you should try to limit the amount of data you require users to enter in a text field. Wherever possible, consider helping the user by providing pop-up lists instead of text fields, by auto-completing text the user enters in the field, and so on.

Support Graffiti Navigation and Tabbing between Fields

Support the Graffiti navigation characters on forms containing several text fields that the user might fill out in succession. Although most users do not take the time to learn the Graffiti navigation characters and handhelds that use Graffiti 2 handwriting recognition do not support them, power users know these strokes and appreciate interfaces that allow them. See [Table 6.3](#) for a list of the Graffiti navigation characters.



Table 6.3 Graffiti navigation keystrokes

Action	Stroke
Next field	
Previous field	

Displaying Data

Fields

Table 6.3 Graffiti navigation keystrokes (*continued*)

Action	Stroke
Move cursor left one character	
Move cursor right one character	

Also consider that users may fill out your form using an external or built-in keyboard. These users expect that pressing the tab key will move the cursor to the next field and that pressing the shift and tab keys will move to the previous field. Support this functionality where possible.

Validate After Text Is Entered

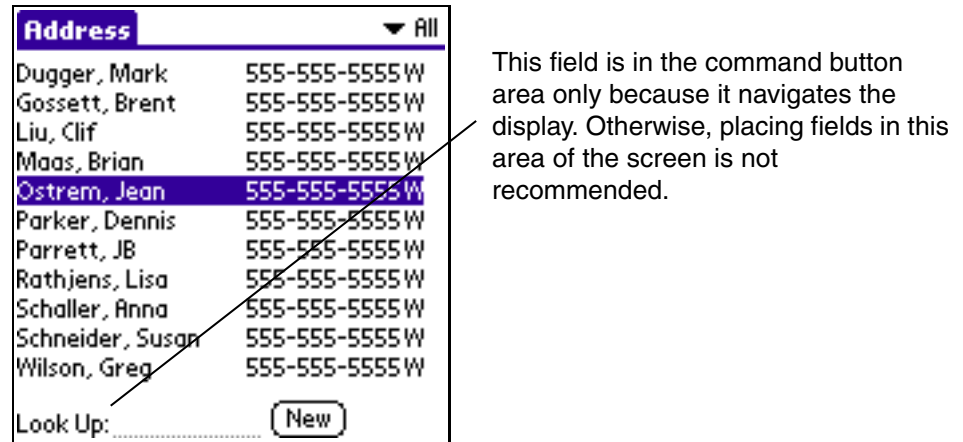
If your application needs to perform data validation on text entered into a field, provide a Done command button and validate all text fields when the user taps the button. It's tempting to perform validation as soon as the text is entered; however, doing so is a poor design decision for Palm OS.

Suppose the user receives a phone call in the middle of entering data and needs to schedule an appointment with the caller. If the user has made a mistake entering data in the current field and you do not allow the user to exit your form until the mistake is corrected, the user could become extremely frustrated. It's better to preserve the data as the user has entered it and allow the user to exit immediately. When your application is relaunched, display the data entry form with the partially entered data. Validate only when the Done button is tapped.

Breaking the Rules

Text fields normally do not belong in the command button area, but you can place a field in this area if it allows the user to navigate the view. The Address Book application's main form uses such a text field (see [Figure 6.10](#)).

Figure 6.10 Text field in command button area

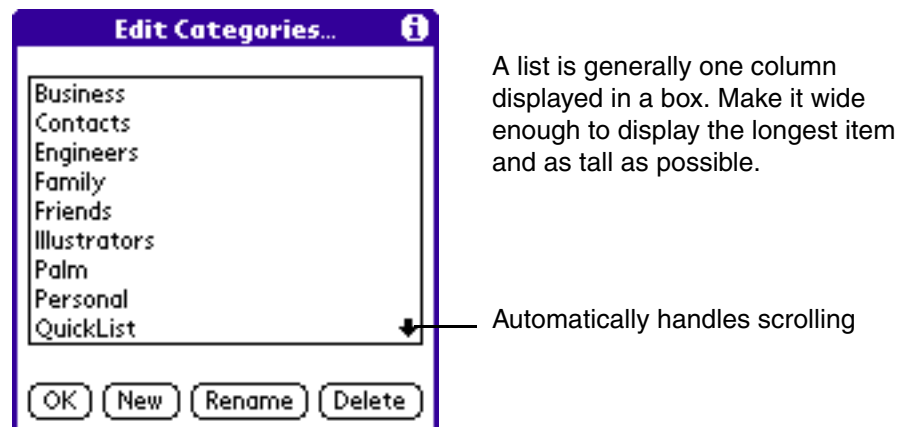


Lists

A list (see [Figure 6.11](#)) provides a box with a list of choices to the user. The list is scrollable if the choices don't all fit in the box.

In Palm OS, a list element associated with a pop-up trigger is called a pop-up list. This section describes lists that don't pop up. See "[Pop-Up Lists](#)" on page 108 for guidelines for pop-up lists.

Figure 6.11 List



Displaying Data

Lists

Table 6.4 List details

Dimension	Value
Width	Width of longest item + 5 or width of longest item + 12 to allow scrolling
Height	11 items tall for a full screen list (modeless form) 10 items tall for a full screen list (modal form)
Left	Varies
Top	18 for a full screen list
Border	1 pixel solid rectangle
Label	Usually not necessary

System Supplied Behavior

Palm OS supplies the following behavior for every list.

Pen Interaction

Tapping a list item unhighlights the current selection and highlights the item under the pen. Dragging the pen through the list highlights the item under the pen. Dragging the pen above or below the list causes the list to scroll if it contains more choices than are visible. If the pen is otherwise dragged outside the list, the selection is unchanged.

Scrollable Lists

If there are more choices than can be displayed, the system draws small arrows (scroll indicators) in the right margin next to the first and last visible choice. When the pen comes down and up on a scroll indicator, the list is scrolled. When the user scrolls down, the last visible item becomes the first visible item if there are enough items to fill the list. If not, the list is scrolled so that the last item of the list appears at the bottom of the list. The reverse is true for scrolling up. The scrolling hard keys on the handheld perform the same behavior. Scrolling doesn't change the current selection.

Look and Feel

Follow these guidelines for the behavior and appearance of lists.

Always Show Current Selection

The list should always show the current selection when the form containing it is first opened. Suppose the user opens a form containing a list with twenty items, scrolls the list to the bottom, selects item fifteen, and dismisses the form. The next time the user opens the form, the list should still be scrolled to the bottom and item fifteen should be selected.

Present Items in a Logical Order

Position list items so that the most useful items are near the top. For example, if you are presenting a list of times of day, start the list with the most active part of the day, such as 8:00 AM. Do not start with midnight.

