Zen of Palm

Designing Products for Palm OS®

CONTRIBUTORS

Revised by Mark Dugger
Illustrated by John Grimes
Contributions by Monty Boyer, John Cardozo, Gina Clark, Bob Ebert, David Fedor, Roger Flores, Rob Haitani, Jeff Hawkins, Michael Lunsford, Michael Mace, Jean Ostrem, Lon Poole, Chris Raff, Maurice Sharp, Phillip Shoemaker, Joe Sipher, and Carl Stone.

Zen of Palm
Document Number 3100-002-HW
June 13, 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

Introduction

Palm, Inc. is the leader and standard bearer in the handheld market. Building on the experiments and near misses of predecessors, Palm introduced the first handheld device to achieve resounding success in the marketplace. Several years later, we at PalmSource, the software spin-off of Palm, Inc., continue to produce the operating system that runs on the best handhelds in the world.

We also proudly develop, upgrade, and distribute the premier operating system for handheld devices. The Palm OS® is the platform for a whole family of Palm-Powered™ products, including many fine devices from other companies that license the Palm OS. With a market share of 80%, the Palm OS platform is the first choice of any developer who wishes to reach the widest customer base of handheld owners in the world.

If you are a developer interested in writing applications for the Palm OS platform, you should be aware of the diversity of Palm-Powered devices that make up this market. They start, of course, with the Personal Digital Assistants that help customers organize their personal data, including address book, datebook, to do list, and notes. Wireless devices are the newest frontier, combining the advantages of a PDA plus wireless connectivity to E-mail and the internet. Wireless products include the Palm™ VII series and smart phones. Beyond these more familiar devices, there are successful products like the inventory readers with scanner attachments from Symbol and scientific and medical instruments that use the Palm OS.

PalmSource, Inc. achieved this unrivalled level of adoption by developing design philosophies and practices that enable the creation of breakthrough products. From the beginning, our design has been guided by utility, portability, and focus. These principles are an essential part of the culture, not just a catchy bit of differentiation.

And they apply equally to hardware and software. In fact, at PalmSource we try to integrate the two so completely that users

don't even need to think about the difference. If you are a developer, your applications can blend in seamlessly with the Palm OS platform. To achieve this integration, start with same design philosophies and methods we use at PalmSource.

At first, you may find the principles a little unfamiliar, especially if you have been developing for PCs or laptops. But that makes sense. After all, a handheld is not a PC! By designing the PalmSource way, you will learn to develop applications for a new and exciting kind of electronic device. The challenges and rewards will be new, too. You are entering the world of handhelds.

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/

1

The Path to Enlightenment

The design philosophies and practices developed at PalmSource, Inc. target a specific kind of computing instrument—the handheld device. Handhelds include personal digital organizers, wireless communicators, smart phones, medical and scientific instruments, inventory trackers, and a host of other devices. The universe of Palm Powered™ handheld products, though varied, is unified by three principles:

- Palm OS® solutions are user-centric.
- Palm Powered™ handhelds can be used anywhere, anytime.
- Palm OS applications have a very specific focus.

First is the user-centric nature of Palm OS solutions. Handhelds and the applications they run must be immediately usable and completely convenient. Customers buy a handheld because they need to solve a pressing problem. They do not purchase one from some vague sense that they need a computer. Likewise, Palm OS applications must adhere to the usability and convenience customers have come to expect.

Next, Palm OS solutions must be available anywhere, anytime. The handheld customer requires computing power in places that PCs and laptops cannot conveniently go. If the effort to stuff features into a handheld goes unchecked, the device may become too heavy or bulky to be carried around easily. By definition, a handheld has to fit in the hand—and a human hand, at that.

Finally, a handheld application has a sharply defined "sense of purpose." Tightly focused on a particular task of importance to the user, it is the right tool for the job. The specificity of Palm OS applications follows from the nature of a handheld device. Unlike generic computing machines, handhelds are specially-designed devices used in very specific ways. To understand the difference,

imagine you have a wool suit that is badly wrinkled. You could use a steam roller to iron it, but you would be better off with a steam iron. Focus is the name of the game. The Palm OS platform is not the realm of large, vaguely-defined, monolithic applications.

Design philosophies and practices in the world of personal computers are very different from the Palm OS design philosophies and practices. This booklet explains why they're different and how they're different so that you can create successful Palm OS applications.

Rather than jump right into the nuts and bolts of the Palm OS user interface—placement of buttons and icons and so forth—we'll take a step upward and survey a 10,000 ft. view of Palm OS design philosophies and practices. This design overview will explore the following four topics:

- Design Philosophies
- Design Practices
- Design Validation
- Design Improvements

Because designing for the Palm OS platform may involve new concepts and ways of thinking, we are going to examine these topics through the means of several riddles. In the manner of a Zen koan, the answer may overturn the conventional view and expand your understanding. With due apologies to all Zen masters and students, we hope that the paradoxes wrapped in these riddles will provide at least a little enlightenment about application design in the handheld world. Each riddle introduces a section of this booklet, and you may choose to contemplate the paradox while reading the discussion. If you prefer, you may simply read the main text of the section, which fully elucidates the topic. Feel free to skip the riddle and the cartoon (though you may find it hard not to peek).

Design Philosophies

To discover the essence of the Palm OS design philosophy, ponder the first riddle.

Riddle #1 **Q: How can a gorilla learn to fly?**

Hint: You must understand the essence of the gorilla.

Think about the inherent differences between a gorilla and an eagle. Because of their essential natures, the gorilla rules the jungle and the eagle rules the sky. Now think about the intrinsic differences between handhelds and personal computers (PCs).

A handheld is not just a little desktop or laptop PC. A handheld is something else. This is a fundamental lesson that is not so easy to comprehend. Of course you can spot some obvious differences, such as size, but there are implications and usage patterns that are more difficult to discern yet are vitally important when you design for handhelds.

The Essence of PCs

In the PC world there is a linear relationship between features and value. More features are always better. Steve Ballmer, Microsoft Corp. President and CEO, put it this way, "Software should get bigger every year." [1]

In PCs, more features are better

This is the essence of PC thinking: it's always better to have more features. The customer will be able to do more tasks and, not coincidentally, will have to upgrade hardware and software to get those benefits.

Lost in this formula is the user experience. Lost are the questions: More features for what purpose? What does the user really want to do most with his computer? How long does it take to learn to use, and how hard is it to remember, once learned? Does the computer become a tool for a job, or an alternate experience of its own, in which productivity and utility are forgotten?

In the relationship between features and user experience, a PC is like an sports utility vehicle (SUV). An SUV is large and can carry a lot of people and things. You can add heavy accessories to it without much of a penalty. For instance, you can add a ski rack on the roof, a bicycle carrier on the back, and a minibar in the backseat. If you were going on a camping trip, you could even store a week's worth of provisions in the rear compartment. In a pinch, you could even set up a small cot and sleep in it! None of these add-ons is a real liability. In all, bigger is better, and more gets you more.

Likewise, in the PC world, more features are better. New circuitry may soak up more electricity, but you are not likely to notice the increase. New components may add weight to the PC, but again, it's no big deal.

The Essence of Handhelds

A handheld is a different creature: it is like a sports car. An SUV is fine until you need to race in the Indy 500 or escape the bad guys in

1. Steve Ballmer, "Building a Platform on Web's Technology," interview by Karlin Lillington, *San Jose Mercury News*, 17 December 1999, Business section, morning final edition.)

a high-speed chase. A sports car doesn't have time for extras that will weigh it down. It has to be maniacally focussed on speed and maneuverability.

Handhelds excel at perceived speed.

A handheld must be quick to use. A handheld is like a sports car because it gets the user from one place to another quickly. The actual technical specifications of a handheld are of little real interest to the user. What matters is how quickly she can reach for the device, open it, find the appropriate information, and proceed with her other tasks. How long this interaction takes can be described as perceived speed. It is a measure of the user's subjective experience with the handheld. The user doesn't care how fast the wheels are spinning, if the car is elevated on a rack. People use handhelds to do things--now!

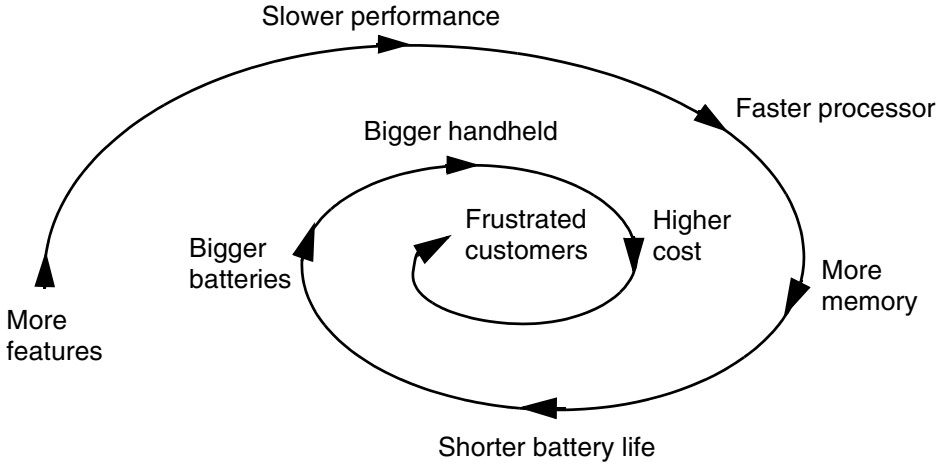
Too many features frustrate customers.

A necessary precondition of perceived speed is an uncluttered and essential set of features. It's fine to sell someone a knife that has 56 different uses, but if the user can't find the main blade or the bottle opener without flipping dozens of identical-looking levers, the knife is a novelty, not a tool. Likewise, a handheld application has to offer what the user needs to do and offer it in a way that's quick to learn and easy to use.

A handheld must be free to roam about.

Furthermore, handhelds take on features at a cost. Some hardware features may sound cutting-edge on the marketing brochure, but could increase power consumption enough to seriously degrade battery life. If one of the requirements of the handheld device was to operate without a recharge or new batteries for several days at a time, then the handheld's designers have lost focus. An unfocused and undisciplined loading of features makes the handheld bigger and heavier, leading to a spiral of doom, as shown in Figure 1.1.

Figure 1.1 More can be less



Handhelds must be wearable

Handhelds must be more than just portable. Handhelds must be so small and light that a person can carry one everywhere, in a pocket or a purse, without even thinking about it. If the device is a burden to carry, it will get left behind and not used. Ideally, it is like a clothing accessory that the user can "wear." PCs—even laptop PCs—don't have the same portability or power constraints as handhelds.

Handhelds are about the user

All this means one thing: there is a point of diminishing returns when adding features to a handheld. Adding too much degrades the user experience, as shown in Figure 1.2. And the user experience, not a list of features, is what a handheld is all about.

Figure 1.2 Feature list vs. user experience

Inverse Usage Patterns

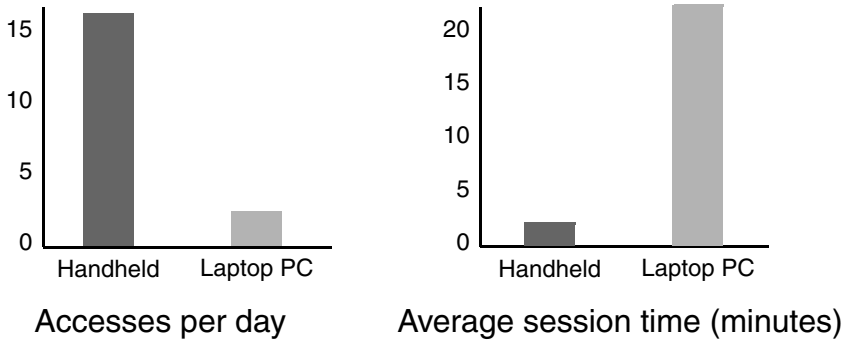
Usage patterns also differ fundamentally between handhelds and PCs. People tend to sit down at a desktop or laptop PC for a few long sessions, using the keyboard, large screen, and hard disk to create and edit large amounts of information. For example, a user opens a word processor or spreadsheet and works for half an hour.

Handhelds are used frequently but briefly

People generally use handhelds in frequent, short bursts—more like a watch than a PC. They take a handheld out of their pocket or briefcase to review and update small chunks of information. For example, they look up a phone number or quickly check their schedule. Figure 1.3 graphically contrasts how frequently and how long people use handhelds and laptop PCs.

In fact, the usage patterns of handhelds are exactly opposite those of PCs. Therefore, taking similar approaches to product design is a fundamental mistake.

Figure 1.3 Opposite usage patterns



Source: Palm, Inc. user surveys.

Different Design Approaches

When designing for handhelds, you need to take a different approach than when designing for PCs. Trying to fit a full desktop application in the palm of your hand is the worst mistake you can make. It will ultimately lead to failure.

PC Approach

In the PC world, users demand and expect to be able to do many kinds of complex activities. The PC is a tool for doing everything and anything. As a result, applications are typically stuffed with features. They try to cover as many contingencies as possible. In this world where there are few trade-offs for adding functionality, the more features you can give to customers, the better.

Handheld Approach

The practicalities of the handheld world call for a different design approach. Rather than brute force, you have to focus on clever solutions. Hone in on what really matters. A handheld application that is overstuffed with features actually fails its users. You must carefully consider what's important to put in and what's important to leave out. You will find this approach more challenging, but the sleek, usable handheld application that results will be worth your effort.

Power is the ability to get the job done

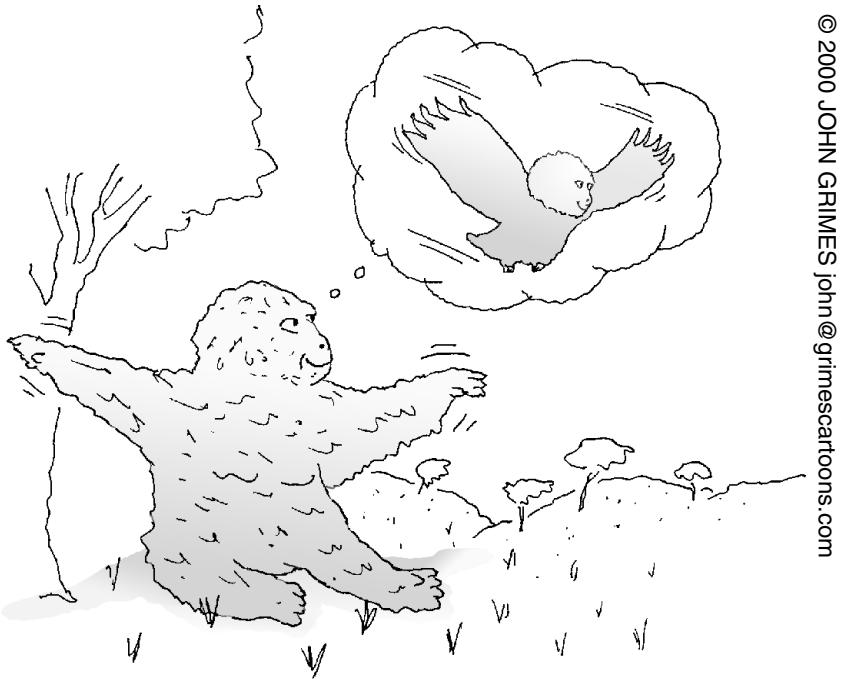
Don't misunderstand us: customers of Palm-Powered products do want power! But the power they seek is the ability to get the job done. If more features and functions makes a product cumbersome to use, they don't provide any real power.

If a few well-chosen features enable you to get the job done, you've created an application that is actually more powerful than a feature-heavy one that obscures the user's primary goals. Remember that utility and convenience equal power.