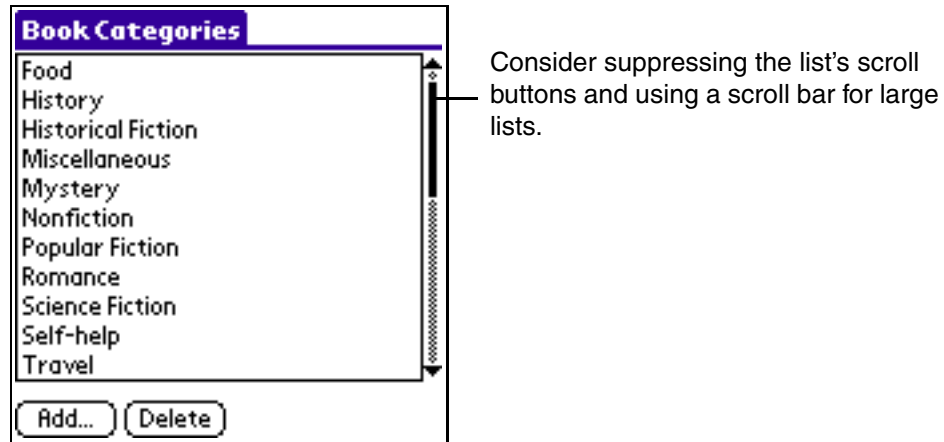
Use a Scroll Bar for Large Lists

Although lists come with their own scroll indicators and handle scrolling automatically, you can suppress the display of these indicators programmatically and attach a scroll bar to the list instead (see [Figure 6.12](#)). You should consider doing so if the list is the main element on your form and it contains a large number of items. The scroll bar is preferred for large lists because the scroll bar provides an indicator of location. See [Chapter 7](#), “[Scrolling](#),” on page 147 for more information.

Displaying Data

Lists

Figure 6.12 List with scroll bar



Breaking the Rules

This section points out a few applications that break the list guidelines and tells you if doing so was appropriate.

Choosing Lists Over Tables

Lists and tables are not interchangeable. A list always displays a box around its contents. This box conveys that the information inside of it is secondary to some other type of data in the application.

Many application developers shy away from using tables because of the degree of programming difficulty involved. Instead, they use a list element because lists are easier to work with programmatically (see [Figure 6.13](#)).

Figure 6.13 Improper use of a list



This form must use a table instead of a list because the primary purpose of this application is to display a list of book titles.

The form shown in [Figure 6.13](#) is an early design of the main form of the Books application described in [Chapter 1, “Palm OS Application Design.”](#) This application’s primary purpose to catalog the books that a person owns. The book titles displayed in the form are thus the main data of the application and should be displayed in a table.

There are ways to programmatically suppress the drawing of the box around the list. As long as your application suppresses the list’s border and scroll buttons so that it looks and feels exactly like a table element would, then you may use a list element instead. Note that you can use lists to simulate multi-column tables as well as single-column tables.

Tall Lists

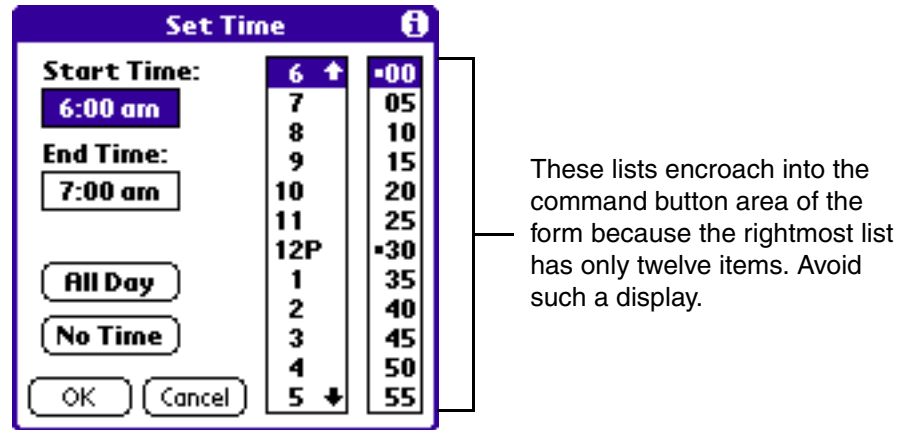
If you use as much of the data area of a form as possible to display a list, the list can show eleven items maximum. This can be problematic if you have a list with twelve items. You hate to make your users scroll just to reach the last item in the list.

The Set Time dialog shown in [Figure 6.14](#) has this problem. The minutes list has exactly twelve items. To show all twelve items, the list has encroached on the command button area of the form. The list containing the hours was resized to match. To make room for these lists, two command buttons were moved from this area to the left side of the form.

Displaying Data

Tables

Figure 6.14 List in command button area



Another possible approach to this problem would have been to have each of the two lists show only eight items because it's likely that a user is scheduling appointments during an eight-hour work day. This would have made room for all four buttons at the bottom of the form.

The designers of this form weighed the two possible approaches and decided that it is easier for the user if he or she can see all possible choices for the minutes field at once. This usability factor outweighed any other considerations on control placement guidelines.

You'll probably not run into a similar situation when designing your interface. If you do, avoid mimicking this interface without clearly considering the trade offs involved. Grow the list into the command button area only if you believe the possible user benefits of doing so clearly outweigh any other design considerations.

Tables

Tables (see [Figure 6.15](#)) are a convenient way to display a set of data and organize that data into one or more columns.

Figure 6.15 Table



Address		▼ All
Dugger, Mark	555-555-5555 W	
Gossett, Brent	555-555-5555 W	
Liu, Clif	555-555-5555 W	
Maas, Brian	555-555-5555 W	
Ostrem, Jean	555-555-5555 W	
Parker, Dennis	555-555-5555 W	
Parrett, JB	555-555-5555 W	
Rathjens, Lisa	555-555-5555 W	
Schaller, Anna	555-555-5555 W	
Schneider, Susan	555-555-5555 W	
Wilson, Greg	555-555-5555 W	

Look Up:

Address Book uses a two column table for its main view.

Table 6.5 Table details

Dimension	Value
Width	160 (modeless form) 148 (modal form)
Height	120 pixels (modeless form) 108 pixels (modal form)
Row height	12 pixels (single-line rows)
Top	18
Left	0 for modeless form 4 for modal form
Border	None.

System Supplied Behavior

Tables are highly configurable objects. The system supplied behavior depends heavily on the type of table you create. Palm OS supports the use of certain user interface elements in the table, namely editable text fields, check boxes, and pop-up lists. Palm OS ensures that these elements work exactly as they do when they appear outside of a table.

Displaying Data

Tables

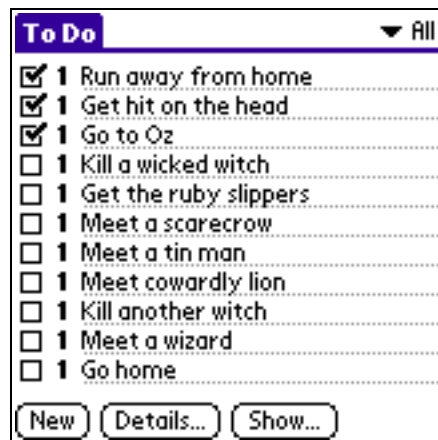
If the table displays non-editable text, Palm OS ensures that the text is highlighted when the pen is down in the bounds of the table item and unhighlighted when the pen is released. Note that only a single item is selected. Palm OS does not support the selection of an entire table row.

Look and Feel

Tables do not have borders or column headings because the short-term clarity is not worth the long-term loss of space. From a user interface standpoint, there are two types of tables: tables that allow direct editing and tables that don't.