Solution to Riddle #1

Now we're ready to answer the first riddle. We understand the essential difference between PCs and handhelds. We know that when you add feature after feature to a handheld, you reach a point of diminishing returns and customer frustration. We know that people use handhelds and PCs differently—a few long sessions with a PC and many short sessions with a handheld. In these short sessions, too many features make a handheld cumbersome. On handhelds, the most straightforward designs are the most powerful.

Q: How can a gorilla learn to fly?



A: Only by becoming an eagle.

Learning to fly means learning to think like an aerodynamic creature—one that takes off, ascends, swoops, glides, and lands. It means leaving transportation of coconuts to the gorilla.

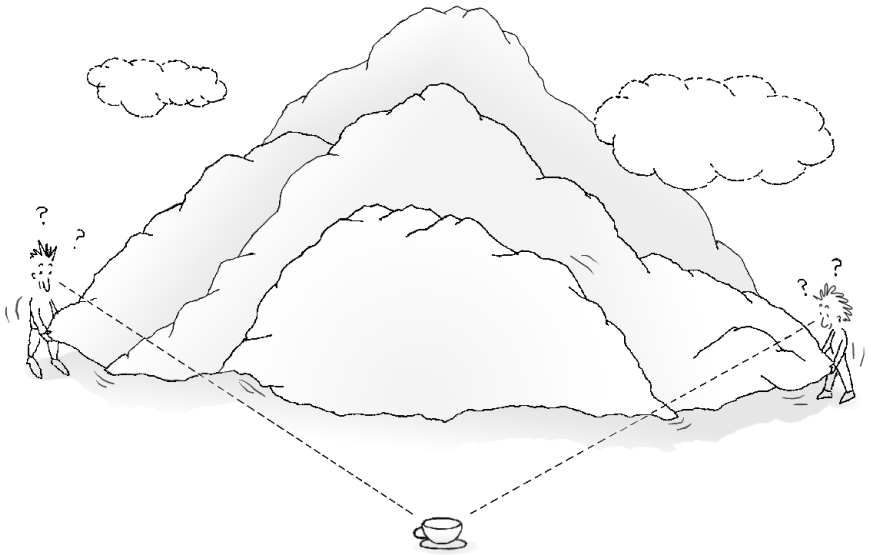
If you want to soar like an eagle and design successful products for the Palm OS platform, you must be willing to set aside instincts, knowledge, and experience that you acquired designing PC products.

A note of encouragement: Once you learn to design successful handheld applications, you will find pleasure in whittling down a vaguely-defined, feature-riddled PC application and making it fly on the handheld. You will learn that although the eagle might covet the gorilla's massive strength and long arms, if it had these features it could not get off the ground.

Design Practices

Now that we've established our philosophy, let's take a look at design practices. We will consider a more obscure and strange riddle.

Riddle #2 **Q: How do you fit a mountain in a teacup?**



Hint: Why put a mountain in a teacup?

You will not solve the second riddle or design a successful handheld product if you are still in the more-is-better PC mindset. Less-is-more thinking means taking a step back to ask what matters. Let's explore what matters in Palm OS applications.

A Balance of Features

Palm's first product was successful because it struck a balance among the following key qualities:

- Pocket size
- Fast response
- Easy to use
- Low cost and high value
- Worry-free battery life

- Seamless connection with PCs

All of these qualities are important individually, but Palm scored by integrating them all in a balanced way. Before Palm, excellence in one characteristic tended to rob the product in other characteristics. For example, data-bank organizers are tiny, fast, cheap, and have long battery life—but are so hard to use that they end up in drawers. The Newton and Magic Cap products were easier to use but were too big, cost too much, and devoured batteries. Palm OS products must incorporate all of the qualities listed above.

It's worth noting that the original Palm Powered device did not incorporate all of the key qualities by chance. These were the design goals that were laid out in advance: It must be small enough to be unconsciously portable. It had to be fast and easy to figure out. It must connect quickly to a PC. It's batteries must last a long time. It must meet customer price points.

Focus on benefits to users

The list of key qualities emphasizes benefits to users, not the underlying technical features that produce the benefits. PC product development typically proceeds from the opposite direction. The starting point is a checklist of technical features that sound impressive: 500MHz, 128MB, 12GB, and so on. The actual benefits users get from these features aren't always so clear.

Palm OS products succeed not on the sheer size of their technical features, but on the merits of their user benefits. Table 1.1 shows the relationships between the key user benefits of Palm OS products and the technical attributes that produced those benefits in the original Palm Powered handheld.

Table 1.1 User needs determine technical attributes

User benefit	Enabling technical attributes of first Palm Powered handheld
Pocket size	• Small display • High level of hardware integration • Small software footprint • Compact data structures
Fast and easy	• Compact, task-oriented software • Software has low overhead, low abstraction, and low modularity • One application runs at a time • Data stored in a low-overhead database in RAM • Graffiti® power writing • Minimal synchronization time
Low cost and high value	• Inexpensive components • Small RAM and ROM sizes[1] • Minimal hardware expansion • Secondary storage omitted from original product; support added in later models.[1]
Worry-free battery life	• Static/pseudo-static RAM • Small RAM and ROM sizes[1] • Automatic shut-off • Secondary storage support omitted from original product; added in later models after battery life issues were solved.[1]
Seamless connection with PCs	• Integrated, cooperative synchronization between handheld and PC applications • Common data manager design for easy application tracking of changes and low handheld processing during synchronization • Customized conduits synchronize all kinds of data during a single session • Synchronize via serial or USB connection

1. Current Palm Powered handhelds have expansion slots and serial ports.

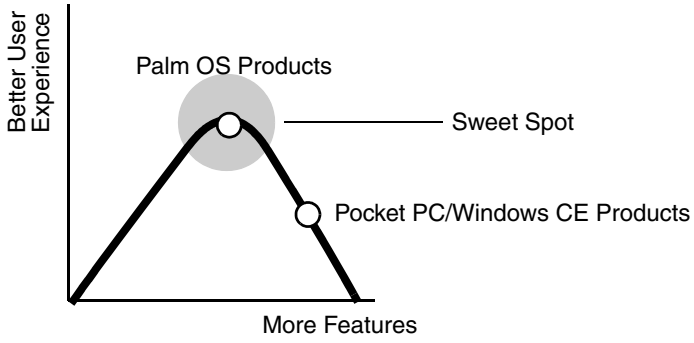
Nirvana: The Sweet Spot

When designing applications for handhelds, you must carefully determine which features to add and which ones to leave out. Recall from the earlier discussion about the essence of handhelds that including too many features in a handheld product degrades the user experience. For instance, too many features in an application can clutter the relatively small screen with lots of buttons and icons and such.

Balance features with the user experience

Your product needs enough features for the optimal user experience and no more. You may even have to omit some interesting features for the sake of a better overall user experience. The sweet spot at which you achieve an optimal balance of features and user experience appears at the apex of the curve in a graph of features vs. user experience, as shown in Figure 1.4.

Figure 1.4 Different views on new features



Determining an optimal set of features is like finding diamonds in a mountain. You don't want the whole mountain, just the important chunks.

The graph in Figure 1.4 also provides a reminder of the difference between PC thinking and handheld thinking. PC thinkers look only at the features axis. They neglect the question of whether the user benefits from these features and whether new features make the product too complicated to use.

Handheld thinkers look at the overall curve. They see that piling on features in the style of in the PC style would move Palm OS products lower on the user-experience axis. When PC thinkers

recommend rushing along the features axis to make a product seem better, handheld thinkers know this could make the product worse.

In the early days, critics said to Palm: "You guys are idiots for not matching PC features. Customers want more features no matter what. And they're going to decide to buy based on a check list of features." Palm product designers did their testing, held their ground, and brought out a product that people—ordinary people—could use from the first day and every day thereafter.

This conviction and leap of faith paid off. The Palm OS platform now commands 80% of the handheld market. PalmSource licensees market a host of useful and popular products based on these underlying principles.

Added features must improve the user experience

Handheld thinking is about balancing available technology and utility. When faced with the possibility of a new feature, ask two questions:

- What do we gain in user satisfaction?
- What does it cost in terms of user confusion and hardware resources?

Let's consider an example. A proposal to add a menu bar that's always visible should be vetoed reflexively, because the handheld screen is too small. But how about displaying the menu bar whenever the user taps the application's title tab? This improvement, part of the Palm OS since version 3.5, has no negative effect on the novice user, but adds flexibility for the power user. It also costs nothing in hardware resources.

Pragmatic Innovation

Achieving an optimal balance of features and convenience—that is, focusing on what matters—is the core of the Palm OS design philosophy. An ability to focus on what matters and provide a practical solution is what you need to create successful Palm OS products. One approach is the following:

- Identify the problems
- Find the simplest solution to each problem
- Get rid of everything else.

Innovate to make technology really useful

This is the approach taken by the team that created the original Palm Powered handheld device, and can be summarized by the phrase "pragmatic innovation." This vital part of the Palm OS culture simply means: do not use technology for technology's sake. Instead, try innovating. Make technology really useful. Limit yourself to a mature technology or to one that you can make work efficiently.

For example, Graffiti writing was the first form of handwriting recognition that was actually practical. A compelling technology such as handwriting recognition that is implemented poorly is useless and even destructive. You need to apply a new technology in a way that will satisfy the customer. If you can't, the technology (or your implementation of it) isn't ready yet.

Determining the Need

You get pragmatic innovation started by identifying the problems that your handheld product needs to solve. Determine what the users of your product need to do.

To accomplish this, you must sometimes question what customers ask for! This doesn't mean to overrule the customer. It does mean that you may have to refine your questions. Return to your representative users with prototypes that realize the trade-offs implicit in what they originally asked for. Given technological limitations, propose alternate solutions and see how the users react. It is your job to meet the user's need—today, using available technology, innovating wherever possible, but with on eye on the clock. The customer is waiting.

Find the real problems behind what customers ask for

For example, when Graffiti writing was developed, customers were asking for natural handwriting recognition. But natural handwriting recognition required a much faster processor and plenty of 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 designers recognized that the real problem was how to enter text. This problem could be solved without natural recognition. They solved the text entry problem with a simple writing method that works efficiently yet is fast, accurate, easy to

use, and doesn't take too long to learn. In addition, Palm OS applications were designed to require minimal handwriting. Other input methods that require only tapping were more efficient than any kind of handwriting recognition.

Similarly, when Palm developed its first handheld, customers asked for two-way wireless e-mail built-in. The technology available at the time would have significantly increased the cost and size of the handheld, pushing it right out of the sweet spot. Palm decided that this problem could be solved by developing seamless connection to a PC. Time and money went into streamlining and perfecting the synchronization process. The result was one-button synchronization.

In both cases, the solution was one appropriate to the moment. As time passed, PalmSource and the Palm Economy (that is, our licensees and our third-party developers) came up with even better solutions. Consider the case of Graffiti. First, a third party released a software alternative to Graffiti. Then came a hardware solution—the foldable keyboards. Most recent of all, are the mini "thumb" keyboards that snap into the serial port. Consider the case of two-way wireless e-mail. The first response came from a hardware manufacturer who produced a snap-on wireless modem for the Palm III. In 1999, Palm, Inc. introduced a handheld with wireless connectivity built in, the Palm VII. PalmSource continued to support Palm Powered modems by releasing the MIK (Mobile Internet Kit).

Answer the needs of the customer today. Then you or someone else in the Palm Economy can circle back and invent an even more sophisticated answer later.

Don't be afraid to respond to an answer with more questions.

The trouble with listening to customers too literally is that they do not always understand the limitations and consequences of the technologies they ask for. They don't realize that getting exactly what they ask for can be quite unpleasant.

If you're developing a custom application for a company that plunks down a huge checklist of features, go down the list feature by feature and find out why each one is there. Understand what the customer actually needs to do. Come up with appropriate and efficient solutions. Then present them to the customer.

Ignore conventional wisdom

Not only must you look beyond what customers ask for, you need to ignore a big part of what your competition does. Chances are good that they are probably following conventional wisdom. If you follow them, you will not, by definition, be innovating or differentiating.

As stated before, figure out what tasks your customers are trying to do. Those are the problems your product should address. In the case of Graffiti writing, the designers saw that natural recognition was a means rather than an end. What customers really needed was some kind of writing method that works well. Similarly, the designers concluded that the built-in applications had to be practical. If handheld applications had all the bells and whistles but were not practical, those "power" features would have no real value.

When you ignore convention, you must be ready to stand your ground. If your design is good and you have tested it with users to confirm that it works (as described under "Design Validation" on page 27), then don't let other people talk you out of it. Listen to qualified critics who make comments based on thoughtful observation. If they don't understand your design, try to learn why not and eliminate any sources of confusion. But watch out for people who are uncomfortable simply because your idea does not conform to conventional design. These people will confuse you and pull you back to where they are—conventionality.

Stay your course. Success paves the way for more innovation. The success of the initial Palm Powered devices gave birth to the Palm OS platform. In time, third-party developers came out with their own versions of the built-in software that made different ease-of-use trade-offs. By staying the course in the first place, PalmSource ultimately brought users a variety of solutions to their personal information management needs. Moreover, PalmSource enabled the creation of thousands of third-party applications, some of which do things we never even thought of!

The 80/20 Rule

You can forestall feature overload by applying the 80/20 rule: Focus on what users do 80 percent of the time and try to ignore the other 20 percent. Accommodate what most people need to do, but don't add complexity just to address fringe cases. Keep in mind that you

and most of your coworkers are not "most people." You use technology intensively, so you probably shouldn't apply the 80/20 rule to your own activities. Apply the rule to what typical users do.

You'll find that this approach will sometimes lead to solutions that are unusual or unorthodox. For example, using the Address Book button to beam a business card is architecturally odd. Palm OS software designers, however, applied the 80/20 rule and decided that one of the most important uses of infrared (IR) would be to send business cards. Then they came up with a way to make it happen just by pressing the Address Book button for two seconds.

Another example of the 80/20 rule: Pressing the Date Book button takes you to the current date. Why? Because 80 percent of the time, users want to see what they have scheduled for today. Additionally, the standard Date Book application has repeating events but doesn't cover every possible case of repetition. Thus, it accommodates what most people want to do but refuses to add complexity merely to include fringe cases. For example, to schedule an appointment for the second and fourth Thursday of every month, you must enter it twice in the Date Book (one repeating on the second Thursday and the other repeating on the fourth Thursday).

Focus on what users do 80 percent of the time

Your challenge is to take a hard look at the problems you are solving. Ask yourself what it is that people want to do with your application and how often. What do they want to do occasionally? What do they want to do every week? Every day? Several times a day? Employ the 80/20 rule.

Scaling the Problems

As you specify the problems that your Palm OS application is going to solve, it's important to scale them to the handheld world. In the PC world, designing a robust architecture that supports all conceivable cases is desirable and sometimes required. In the handheld world, you do not have the luxury of covering everything. Overly broad solutions actually shortchange the user! You need to design a handheld product as you would prepare for a backpacking trip, not for a car camping trip. If you're driving somewhere, you toss in extra shoes, canned food, beverages, whatever you please. You don't really think much about it. If you're backpacking,

however, every ounce matters and every item you bring needs to really deserve to be there. For example, a six-pack of soda might be tasty, but would you really want to lug it around for five days?

Whether you're backpacking or designing a Palm OS product, you don't pare down from 100 percent. You start at zero and work your way up. Start with nothing, and add only essentials—one by one.

The Memo Pad is a good example of scaling problems for a handheld. It's designed for jotting down notes. It wasn't designed for creating lengthy, formatted documents, which is the kind of word processing people do on a PC. It had only one font at a time, only one text style, and no paragraph formatting ruler. The Memo Pad was designed for a small screen, limited memory, and the difficulty of text input on the go. Text processing on a Palm Powered handheld has come a long way since 1996—thanks in part to our third-party developers. The Memo Pad, however, still serves its original purpose: inputting short notes for future reference.