The To Do List main form (see [Figure 6.16](#)) is an example of a table that allows direct editing. To create a new item in the list, you tap the New command button. From there, you can directly enter the text of the task, assign its priority and due date, and check the box when the task is done.

Figure 6.16 Table with direct editing



The To Do List is a table where the user can edit the table rows directly.

The Address Book list view (see [Figure 6.17](#)) is an example of a table that doesn't allow direct editing. In such a table, you must decide what happens when the user taps an item. In most cases, the application should display a form that edits the data in that item.

Figure 6.17 Table without direct editing

Address ▼ All	
Dugger, Mark	555-555-5555 W
Gossett, Brent	555-555-5555 W
Liu, Clif	555-555-5555 W
Maas, Brian	555-555-5555 W
Ostrem, Jean	555-555-5555 W
Parker, Dennis	555-555-5555 W
Parrett, JB	555-555-5555 W
Rathjens, Lisa	555-555-5555 W
Schaller, Anna	555-555-5555 W
Schneider, Susan	555-555-5555 W
Wilson, Greg	555-555-5555 W
Look Up: <input type="text"/> <input type="button" value="New"/>	

In the Address Book, you cannot edit the table data directly. Instead, tapping a table item takes you to another form.

As stated previously, Palm OS only allows selection of a single table item at a time. Your application must be designed so that each item selection has a different and unique purpose.

Consider the table in the Address Book list view. This table has two columns, but they might not be the columns you think they are. The first column displays both the contact name and phone number or email address. The second column is a very narrow column that displays the note icon if a contact has a note attached to it. Tapping an item in the first column displays more information about the selected contact. Tapping an item in the second column (even if the item is blank) displays the Note form, in which users can enter a free-form note about the contact.

If you have a similar display, you'll probably be tempted to show the contact name in one column, the phone number in a second column, and the note icon in a third column. Such an interface is appropriate only if tapping the phone number performs a different function than tapping the contact name. If tapping either the name or the phone number takes the user to the same form, both the name and the number should highlight at the same time.

Displaying Data

Tables

Scrolling

Palm OS® provides two controls that can scroll other controls:

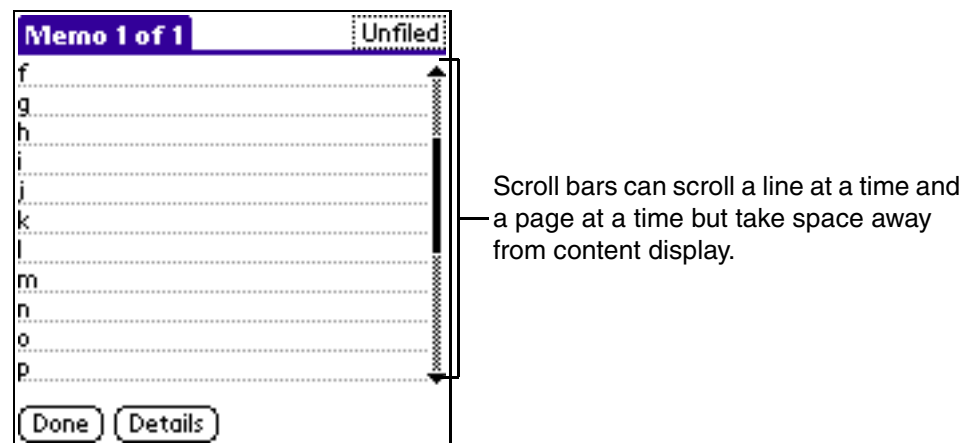
- [Scroll Bars](#)
- [Scroll Buttons](#)

The first section tells you how to choose which of the elements listed above is best for your application. The rest of the chapter describes behavior and appearance guidelines for each element.

Choosing between Scroll Bars and Scroll Buttons

To scroll form content, such as tables, lists, fields, or bitmaps, you have your choice of scroll bars or scroll buttons. Scroll bars appear on the right side of the screen (see [Figure 7.1](#)).

Figure 7.1 Scroll bars



Scroll buttons are arrows that appear at the bottom of the form (see [Figure 7.2](#)).

Scrolling

Choosing between Scroll Bars and Scroll Buttons

Figure 7.2 Scroll buttons



Scroll buttons display in the lower right corner. They should only scroll a page at a time. They take space away from the row of command buttons.

When choosing between scroll bars and scroll buttons, consider the following:

- Will users want to scroll a line at a time?

Scroll bars support scrolling both a line at a time and a page at a time. Scroll buttons scroll a page (one screenful of information) at a time.

For most multi-line editable text fields, most users want both modes of scrolling available, so multi-line text fields should always use a scroll bar. For other user interface elements, either scroll bars or scroll buttons will do if scrolling a page at a time is sufficient.

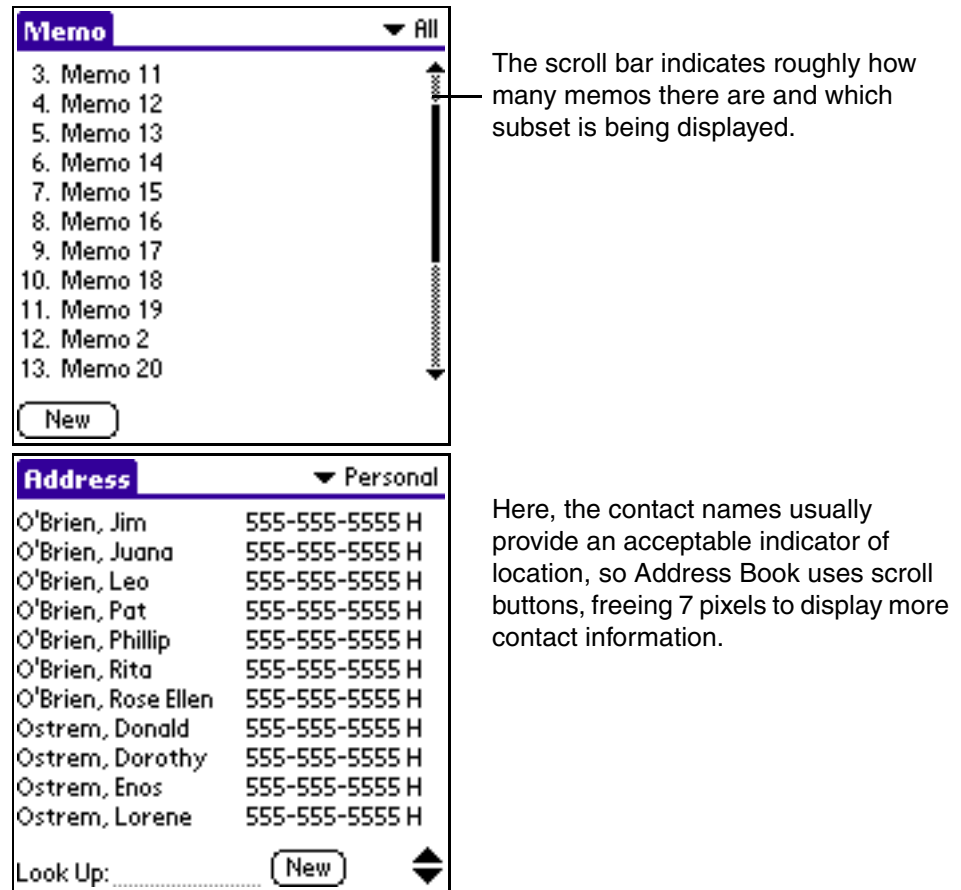
- Do you need an indicator of location?

Scroll bars provide a good indicator of the location of the data within the database (that is, whether the user is viewing the beginning, end, or middle of the data). Although scroll buttons show when the user is at the beginning or end of the data, they give no indication of how far until the beginning or end when the display is somewhere in the middle.

Because scroll bars are the better indicator of location, you should use scroll bars as long as you can sacrifice seven horizontal pixels of your data. If there already is a fairly clear indicator of location and you can't sacrifice the space, use scroll buttons. For example, the Address Book list view is sorted alphabetically, so users typically know where they are

within their own data. Address Book uses scroll buttons so that it can display more of the contact's phone number or email address (see [Figure 7.3](#)).

Figure 7.3 Showing location



- Will your target audience be able to navigate using a scroll bar?

Scroll bars are tiny targets for the pen, making them difficult to use in certain situations. Your target audience may have difficulty using scroll bars if they use your application while in a car, on a train, or if they are older. Because scroll bars are on the right side of the screen, left-handed users often can't

Scrolling

Scroll Bars

see how far they've scrolled. They become annoyed with applications that rely entirely on scroll bars.

If a significant percentage of your target audience is mobile or older, you may decide that they are better supported by scroll buttons. However, users are often satisfied as long as they can use the scrolling hard keys on the handheld. If you want to use scroll bars, just be sure to support the scrolling hard keys as well.

- Do you display any right-justified data?

If you display any controls on the right edge of the screen, you probably are better off using scroll buttons. Otherwise, a user who aims for the scroll bar and misses might mistakenly tap your control.

For example, the rightmost column of the Address Book list view is a control. The user can tap this column to enter a note about the current contact. If the Address Book were to have a scroll bar immediately to the right of this column, the user might mistakenly tap the note column when aiming for the scroll bar.

Scroll Bars

A scroll bar scrolls the display of another user interface item (see [Figure 7.4](#)). Scroll bars can be attached to multi-line text fields, tables, lists, or forms.

Figure 7.4 Scroll bar

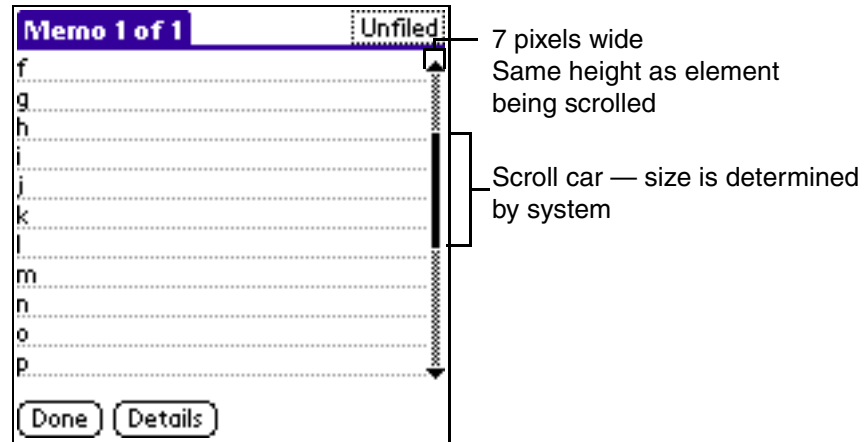


Table 7.1 Scroll bar details

Dimension	Value
Width	7
Height	Height of element that the scroll bar scrolls
Top	Align with top of element to be scrolled
Left	153
Border	None

System Supplied Behavior

A scroll bar has four values that determine its behavior:

- The minimum value, which is typically 0.
- The maximum value, which is often not set until run time. This value is the number of lines it should take to scroll to the end of the element, which is computed with the formula:

$$\text{number of lines of text} - \text{page size} + \text{overlap}$$

where *number of lines of text* is the total number of lines or rows needed to display the entire object, *page size* is the number of lines or rows that can be displayed on the screen at one time, and *overlap* is the number of lines or rows from

the bottom of one page to be visible at the top of the next page.

- The initial value, which is usually the same as the minimum.
- The page jump value, or the amount by which the value is increased or decreased when the user scrolls. This should be at least one less than the number of lines that can be displayed at one time to provide context. That is, if the field can display ten lines of text, the page jump value should be nine.

Palm OS ensures that the scroll bar does not appear until there is more than a screenful of information to be presented. It also ensures the behavior described in [Table 7.2](#).

Table 7.2 Default behavior of scroll bars

When...	Then...
User drags scroll car	Scroll car moves in the direction indicated
User taps scroll arrow	Scroll car moves up or down an amount that indicates a single line
User taps in gray area immediately above or below scroll car	Scroll car moves up or down by an amount that indicates a page
User drags in a text field beyond the edge of the text field	Scroll car moves up or down along with the text

The system does *not* scroll the user interface element to which the scroll bar is attached. Your application's code must do that.

Look and Feel

Follow these guidelines for implementing a scroll bar.

Vertical Scroll Bars Only

Palm OS only provides support for scrolling vertically. Horizontal scrolling is not supported.

Support Immediate Mode Scrolling

There are two scrolling modes possible: immediate mode and non-immediate mode. In immediate mode scrolling, the scrolling happens as the user drags the pen. This is preferred because it allows the user to see how far the field is moving. Non-immediate-mode scrolling waits until the user lifts the pen before scrolling the display.

Support the Scrolling Hard Keys

You should always allow the scrolling hard keys on the handheld itself to scroll any user interface element. They should have the same effect as tapping the gray area above and below the scroll car. That is, they should scroll up and down a page at a time.

Breaking the Rules

Because Palm OS does not support horizontal scrolling, you should avoid creating a user interface that requires it. In most cases, users do not expect to have to scroll a display horizontally. Web browsers, however, often must support horizontal scrolling to support Web pages with large graphics or large tables.

Scroll Buttons

The scroll buttons are two arrows that appear at the bottom of the screen (see [Figure 7.5](#)). If the user taps the arrow with the pen, the form scrolls in the direction indicated. If the user holds the pen down on an arrow, the form repeatedly scrolls in the direction indicated.

Figure 7.5 Scroll buttons

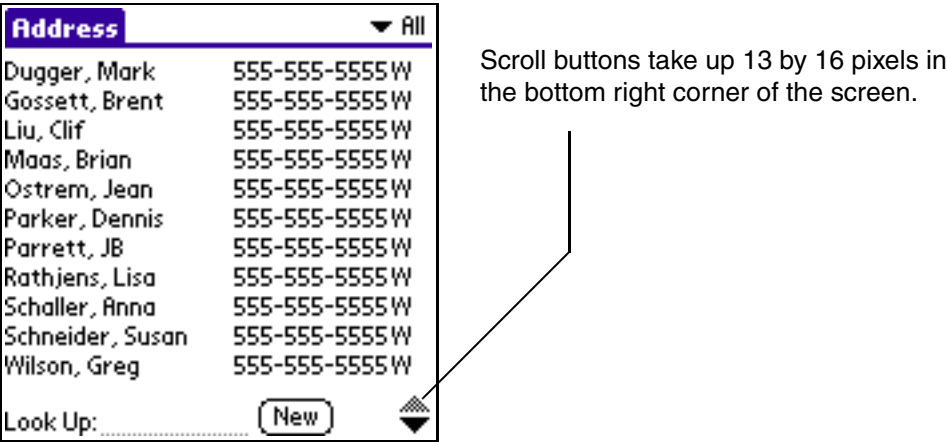


Table 7.3 Scroll button details

Dimension	Value
Width	13 pixels
Height	8 pixels per button
Top	144 for up button 152 for down button
Left	147
Font	Symbol 7
Contents	0x01 for up button 0x02 for down button
Border	None

System Supplied Behavior

Palm OS ensures the following:

- The button highlights while it is being pressed.
- The scroll buttons don't appear until there's more than a screenful of information to display.
- The up arrow is grayed out when the first screenful is displayed.