Decompose a large PC application into multiple handheld applications.

Sometimes you may want to decompose a large PC application into two or more smaller applications for the handheld. For instance, a large PC word processing application can accommodate short notes or longer business letters. On a handheld, however, the customer is better off using several smaller, more focused applications. For instance, she should use the Memo Pad for jotting down brief information for handy reference. On the other hand, to write business letters, she should download one of the many fine word processing applications produced by members of the Palm Economy.

Sharing the Work

Your handheld application doesn't have to solve all the problems itself. You can assume most handheld users have PCs as well, and you can let the PC do your handheld application's heavy work. This means you can scale down your handheld application by shifting some of the work to a companion PC application. If you were designing a car, you wouldn't have to put a large refrigerator in the car. You could safely assume that the owner has a refrigerator at home. If you needed to be sure that the owner can keep the

occasional beverage cool on a long drive, you might just throw in an small ice chest.

Let a companion PC application do the heavy work

Work sharing is exactly what the PalmSource Expense application does. This application is designed for quickly jotting down expenses such as "Taxicab $6". The Expense application does not print expense reports, do exchange-rate conversion calculations, or even show a running total of expenses. It purposely has a very simple interface so the user can quickly note an expense. After synchronizing with a PC, the user can get a running total, currency conversions, printed reports, and so forth from a PC spreadsheet or database application. The point is that your handheld application does not have to take the place of an entire PC application. You can design your handheld application as a satellite to a desktop application. Think of a handheld as a device for accessing and managing content, and think of a PC as a device for data-processing and creation of large amounts of content.

Solutions—not Features

After identifying the problems that your product is going to deal with, you need to find solutions. There are two approaches. The wrong approach is to list a bunch of features that an application like yours should have and then try to implement as many as possible. For example, how can we make the PC feature set work on the Palm Powered handheld? Avoid questions like that.

The right approach is to figure out how your application can accomplish precisely what the user needs to do. Look for solutions that are fast and easy to use. Try to delight the user.

Minimize clutter

One way to make your application easy to use is to minimize clutter. To some extent, this will be a natural outcome of reducing your application's feature set. Beyond that, look for ways to organize your screen layouts so that they aren't overloaded with objects. With fewer objects on the screen, users can more easily focus on the remaining objects. Naturally, you'll want those remaining objects to represent important features. Thus, minimizing clutter makes important features more accessible and ultimately makes the application easier to understand and use.

Reduce the step count for common tasks

You can also make your application faster and easier to use by minimizing the number of steps for frequently used features. While keeping in mind the goal of minimizing clutter, try to organize your screen layouts so that the more commonly used a feature is, the fewer actions are required to access it. If you can make the most common functions accessible by just one touch or one tap, people will love your application.

Conceal risky functions

There are some functions that you might not want to make easily accessible even though they are commonly used. For example, users must perform several steps to delete records in the Address Book and Date Book applications (unless they use a shortcut stroke). This is an intentional safeguard against accidental deletions. Don't make a function too easily accessible if it can destroy data or is dangerous in some other way.

Note that although the Delete function is hidden, the New function is as easy to access as possible. Symmetry is not always desirable in a user interface. At work, you probably have a stapler and a staple remover. They perform symmetrical functions, yet it makes perfect sense to keep the stapler on your desk and the staple remover in a drawer.

Include power features discreetly

In addition, you may want to include "power" features for advanced users of your product. There's nothing wrong with power features as long as they don't get in the way and trip up novice users. For example, if you want to create a new event in the standard Date Book application, you can just start to write the time or the title of the event. Just write a 3 to create an event that starts at 3:00. That method is faster than tapping New, and it's there for users who want to learn about it. Yet it adds no clutter to the user interface and will never get in the way of the novice user. What's great about power features is that they don't have to be obvious. They can require a shortcut stroke or some other "secret handshake." Power users by definition are willing to dig to find out all the tricks.

Intuitive

Another important part of making your application easy to use is making it easy for people to discover its features and figure out how to utilize them. To make your application feel intuitive, keep the

user interface consistent with built-in applications. Adhere to the Palm OS user interface guidelines. A consistent user interface is a familiar user interface.

An application that feels familiar feels easy to use even on first experience. When users first look at your application, they will explore it, and they will try to figure it out. Every little thing that they have to think through is like a little speed bump. Each time a user encounters something unusual in your application's user interface, something that works differently from other applications, the user must take a couple of extra seconds to figure it out. (Or the user might not figure it out at all.) These disturbances have a negative cumulative effect.

Stick to the user interface guidelines

Your application gets a positive cumulative effect if it sticks to the user interface guidelines. It leverages everything the user has learned about the standard user interface from other applications.

For example, the user recognizes a command button and immediately knows how it works. The user taps the button, and it behaves exactly as expected. This button adds nothing to your application's learning curve.

If you can limit the number of speed bumps, users will get through your application and think, Wow! That was really easy to figure out. They just picked it up, started playing around and got it to work.

Easy to Remember

Even more important than making procedures intuitive is making them easy to remember. A user who can't remember the procedure for accomplishing a particular task must repeatedly rediscover how to accomplish the same task.

Usually a procedure that's easy to remember is one that's also easy to figure out in the first place, but not always. Certain procedures may not be obvious to some users, especially novices, who might have to be taught the more obscure techniques. Nevertheless, these techniques can still be easy to remember. For instance, it's not obvious that you can beam a business card by holding down the Address Book button, but the technique is easy to remember. As one person said in a Palm OS user test, "It's really intuitive, once you

figure it out." If you can't make a procedure universally intuitive, make it memorable.

Example: Train Catcher

Let's take a look at an example application that illustrates some of the design principles and practices we've discussed. This is the imaginary Train Catcher application, which you might find useful if you were traveling by subway in Japan. When you are transferring trains and there are six different trains leaving from four tracks, Train Catcher helps you figure out which train to take. Figure 1.5 shows a first design for this application.

Figure 1.5 Example application design, first pass

At the top of the screen, three pop-up lists let you specify which train line you want, your starting point, and your destination. Below that you specify the time of the earliest train you want to take; this would be the current time by default. The bottom half of the screen presents some options: you can select the weekday schedule or the weekend schedule, and you can set how the list of trains will be sorted. To use this application, you would configure all these settings, tap a button, and the application would generate a list of all the trains you could take.

This first pass at the design is very busy. There's a lot of stuff on the screen, and your eye wanders all over the place. You notice the icons at the bottom of the screen, but you don't really know what they do. In fact, some time after the designer of this example application put

those icons there, even he wasn't sure what they were intended to represent.

It's not so unusual for the designers of an application to forget why they included some features or how a feature is supposed to work. When that happens to you, it should raise a huge red flag. If you're knee-deep in the design and you can't remember how it works, then you certainly can't expect an average user to have an intuitive feel for it or figure it out.

This design needs a second pass to clean it up a little. Let's do a layout that's more like the standard Palm OS layouts, as shown in Figure 1.6.

Figure 1.6 Example application design, second pass

Let's line up the three pop-ups at the top of the screen. Let's add a couple more words to clarify the purpose of the time field. We can eliminate the weekday-weekend indicator. Generally, the user will be searching for trains that are running today. Your application can easily find out what day it is, and show the appropriate weekday or weekend information.

For customization purposes, we might put the weekday-weekend setting on a preferences screen. The sorting option can definitely move to the preferences screen because it probably will not change very often. Since we couldn't remember what the icons do, let's get rid of them. One of the icons probably started the search for trains, which is the application's most important function, so let's make a text button for that function.

We now have room to add a button for another useful function that shows when the last train leaves. In a city where some trains don't run 24 hours a day, this button tells you how late you can stay without missing the last train and having to take a taxi home. This new button does not clutter the layout, because we removed nonessential objects. This last-train function is exactly the type of feature you will add if you are focused on solving the user's problems, as opposed to padding the design with incidental features such as a weekday-weekend indicator. Users will say, "Great! That's just what I wanted to do."