- The down arrow is grayed out when the last screenful is displayed.

The system does not know what user interface element is to be scrolled; therefore, your code must inform the system when there is more than a screenful of information to display, when the top of the data is displayed, and when the bottom of the data is displayed. If your application informs the system of these conditions appropriately, you'll see the behavior described above.

Also note that your application's code is responsible for actually scrolling the display.

Look and Feel

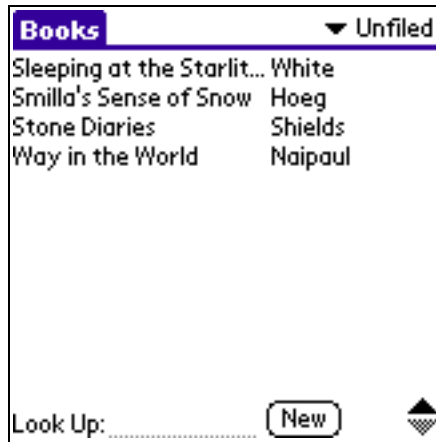
Follow these guidelines when implementing scroll buttons.

Always Show a Screenful of Information

Screen space is so limited that you should always show users as much of their data as you possibly can. For this reason, you should always adjust the scrolling so that a full screenful of information is displayed.

Suppose your form has a table containing ten rows and there are fourteen items to be displayed. If the user taps the down button to scroll down ten items, the next screen only has four items to be displayed (see [Figure 7.6](#)). Instead of displaying just those four items, scroll backward enough to display the previous six items as well.

Figure 7.6 Form at end of data



This display has scrolled until the end. The last screenful has only four items to display. It should scroll back until it shows a complete screenful of items so that users can see as much data as possible.

Support the Scrolling Hard Keys

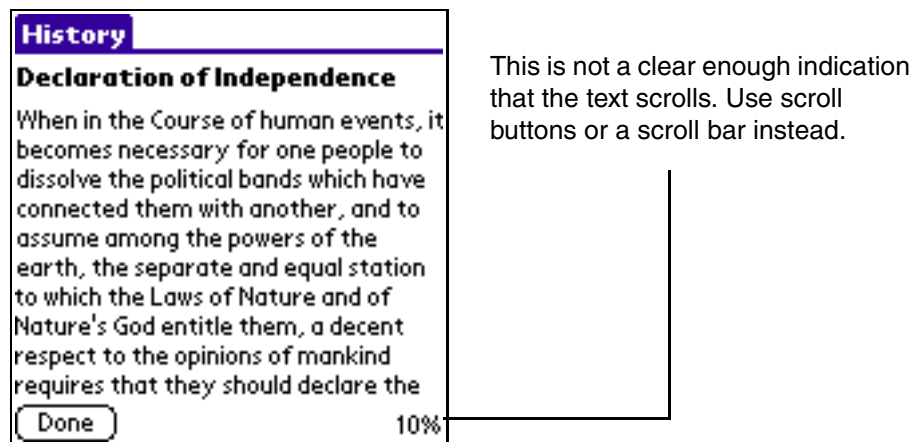
Users expect to be able to use the scrolling hard keys in all applications that support scrolling. The up hard key should perform the same function as the up scroll button, and the down hard key should perform the same function as the down scroll button.

Breaking the Rules

Never rely solely on the scrolling hard keys to scroll your interface. Always include either the scroll buttons or a scroll bar to indicate that scrolling is possible.

Some applications omit both the scroll buttons and the scroll bar in favor of another indication that there is more information to display. Often, this takes the form of a label or graphic that specifies the percentage of data that has already been displayed (see [Figure 7.7](#)). A percentage is not a clear enough sign that you can scroll to see more information. For this reason, you must always have some form of scrolling user interface element in your display.

Figure 7.7 Scrolling with no buttons



Scrolling

Scroll Buttons





Color and Graphics

This chapter provides guidelines for the use of color in a Palm OS® application.

Palm OS Color Support

Palm OS supports handhelds with the system palettes shown in [Table 8.1](#).

Table 8.1 Supported system palettes

Palette	Colors
Monochrome	
2-bit grayscale	
4-bit grayscale	
8-bit color	

Color and Graphics

Colors of User Interface Elements

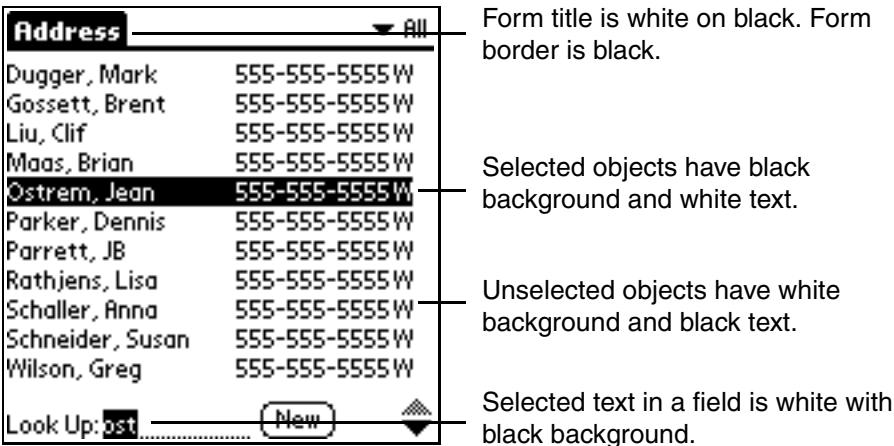
In addition, some handhelds can display application icons and other graphics with 16-bit color. On these devices, 16-bit color is used for graphics only; buttons and other user interface elements continue to be displayed with 8-bit color.

Colors of User Interface Elements

Palm OS uses the same color scheme for user interface elements on monochrome and grayscale screens. The scheme for color screens is different.

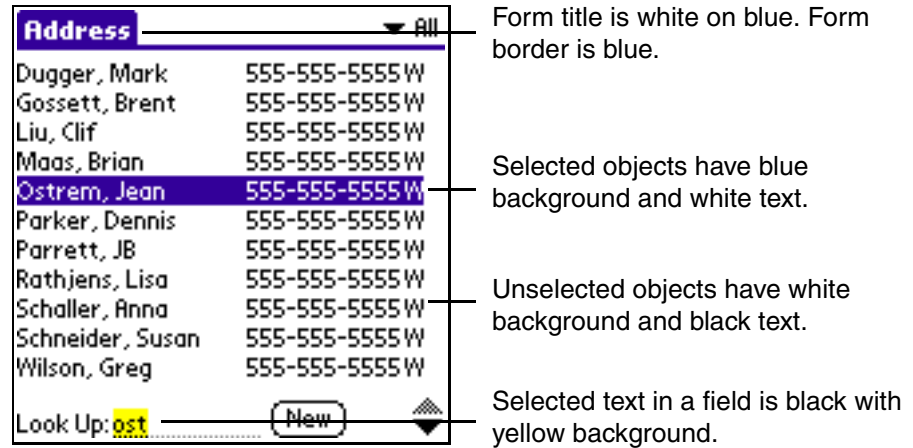
In general, monochrome and grayscale screens use black text on a white background. Highlighted or selected text or objects show white text on a black background. (see [Figure 8.1](#)).

Figure 8.1 User interface colors on monochrome/grayscale



On color screens, selected objects show white text on a blue background. Selected text is black with a yellow background (see [Figure 8.2](#)).

Figure 8.2 User interface colors on color screen



Respect the system color scheme. Do not change it arbitrarily. Consider that there may be a third party application that allows users to select their own color schemes. If you set your application to use a custom color scheme, you violate those users' preferences.

If you do change the colors for user interface elements for your application, the system will change them back when the user switches applications. You'll need to set the colors again when control returns to your application.

Do not rely solely on color to convey a meaning in your application. Many Palm Powered™ handheld users have monochrome or grayscale devices and won't see your color changes. Even on color devices, realize that some users have color-vision deficiencies and will miss your visual cue. Always use color in conjunction with some other visual cue so that those with color-vision problems also understand the meaning.

Graphics

When designing graphics, use color and grayscale prudently to help the graphic convey its meaning.

For the best possible appearance, you should provide a version of your graphic for each of the supported palettes: monochrome, 2-bit gray, 4-bit gray, 8-bit color, and 16-bit color. This means for the

application icon, you'll provide up to ten different versions of the icon: one large icon and one small icon for each palette.

If you don't provide a graphic for all supported palettes, Palm OS uses the bit depth representation closest to the bit depth of the screen. For example, if you provide only a monochrome graphic and an 8-bit color graphic, handhelds that run in 2-bit grayscale and 4-bit grayscale will choose the 1-bit graphic. All color handhelds will use the 8-bit graphic.

Keep in mind that screens with different resolutions are available, and your graphic may look quite different at the different resolutions. A 160 dpi high-resolution screen is capable of displaying a much finer granularity than an 80 dpi screen. If you have previously designed a low-resolution 8-bit color icon, you may find you need to add detail to its high-resolution counterpart.

For example, [Figure 8.3](#) shows different variations of the icon for a Mail application. The icon on the left is the 80 dpi 8-bit color version of the icon. If this icon were simply translated into high-resolution color by smoothing the rough edges, it would look like that shown in the middle. This icon design is somewhat flat. The icon on the right shows more detail. It adds some depth to the envelope and includes a postmark and more detail in the stamp.

Figure 8.3 Mail application icon at different resolutions



For more information about designing icons, see “[Application Icons](#)” on page 27.

Ten Things to Remember

This appendix summarizes ten design guidelines for Palm OS® applications that developers often forget or ignore. It then tells you which sections in the book explain these guidelines in more detail.

1. Consider the user.

The most common mistake handheld application designers make is the most common mistake made by all application designers everywhere: they fail to learn about the user before designing the application and thus fail to provide an application that solves the user's problems.

[“The Design Process”](#) on page 10 describes a process that helps you to create an application with the user in mind at all times.

2. Don't overcrowd the screen.

A user must be able to learn your application quickly and navigate through the display quickly. If you clutter the screen with too many controls, you slow the user down. See [“Fast and Simple”](#) on page 3 for more information.

3. Don't provide an Exit command button or an Exit menu item.

Palm Powered™ handheld users do not exit applications. They simply move to another application. See [“Don't Provide Save or Exit Commands”](#) on page 77.

For a description of when breaking this guideline is allowed, see [“Including an Exit Command”](#) on page 94.

4. Let the user leave your application.

You must allow the user to exit the application at any time. If you're displaying a modal form or an alert dialog, you must provide a default button. When the user exits your form, the

Ten Things to Remember

system simulates the default button tap. See the following sections:

- [“Exiting the Application”](#) on page 40
 - [“The User Can Exit at Any Time”](#) on page 56
 - [“The User Still Can Exit at Any Time”](#) on page 62
- [“Modal Forms that Don’t Exit”](#) on page 59 describes the one allowed exception to the rule.

5. Launch quickly.

Applications should not display a splash screen when launching unless this is the first use of the application. Most applications should return to the last form that was being displayed before the user exited the application rather than always displaying a main screen at launch. See [“Launching the Application”](#) on page 39.

6. Use modal forms sparingly.

In general, modal forms should be used for a single specific task only. Modeless forms are greatly preferred over modal forms. See [“Choosing between Types of Forms”](#) on page 45 for further discussion of when to use modeless and modal forms.

7. Don’t duplicate commands.

There’s usually not a good reason to provide both a command button and a menu item for the same command. See [“Don’t Duplicate Commands”](#) on page 74 and [“Duplicating Menu Items”](#) on page 95.

8. Remember the shift indicator and Edit menu.

All forms with editable text fields must include a shift indicator and an Edit menu. Even modal forms require the Edit menu to be present. See the following for further discussions:

- [“Edit Menu Required”](#) on page 132
- [“Shift Indicator Required”](#) on page 132
- [“Enable Auto-Shifting for Most Text Fields”](#) on page 133

9. Set the field focus.

In almost all cases if you have an editable field on a form, the field should have the focus when the form is first displayed. This must be set programmatically. See “[Set the Focus When the Form Is Displayed](#)” on page 133.

10. Leave room for the border of an object.

For all user interface elements, the border is drawn outside of the bounds of the object. The guidelines state that all objects should begin at the left edge of the screen. If, for example, you place a command button on the screen, it should actually start at pixel 1 to allow for the 1-pixel button border.

See “[Left-Align Buttons at the Bottom of the Form](#)” on page 79 for further discussion.