This second pass could still be improved. For example, the field label "Display trains after" could be rewritten less ambiguously. It might say: "Show trains that leave after:" Your label would be a little longer, but also clearer. And you provided the extra space required by simplifying the original user interface.

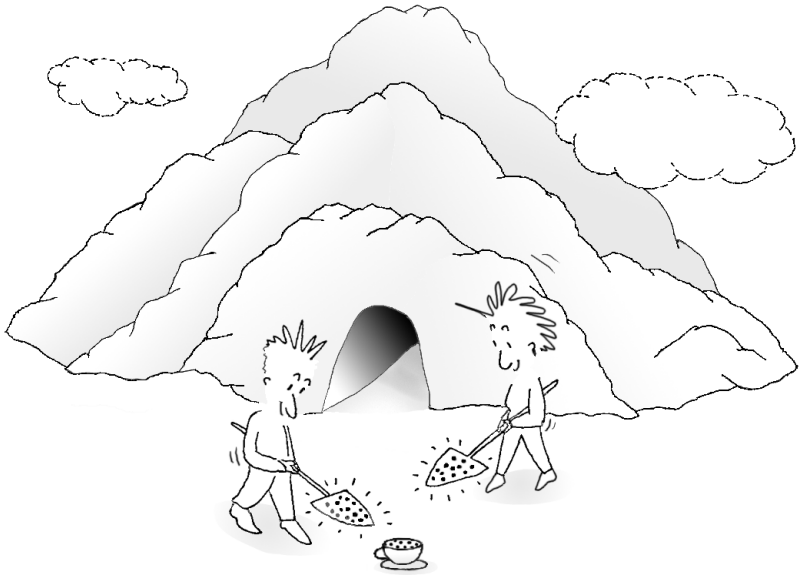
Application design is an iterative process, like editing written material. In the first pass you just throw everything on the page. Then you hone it down. You show the second pass to people and get feedback, and you realize it's still not good enough. Then you get user feedback on several more passes. Iterative design is what really makes a great application, rather than just giving it a shot and calling it a day.

Taking the editing analogy one step further, designing a handheld application is more like editing a poem than editing a novel. In a novel, you're not very concerned with size. In a poem, every single word and every punctuation mark has its place. A 17-syllable Haiku poem is more exacting to write than a 600-page novel.

Solution to Riddle #2

Now we're ready to answer the second riddle. We know a mountain of features won't fit into a teacup. We want the optimal balance of features and user experience, and know one way to achieve this is through pragmatic innovation. That is, identify the problems, find the simplest solution to each problem, and get rid of everything else.

Q: How do you fit a mountain in a tea cup?



A: Extract the diamonds and leave the rest. (Do you really want the rocks and dirt?)

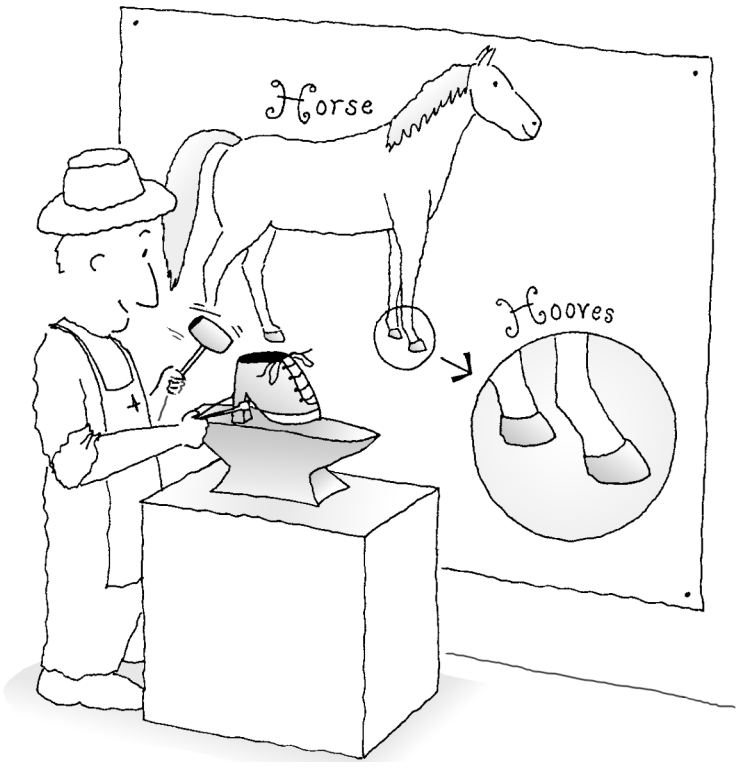
Make a product that does what people need, not necessarily what they ask for right off the bat. Make it useful and elegant: minimize clutter, reduce the step count for common tasks, conceal risky functions, and include power features discreetly.

Scale your product for the handheld world. Let a companion PC application do the heavy work. Use the 80/20 rule to hone in on the most frequently used features and jettison the rest. That takes a leap of faith, because it's possible to get rid of the wrong stuff. You might end up stripping out a lot of features and finding that some of them were essential after all. It's challenging to figure out just which features are important, keep those, and leave everything else out. But if you do that, you'll create a far more compelling application.

Design Validation

The third riddle along our path to enlightenment is concerned with finding out the quality of your design in real-world circumstances:

Riddle #3 **Q: How does the blacksmith learn to make the perfect horseshoe?**



Hint: Humility lights the path to knowledge.

This riddle has to do with validating the design of your product's user interface. Every book ever written on the subject of user interface design stresses the importance of user testing. This riddle, however, suggests that certain approaches to user testing are in keeping with the Zen of Palm—approaches taken by PalmSource, Inc. and its licensees in designing handhelds and handheld applications.

Validating Design Quality

If you're skeptical that user interface testing is really necessary, think about how different your customers are from you and the other designers and engineers you work with. Being part of the computer world, you all naturally use computers in the course of your jobs. You aren't threatened by computers, whether of the desktop or handheld variety. You enjoy working with them and understand them. Your familiarity with computers makes it difficult to relate to people from nontechnical worlds. But computers, including handhelds, can be totally alien to somebody whose primary job is to know real estate or to manage a pet store. Nontechnical people might not feel comfortable with computers and might not like using them.

To get a feel for what it's like being uncomfortable with computers, try to imagine a world where a new, more efficient means of writing computer programs requires that you know French literature or philosophy. Imagine that you're trying to compile your application and you get an error message that says, "Epiphenomenal event. Retry using Cartesian Dualism." You think, I'm an engineer, not a philosopher! You'd resist using this compiler in the future.

Basic User Testing

Because so many people are uncomfortable with computers and computer products, it's vital that you get their feedback in user testing. It doesn't have to be expensive. It doesn't have to be particularly difficult. You can get useful results from simple user testing even if you're an experienced designer.

Remember that your emphasis is on actual users rather than theoretical exhaustiveness. So you should test that your application is useful in real world situations. Keep in mind the following criteria:

- Does the application accomplish a complete, if limited, task?
- Can the user figure out what the application does from its name and, especially, from its user interface?
- Does your user interface imply all sorts of tempting functionality that is not actually implemented in your

application? (Perhaps a redesign is indicated to avoid user confusion and disappointment.)

- How long does it take a user to figure out the full task?
- How long does it take the user to remember how to perform a task that he or she previously figured out?

Let's explore some ideas for doing simple user testing. The most important idea is that lots of informal testing is better than a little or no formal testing. You don't need to have two-way mirrors and hire a consulting company. If you're an individual developer or a small company, basic testing might be all you can do. If your company has the resources to do comprehensive testing, then basic testing can be the first phase of a comprehensive test plan.

Test early

Whether simple user testing is all you can do or is part of a larger plan, start testing early. Don't make the mistake of waiting until your application is in alpha to begin testing the user interface design. If you think that you should first get something running and then start user testing, you may find yourself trapped. At that point you've invested a lot of work into getting something running, only to discover in user testing that the design is all wrong. Are you going to scrap the engineering work that's already done? It's probably too late to do that, so you have to compromise the design.