Ten Things to Remember

Index

Numerics

80-20 rule 3, 73

A

about dialogs 67–68
 command buttons 84
 displaying 89
 guidelines 68
About menu item 89
active form 50
Address Book
 category selector trigger 121
 Delete command 74
 Edit form 31, 50, 116, 121
 exiting 40
 fields in 128
 forms in 45
 initial form 31
 minimize taps 75
 navigation 18
 notes 69, 145
 optimize frequent tasks 7
 power user features 7, 77
 Record menu 95
 scrolling 148, 150
 startup screen 18
 tables 144
 title bars 51
alarms 43
alert dialogs 60–65
 command buttons 63
 controls on 65
 default button 62
 Delete confirmation 59, 85
 disabled buttons 81
 exiting 40, 62
 guidelines 61
 message 63
 reminders 63–64
 title bar 62
 types 62
alpha development 21
application design
 data entry 24
 principles 1–10

application exit. **See** exiting the application
application icons 27, 162
application launch guidelines 39–40, 164
Application Launcher. **See** Launcher
application version number 28
application views 102
arrangement of controls on form 34
Attention Manager dialog 60, 63–64
auto-completion 135
auto-shifting 133

B

background of slider 124
BarBeamBitmap 94
BarCopyBitmap 94
BarCutBitmap 94
BarDeleteBitmap 94
BarInfoBitmap 94
BarPasteBitmap 94
BarSecureBitmap 94
BarUndoBitmap 94
basic design principles 1–10
batteries 8, 9
beaming 9, 12
behavior guidelines 39–43
bitmaps. **See** graphics
Books sample application 11, 13
 initial design concept 20
 lists and tables 141
 proposed implementation 18
 user assessment 15
 user scenarios 17
borders of controls 80, 165
built-in applications 6, 25
button toolbars 2
buttons. **See** command buttons, push buttons,
 repeating buttons, scroll buttons

C

Calculator 53
Cancel button 57, 80
 about dialogs 67
categories 38–39, 50
 Launcher application 29

- pop-up list 110, 112, 122
- selector trigger 121
- Unfiled 38
- check boxes 100, 106–108
 - contents 107
 - guidelines 106
 - labels 107
 - when not to use 104
- color 17, 159–161
 - 16-bit 160
 - use of 161
 - user interface elements 160
- combo box simulation 104–106
 - command buttons 83
 - pop-up lists 114
- command buttons 78–86
 - See also** under specific button name
 - about dialogs 67, 84
 - alert dialogs 63
 - alignment 79, 84
 - centering 84
 - choosing number on a form 5, 73
 - combo box simulation 83, 105
 - default button on dialogs 56, 62, 163
 - disabled 81
 - double taps 79
 - graphics in 36, 81
 - guidelines 78
 - i button 69
 - menus, compared with 71–77
 - menus, duplicating 74, 95
 - modal dialogs 57
 - names 80
 - placement 79, 83
 - progress dialogs 66, 84
 - repeating 82
 - system supplied behavior 79
 - toggle buttons 120
- command toolbar. **See** shortcuts, toolbar
- commands 71, 73
 - destructive 74
 - duplicating 74, 95, 164
 - Exit 40–41, 77, 94
 - novice user commands 77
 - pop-up lists 112
 - Save 77

- common names of command buttons 80
- compatibility worldwide 43–44
- completing the design 21
- conduits 9, 10, 48
- confirmation alert dialog 62, 74
- consistency 6, 75
- Constructor for Palm OS 70, 104
- controls
 - See also** under specific control name
 - on alert dialogs 65
 - borders of controls 80, 165
 - custom 37
 - graphical 36, 81
 - placement guidelines 34
 - right edge of screen 150
 - table of 32
 - in title bar 51
- copyright notices 68
- cursor 130

D

- data
 - columnar 128
 - display 127–129
 - entry 2, 23–26, 47
 - right-justified 150
 - secondary 129
 - validation 48, 136
- Date Book
 - alarm dialog 60, 85
 - appointment reminders 63–64
 - command duplication in 95
 - Go To Date form 59
 - hard key remapping 42
 - initial form 4
 - minimize taps 5, 75
 - new record 95
 - optimize frequent tasks 6
 - power user features 7
 - Preferences dialog 103
 - push buttons in Change Repeat dialog 118
 - Record menu 95
 - title bar 51, 52
 - user scenarios 16
- dates 42, 103

- days of the week 42
- default application category 29
- default button on modal forms 56
- Delete button 74, 80
 - ellipsis 85
- Delete confirmation 59, 74, 85
- design completion 21
- design concept 19–21
- design goals 13–14
- design principles 1–10
- design process 10–20
- Details button 80
- Details dialog 80
 - and categories 38, 122
 - novice user controls 113
- dialogs. **See** about dialogs, alert dialogs, modal dialogs, progress dialogs, tips dialogs
- digitizer 24
- disabled buttons 81
- disabled menu items 89
- displaying data 127–129
- divider lines in menus 90
- document reader 36, 129, 156
- Done button 57, 80, 136
- double taps 79, 130
- duplicating commands 74, 95

E

- Edit button 80
- Edit Categories command 112
- Edit Categories dialog 129
- Edit command in pop-up list 106
- Edit menu 89, 90
 - modal dialogs 58
 - omitting 164
- editable fields 131
- editing in place in tables 144
- ellipsis guidelines 81, 84, 90
- error alert dialog 62
- executing commands 71
- exit command 40–41, 77, 94
- exiting the application 40–41, 77, 94

- alert dialogs 62
- modal dialogs 56, 59

Expense 113

F

- feedback sliders 124
- fields 129–137
 - auto-completion 135
 - auto-shifting 133
 - data entry 135
 - editable 131
 - features not supported 131
 - guidelines 130
 - insertion point 133, 165
 - keyboard considerations 135, 136
 - maximum character limit 131
 - multi-line 131, 135
 - placement 131, 136
 - scroll bars 135
 - scrolling 131, 147
 - shifting 133
 - single-line 134
 - system supplied behavior 130
- Find facility 41
- find, incremental 111
- finger navigation 86
- focus in field 165
- fonts 35
 - and text fields 131
- forms 31, 48–55
 - active 50
 - base form 39
 - guidelines 49
 - initial form 4, 31, 39
 - menus 74, 89
 - modal. **See** modal dialogs
 - modeless vs modal 45–48
 - navigation 12, 50, 73
 - Please Wait form 66
 - scrollable 50
 - size of 50
 - system supplied behavior 49
 - title bar 39, 50–55
 - views on a form 102

G

- gadgets 37
- games 52, 73
- global Find facility 41
- goals 13–14
- Graffiti 2 power writing software 2, 24
- Graffiti or Graffiti 2 menu command stroke 91
- Graffiti power writing software 2, 8, 24
 - navigation characters 135
- Graffiti Shift Indicator. *See* shift indicator
- Graffiti shortcuts, *See* shortcuts
- graphical controls 36, 81
- graphics 161–162
 - application icons 27, 162
 - in controls 36, 81
 - palettes 161
 - scrolling 147
 - sliders 124
- grayscale displays 160

H

- handheld users 14–15
 - See also* users
- hard keys 25
 - exiting application with 40
 - powering on device with 59
 - remapping 42
 - scrolling 153, 156
- hardware controls 25
- help 69
- high-resolution screen considerations 162
- horizontal scroll bars 153
- HotSync operations 9, 25

I

- i button on modal dialogs 69
- icons
 - application 27, 162
 - input area 25, 42
- implementation, proposed 17–19
- incremental search 111
- indicator of location 148, 156
- information alert dialog 62
- initial design concept 19–21

- initial form 31, 39
 - DateBook example 4
- input area 24
 - icons 25, 42
 - placement of controls near 34, 51, 57
- insertion point 130, 133, 165
- IR communications 9

K

- keyboard shortcuts, *See* shortcuts
- keyboards
 - dialog 24
 - external 26, 136
 - lack of 2

L

- labels 34
 - check boxes 107
 - compared to contents terminology 35
 - pop-up lists 112
 - selector trigger 120
- launch guidelines 39–40, 164
- Launcher 26–30
 - form 50
 - icons in 27
 - Info dialog 28
 - Please Wait form 66
- lists 129, 137–142
 - See also* pop-up lists
 - box, suppressing drawing of 141
 - guidelines 137
 - maximum number of elements 141
 - scroll bars 139
 - scrolling 138, 147
 - system supplied behavior 138
 - tables, compared with 128, 129, 140
- localization guidelines 43–44
 - pop-up lists 102
 - push buttons 102
- location indicator 148, 156
- low-battery warnings 43

M

- Mail
 - icon 162

- title bar 51
- maximizing battery life 9
- Memo Pad
 - category selector trigger 121
 - Details form 107
 - Edit form 121
 - ellipsis 85
 - exiting during data entry 48
 - fields in 127
 - menus and buttons 75
 - scrolling 149
- menu command stroke 91
- menus 2, 86–97
 - boundaries 88
 - command buttons, compared with 71–77
 - command buttons, duplicating 74, 95
 - disabled menu items 89
 - displaying previous menu item 88
 - divider lines 90
 - Edit 58, 89, 90
 - ellipsis guidelines 90
 - guidelines 87
 - names of menus and items 90
 - New menu item 74
 - Options 89, 90
 - Record 90, 95
 - shortcuts, **See** shortcuts
 - submenus 88
 - system supplied behavior 87
- messages 63
 - alarms 43
 - low-battery warnings 43
 - synchronization 43
 - system 43
- minimize taps 5, 75, 122
- modal dialogs 55–60
 - See also** about dialogs, alert dialogs, progress dialogs, tips dialogs
 - alignment 57
 - command buttons 57, 59
 - default button 56, 163
 - exiting 40, 56
 - guidelines 55
 - height 56
 - selector triggers, displayed by 120, 121
 - system supplied behavior 56

- title bar 55
 - when to use 45–48, 164
- modal forms. **See** modal dialogs
- modeless forms. **See** forms
- monochrome displays 160

N

- names of command buttons 80
- navigation
 - between forms 12, 50, 73
 - finger 86
 - Graffiti navigation 135
 - scroll bars 149
- New button 73, 81
- New menu item 74
- No button 63
- Note button 81
- Note view title bar 54
- novice users 77
 - pop-up lists 113
- numbers 42

O

- OK button 57, 80
 - about dialogs 67
 - alert dialogs 63
 - excluding 59
- online help 69
- options 99
 - one of many 100
 - several of many 100
- Options menu 89, 90
- orienting users 51

P

- palettes supported 159
- Palm VII handhelds 16
- passwords 120
- perceived speed 4
- phone lookup 83
- Please Wait form 66
- pop-up lists 101, 108–116
 - combo box simulation 105, 114
 - commands in 111, 112

- Edit command in 106
- guidelines 108
- hierarchical 111
- incremental search 111
- nonstandard triggers 114
- push buttons, compared with 102
- scrolling 102, 110
- selector triggers, compared with 103
- system supplied behavior 110
- triggers 108
- upper limit 102
- when to use 102
- pop-up triggers 108
 - nonstandard 114
 - omitting arrow 113
- power user features 7, 77
- preferences 42
- Preferences menu item 89, 90
- private records 41
- process for user interface design 10–20
- processor 8
- progress dialogs 65–67
 - command buttons 84
 - guidelines 65
 - purpose 66
 - title bar 65
- proposed implementation 17–19
- push buttons 101, 116–118
 - guidelines 117
 - not mutually exclusive 118
 - pop-up lists, compared with 101
 - selector triggers, compared with 101
 - system supplied behavior 117
 - two choices 101
 - when to use 101–102

Q

quitting. **See** exiting the application

R

- Record menu 90, 95
- reducing battery usage 8
- registration information 68
- reminders 63–64
- repeating buttons 82

S

- save command 77
- scenarios 15–17
- screen navigation 12, 50, 73
- screens 8
 - color 161
 - grayscale 160
 - monochrome 160
 - resolutions 162
 - sizes and resolutions 49
- scroll bars 147, 150–153
 - fields 135
 - guidelines 151
 - horizontal 153
 - lists 139
 - scroll buttons, compared with 147, 147–150
 - system supplied behavior 151
- scroll buttons 147, 153–157
 - guidelines 154
 - scroll bars, compared with 147–150
 - system supplied behavior 154
- scrolling 147–157
 - fields 131
 - forms 50
 - horizontal 153
 - line by line 148
 - lists 138
 - page by page 148
 - pop-up lists 102, 110
 - tables 155
- scrolling hard keys 153, 156
- search, incremental 111
- selecting text 130
- selector triggers 101, 119–122
 - categories 121
 - combo box simulation 83, 104
 - guidelines 119
 - label 120
 - modal dialogs 120, 121
 - pop-up lists, compared with 103
 - system supplied behavior 120
 - updating 120
 - when to use 103
- serial communications 9
- Set Time dialog 141

- shift indicator 132–133
 - modal dialog 58
 - omitting 164
- shifting 133
- shortcuts 91–94
 - common 92
 - duplicating commands for 97
 - Edit menu 89
 - enabling in modal dialogs 58
 - toolbar 91, 93–94
- single-line fields 134
- sliders 101, 123–125
 - compatibility considerations 103
 - guidelines 123
 - system supplied behavior 123
 - types 124
 - when to use 103
- SMS Messenger 83
- speed 4
- splash screens 39
- startup guidelines 39–40
- startup screen, **See** initial form
- submenus 88
- synchronization 9, 25
 - messages 43
- system messages 43
- system palettes supported 159

T

- table of controls 32
- tables 128, 142–145
 - borders 144
 - column headings 144
 - columns, choosing number of 145
 - direct editing 144
 - guidelines 143
 - lists, compared with 128, 129, 140
 - programming difficulty 140
 - scrolling 147, 155
 - selection 144
 - system supplied behavior 143
- taps
 - double 79, 130
 - minimize 5, 75, 122
 - triple 130

- text fields. **See** fields
- text selection 130
- thumb of slider 124
- time 42, 103
- time zones 103
- tips dialogs 69–70
- title bars 39, 50–55
 - about dialogs 67
 - alert dialogs 62
 - controls in 51
 - games 52
 - i button 69
 - modal dialogs 55
 - nonstandard 52
 - Note view 54
 - omitting 54
 - progress dialogs 65
 - purpose 51
- To Do List
 - and Attention Manager 64
 - command buttons 81
 - minimize taps 75
 - pop-up lists 113
 - tables 144
- toggle buttons 120
- toolbars 2
 - shortcut toolbar 91, 93–94
- triple taps 130
- types of alert dialogs 62
- typical user session 4
- typical users 14–15

U

- Undo command 77
- usability testing 21
- user input 2, 23–26
- user interface design
 - goals 13–14
 - process 10–20
- user scenarios 15–17
- users 14–15
 - color-vision problems 161
 - interaction 23–26
 - mobile 149
 - neglecting 163

- novice users 77, 113
- older 149
- orienting 51
- power users 77
- preferences 42
- usability testing 21
- user scenarios 15–17

V

- validation 48, 136
- version numbers 28, 68
- vertical market applications 30, 41

- views 102

W

- warning alert dialog 62
- web browsers 36
- Welcome application 16
- word wrapping 131
- worldwide compatibility 43–44

Y

- Yes button 63