Test with simulations

To get an early start, begin user testing of your design with simulated screen shots made with pencil and index cards, HTML pages, Macromedia Director, Adobe Illustrator, Satellite Forms, NS Basic, Visual C++, Metrowerks Constructor, or whatever you like. Don't wait for alpha.

Test selectively

Your early tests don't have to cover your entire user interface. You can test selectively. Start by identifying the specific parts of your user interface that you must try to get perfect, and test those. In addition, test any parts of your user interface that you have doubts about, even if those parts are not essential. These are the approaches that designers took to test the user interface of the original Palm Powered handheld when Palm, Inc. was a small company. An individual developer or small company that can't test an entire user interface can selectively test the most important parts.

Of course, you should not stop with informal, selective user testing if you can formally test the entire user interface. Formal testing with full documentation gives you data on which to base your convictions, leaps of faith, and uses of the 80/20 rule.

Test outsiders, and be methodical

Your test subjects can be anyone who isn't knee-deep in the details of the product—your receptionist, someone's brother-in-law, even people off the street. Bring them in, sit them down, and describe a task you'd like them to perform with your application. Have them try to accomplish the task, and ask them to talk about what they're doing as they proceed. You simply watch and listen. See where people get into trouble.

While you watch, take notes. Orderly documentation will help you accurately evaluate your tests and correctly determine what changes to make.

Try to be consistent in how you set up and carry out the same test with all testers. Give each person the same description of each test. Ask the same questions. Keep a record of each tester's responses and actions. By setting up and going through each test the same way with everyone, you make your results more meaningful.

Test iteratively

As you find trouble spots in your design, tweak it to eliminate or reduce them. Then test some more. Test, tweak, repeat. Refine your design before you get too far along in coding. Iterative user testing was a critical factor for the success of the original Palm Powered handheld. The software designers actually did several prototypes of the entire user interface in HyperCard before writing code.

What You Find

When you test your user interface design, you'll be surprised at the parts of your design that users can't figure out. These parts of the design are incredibly obvious to you because you see them in the context of the whole application. You see how all the details fit into the big picture. Users look at individual features or procedures without that context and simply have no idea what's going on.

You need to know if users can't figure out where they're supposed to touch the screen or what to enter. For example, PalmSource, Inc.'s Network HotSync® software, which enables network and remote

synchronization, has a screen that says "Enter your user ID and password." How much more obvious could that possibly be? Well, this stumped one tester, who wanted to know whether User ID meant his handheld user name, his network user ID, or his e-mail user ID. The designers knew which user ID to enter because they knew the context, but the input request was ambiguous to the user.

Use plain language

User testing also helps flush out terminology that need to be replaced with plain language. We get so used to our own jargon that we forget. To avoid using jargon, think what you would say to a person with little computer experience. For instance, if you're explaining how to use the Find function, you wouldn't say, "Enter your search string into the text field of the Find modal dialog." You would say, "Write the word you're looking for on the line."

To clean up jargon, ambiguities, and other language problems, have a documentation writer, copy editor, or at least a literate friend go through your application's dialogs. It's a good investment of time.

Keep the course of action clear

User testing will reveal still more design flaws, including the "left turn at Albuquerque" syndrome. This refers to the old Bugs Bunny cartoons in which you'd see Bugs digging a long tunnel toward Miami Beach, but he would pop his head up in some strange place and say, "I knew I should have taken that left turn at Albuquerque." Similarly, a user who is trying to figure out how to accomplish a task will start building a mental model of your application. (This is the digging a long tunnel part.) If the user takes a step in the wrong direction early on and heads off in the wrong direction, building a false model on top of that misstep, then eventually the user gets very confused.

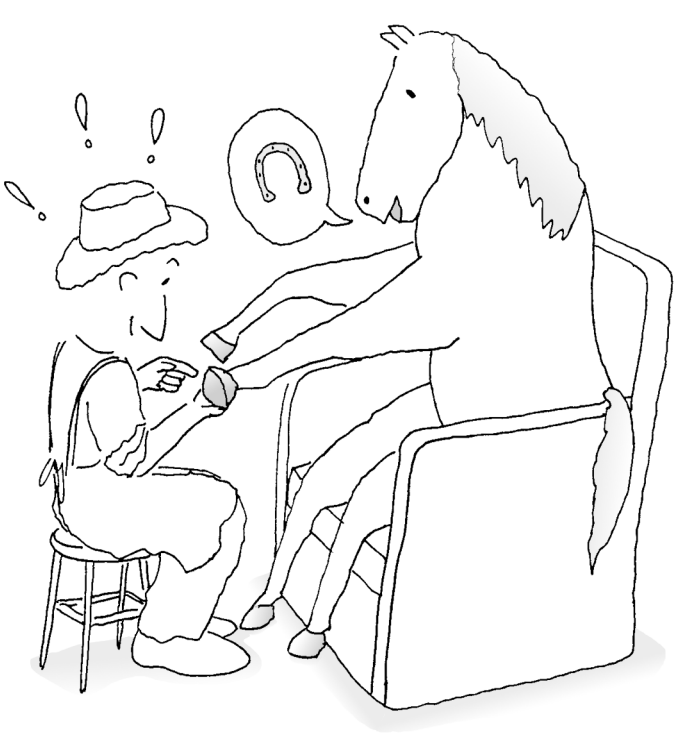
When people are experimenting with your application, trying to figure out how it works, and are faced with several options or courses of action, they may use a process of elimination to decide which alternative to choose. First they rule out the least likely choice, then the next least likely choice, and so forth until only one remains. It's the same method people use to take multiple-choice tests. This approach works better when your application design is simple and uncluttered. Too many choices make it hard to figure out which is the right one. Subsequently the user will also have trouble remembering which of those many choices was correct. If there are

only a couple of choices, the user can more easily figure out and remember which is right.

Solution to Riddle #3

Now we're ready to answer the third riddle.

Q: How does the blacksmith learn to make the perfect horseshoe?



A: Straight from the horse's mouth.

The expression "straight from the horse's mouth" derives from the ancient art of horse-dealing. A prospective buyer can verify suspicious claims about a horse's youth by looking into the creature's mouth and checking its teeth.

In the case of our riddle, "straight from the horse's mouth" has a double significance. On the one hand, you should listen to your users. Like the talkative horse in the cartoon, they know a lot about their own needs, and you need to learn from them. On the other hand, you should also look and learn for yourself. Observe the

horse. See how he walks. Observe different surfaces. What gives him trouble? Uneven surfaces? Pavement? What could help most? Likewise, your users are well worth observing. How do they approach your application? How do your innovations fit their expectations and computer-using habits?

A blacksmith can improve his horseshoes by trying them on horses, observing the fit, and reworking them for better fit. You can improve your product by testing its user interface design, observing where people have trouble with it, and revising the design to eliminate the trouble. This may seem obvious, but it's tough to take the necessary time when you're rushing to get the product done.

Start testing early with simulations. Try to test outsiders and be methodical. If you can't test everything, test the critical parts and the parts you're not confident about. Keep testing as your product evolves. The effort you put into user testing will pay you back several times over by increasing your product's long-term usability, reducing the cost of customer support, and delighting your customers.

Design Improvements

The final riddle concerns improving your designs.

Riddle #4 **Q: How do you improve perfection?**

This riddle has no picture. Perfection is an ideal, so you'll have to imagine it in your own head.

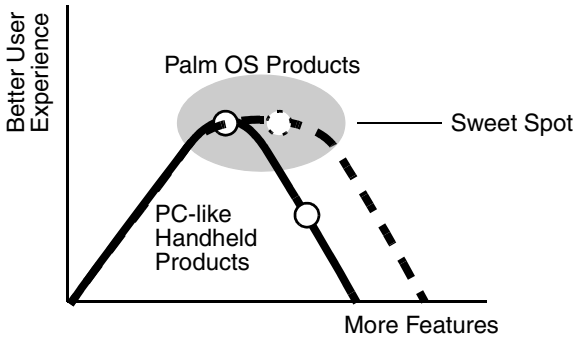
Hint: What is a "more perfect union?"

We have now entered the advanced levels of enlightenment. Being easy-to-learn and easy-to-use is great, but your product must continue to improve in order to keep ahead of its competition. This presents a paradox. If less is more, then in theory, once your product has reached a certain level of elegant focus, once it has nailed that sweet spot, you're done. No more product advancement.

Stretching the Sweet Spot

So how do you keep your product moving without moving it out of the sweet spot? Try stretching the sweet spot, as shown in Figure 1.7.

Figure 1.7 How to add features over time



Pragmatic innovation lets you *shift the curve* and make the sweet spot larger over time. One example of this is infrared (IR). The original Palm Pilot models did not have IR because it was technically very difficult to incorporate in a way that was truly easy to use and didn't drain the batteries rapidly. The user benefits of IR didn't justify the expense of including it. Later, PalmSource engineers invested a great deal of effort to integrate IR elegantly into the Palm OS with a user interface that delighted customers. PalmSource shifted the curve and made the sweet spot larger.

Here's another example of stretching the sweet spot over time. When people asked for a handheld web browser, PalmSource, Inc. product designers realized that what users really needed was to get information from the Internet. PalmSource's initial solution to this problem, web clipping, was another pragmatic innovation that added features without sacrificing user experience. Rather than trying to simulate the experience of using a web browser, web clippings were designed to provide information from the Internet in amounts that a handheld can manage. This lightweight format was ideal for the narrow pipeline of wireless connections. But this balancing act is not meant to last forever. As wireless technology improves, PalmSource's web browser becomes more sophisticated, displaying as much of the web page as the practicalities of the moment permit. Meanwhile, for handhelds connected to the

Internet via a wireline modem, there are third-party web browsers available. In short, the Palm Economy offers a balance of solutions that will re-balance as the surrounding technology changes.

Add features creatively

Note that it's not enough merely to add a feature. The feature must be added creatively and elegantly or the user experience deteriorates and the product moves out of the sweet spot. Products that compete with Palm OS handhelds aren't in the sweet spot even though they have many more features, because too many features lead to a complex user interface. Palm OS products can actually match or surpass the competition's functionality and stay in the sweet spot.

Discovering New Features

You can come up with new ways to stretch the sweet spot without spending lots of money on advanced research and development. There is plenty of opportunity in newer technologies that are already in the market but are badly implemented and too expensive at the present time. You just need to track these technologies and understand their limitations.

Study other products

If you see a competitive product that has a poorly implemented feature, buy the product and try it out. Carry it around and try to determine if the feature has potential and what is preventing it from being a great feature now.

While evaluating a potential new feature, decide whether either or both of the following conditions will enable you to add it to your product:

- The feature uses a technology that has evolved to become less demanding over time. Technologies that were too bulky, too expensive, or too power hungry to fit your sweet spot last year may be less so this year.
- You have a pragmatic and innovative idea for using the new feature in a way that no one has done before. Pragmatic innovation is the key to making a cumbersome technology elegant, thereby keeping it in the ease-of-use sweet spot.

Stretching the sweet spot to improve your handheld product is an advanced design technique. As you might expect, this advanced

technique is more difficult to put into practice than the basic design methods that you already used to get your product into the sweet spot in the first place. Once you let go of PC thinking, it's much easier to apply the 80/20 rule and keep only features that users really need than it is to find new features and make them fit elegantly in the sweet spot. In handheld product design as in other skills, gain experience with basic methods before trying advanced techniques.

Solution to Riddle #4

Now it is time to answer the fourth riddle.

Q: How do you improve perfection?

A: Are you kidding? Perfection is a balancing act. Change the environment, and you have to re-adjust the balance.

In technology, perfection is strictly for the moment. Your "perfect" solution balances available technologies, costs, market forces, and user expectations. Whenever the surrounding forces shift, your balancing act needs to be re-balanced. Put another way: "perfection" is dynamic, not static.

The founders of the United States government understood this paradox, as reflected in the famous phrase from the preamble of the Constitution: "to form a *more perfect* union." High school English teachers often insist that the phrase should say: "to form a *more nearly perfect* union." Something perfect cannot, by definition, be made better.

Now, the writers of the Constitution understood the rules of English grammar, but they also understood a deeper psychological truth: There are stages along an evolutionary development where the balance appears perfect. As conditions change, the "perfection" has to change too. In the late 18th century, the fact that 13 bickering American states could form any union was a miracle. A few years later, the U.S. Constitution sought to improve on that.

Improvements in the Palm Powered world follow a similar "incremental perfection." Study a problem, innovate within the current limitations, release your product. Then enjoy the success, take a deep breath, and start the cycle all over again.

Earlier in this booklet, we gave you a couple of examples of how a particular solution metamorphosed over time in response to external changes. The principles of practical innovation and incremental perfection underlie both the user input technology solutions (Graffiti 2, keyboards, etc.) and the web browsers that the Palm Economy has brought out over the last few years.

The history of telephone technology demonstrates this dynamism.

While these principles are key to the success of the Palm OS platform, they are not unique to it. Several successful technologies have followed the same path. Let's consider, for instance, a ubiquitous technology that has been around so long we can easily overlook its many transformations: the telephone.

When the first telephones came out, a clear problem emerged: How does the caller contact the desired party. Each generation of telephone technology has answered this question differently, according to the resources and user expectations of its time.

In the earliest telephone networks, callers spoke directly to a human telephone operator and asked for their party by name. This quickly became impractical, so each telephone was assigned a number. Users were ready for this new paradigm: telephones were identified by numbers just as buildings were. In the next generation, automatic dialing entered the picture. Using a dialer, the caller bypassed the telephone operator altogether. By that point, users were sophisticated enough to accept a new input technique that brought the benefits of faster—and anonymous—connection.

Most recently, cellular phones have altered the experience again: the user views a list of names, selects one, and the phone dials the number automatically. In a sense, telephone technology has now circled back to its origins. Once again, focus is on the person to whom you wish to speak, not on the number.

And in the not-too-distant future, you can imagine each person receiving a universally unique communication number (at birth?). Associated with a person rather than a location, the number makes a person reachable on virtually any telephone device. The telephone number becomes a permanent and unique identifier and would probably be concealed behind the person's name. (We might also hope for appropriate call blocks and filters.)

At PalmSource we see it like this: if a technology as venerable as the telephone has to continually transform itself, no company or industry can afford to sit still technologically.

The same dynamism is at work in the Palm Economy.

The Palm Economy—PalmSource, Inc., our licensees, and third-party developers—thrives on just such a dynamic balance. To meet our customer's needs, we offer today's best. Almost immediately, an improvement becomes possible, and that improvement is brought to market, as well. The Palm Economy is smarter than any one company—or any two or three companies, for that matter. It leverages the intelligence, determination, and ingenuity of several of the industry's smartest big players, as well as the intelligence, determination, and ingenuity of *thousands* of the industry's smartest small players!

Summary: The Zen Approach

Before we go, let's review a minute. To design great Palm OS products you must set aside the instincts that you may have learned in the PC world. Avoid the siren call of "features for features' sake." It will lead you down the path of suffering and small market share. Instead, focus on the user's experience with your product. Convenience and usability are power.

Most importantly, focus on the inner tranquility of the customer. Do you swear at your computer? (If not, you must know plenty of people who do.) Machine hangs. "*&#*#!!!" Lost data. "*@#$(@#!!" Network down. "&#@^$*#?!" Have you ever noticed you don't feel that way with Palm OS products? Want to see your schedule? Press a button and there it is. You're in control. You don't wait. You don't get confused or frustrated. It's all very elegant and pleasing.

You must prevent your products from becoming complex and frustrating. Yet you must continue to innovate to differentiate your product from the competition. Add more, but only if you sweat the details, focus on solutions, and keep it easy-to-use.

Remember that your goal is not to satisfy some marketing team's check list of features. Your goal is a creative and challenging one. It is to serve your customers while preserving their inner tranquility—the Zen of Palm.