200094_CH01/Ritter 9/12/01 4:20 PM Page 1

Chapter 1

The Case for a Superior GUI

On the surface, designing the *graphic user interface* (GUI) for your LabVIEW application appears to be relatively straightforward. Just choose the appropriate controls and indicators from the Controls palette; drop them on the panel; align, rearrange, and tweak them into a presentable arrangement; and then wire the underlying diagram. Simple enough. So why devote an entire book to this seemingly trivial task?

In a word: *usability*. Consider this scenario: You have just put the finishing touches on your latest LabVIEW application, and now it is time to unveil it to the world. You sit the first unsuspecting "guinea pig" down in front of your program. (Maybe the user for this maiden voyage is a friend or colleague, or perhaps you are not so lucky, and it is your boss or, worse, a customer.) Anyway, you launch the application and step back in anticipation.

Hoping for the best, but prepared for anything, you scan the expression of this first unsuspecting user. You, of course, are looking for some small sign of approval —after all, you have traded several hours of your very existence to bring this bouncing baby application into the world, and as any mother can tell you, the labor of creation is not an entirely painless pursuit. You watch in anticipation.

Then it starts. Perhaps the first sign is a slightly furrowed brow. Or maybe it's the eyes, repeatedly traversing the surface of the computer screen, looking for a landmark—something, anything familiar. Or maybe it's that unmistakable look of a lost child. Those first telling signs can show up in so many different ways.

For a brief, shining moment, self-deception becomes your refuge. "After all," you reason, "looks can be deceiving." This false sense of security is short-lived, however, and soon, the inevitable—"the question." The specific wording of the question is never important; it can take many forms. What cannot be ignored, however, is the underlying reality it reveals: This person does not have a clue how to navigate your user interface! Your mind reels. How is this possible? The concept is so completely natural, the underlying metaphor so elegantly conceived and realized, that it is universally intuitive. Or at least, so it seemed to you.

Concluding (1) that this would-be user must have somehow "duped" the entire education system in order to graduate from high school and (2) that your clever interaction mechanisms are far too sophisticated for the likes of someone so under-endowed in the IQ department, you seize control of the mouse and begin to demonstrate. With a puzzled, glassy-eyed stare, this poor, bewildered soul nods obediently as you expound the virtues of your masterpiece. However, while you are deftly demonstrating the finer points of your design, you cannot help but notice an undeniable sensation developing in the pit of your stomach. Your message is not getting through, and you know it. You start to wonder if maybe, just maybe—and in some very, very small way—your design is somehow just a little bit responsible for that vacant stare.

If you have felt the cold sting of user confusion, some parts of this account may resonate for you. Scenes like this happen to almost every user-interface designer at one time or another. Even professional designers with formal training and years of experience have been known to miss the mark from time to time. In fact, several high-profile examples have been exposed in vivid Technicolor detail by the reviewers for the Internet-based "Interface Hall of Shame" (**www.iarchitect.com/shame.htm**). If you have ever made an interface design "boo-boo" in the past, at least you can be thankful that these people did not decide to review it!

On the other hand, maybe you are one of the lucky ones, and this somewhat disconcerting and potentially embarrassing situation has never happened to you. Congratulations. With the help of the information contained herein, maybe it never will. Armed with some of the key underlying principles and philosophies of GUI design, as well as some practical tips and techniques, you will be better prepared to identify the potential pitfalls and steer clear of the potholes on the road to a better LabVIEW user interface.

Before we plunge in and start laying the foundation for the topics to come, it might be a good idea to start by looking at the rewards you stand to reap as a direct result of devoting additional time and effort to your GUI. Consider the points that follow to be the "carrots" that will spur you on toward user-interface design excellence!

1.1 Motivations for the Superior GUI

A well-designed, thoughtfully executed GUI not only will add quality to your application but also will provide several advantages in the areas of usability and marketability. We will look at several specific points shortly, but first, let's set the stage.

1.1.1 The Point of Contact

LabVIEW programs—and, in fact, all computer programs—share a single common purpose: to help people achieve goals. Without a real-world problem looking for a solution, what reason would any of us have to develop a computer program? Sure, wiring a LabVIEW diagram can be enjoyable—at times almost even cathartic—but there are better ways to pass a Saturday afternoon than building software that no one will ever use.

You may not have considered it from this perspective before, but your GUI is the only point of contact end users will ever have with your LabVIEW application. In other words, a user's only view of your application—from meticulous planning to *input-output* (I/O) hardware selection, from the painstaking attention to data structures and data flow through careful wiring of the diagram—in short, the culmination of all your hours of planning and hard work—is ultimately communicated completely through the controls, indicators, graphics, and layout of your application's GUI. Your application's usefulness ultimately lies not only in the integrity of the VI diagrams but also in the clarity and effectiveness of the GUI and, of course, your end user's ability to pilot it.

For better or worse, your user's perception of quality will be shaped not only by the "look and feel" of the interface but also by the ease with which the intended application goals can be achieved. Hours of painstaking effort devoted to the fine detail of the front panels will go unappreciated if the user cannot navigate the interface effectively. And just as surely as a well-executed GUI design will inspire confidence and trust in your application, a bad design ultimately will lead to end-user confusion and frustration.

Unquestionably, then, GUI design and integration are primary elements that will have a direct impact on the overall quality and, by extension, success of your application. Still, despite this importance, most LabVIEW GUIs receive an inordinately small allocation of resources. Why is this so? The answer appears to lie in the assignment of priorities.

When engineers begin to develop an application to solve a difficult technical problem, naturally, the integrity of the underlying logic receives the highest priority. GUI planning and integration are assigned a lower priority because, simply stated, if the software is technically unable to solve the real-world problem for which it was designed, even the most meticulously crafted GUI in the world will not save it. While the need for integrity in the underlying logic is unquestionable, there is another side to this coin that must not be overlooked. If human operators are unable to run the software effectively—and achieve the expected results—the best algorithms, implementations, and VI wiring in the world will not save it either!

According to Dick Berry, a senior member of IBM's Ease of Use Design Group, a useful analogy can be found with the motion picture industry. "A great film can't happen with only a good script; it requires great actors, great directors, great cameramen, and so on to communicate the script in a manner that is truly compelling to the audience" (Berry, 2000).

At this point you may be thinking, "Sure, I understand the value of the GUI. That's precisely why I use LabVIEW. The LabVIEW system takes care of the GUI so that I can focus on the more important, underlying problems." While it is true that LabVIEW lightens the GUI programming load, it must not be forgotten that most of the advantages appear during execution—in the drawing and management of on-screen GUI objects. The critical areas of layout, design, and interaction modeling are still up to you. Clearly, the GUI objects on the Controls palette are the building blocks for your GUI, just as the functions on the Functions palette are the building blocks for your VI diagrams. It follows, then, that just as a well-designed VI diagram depends on planning and the careful application of functions and sub-VIs, a well-designed GUI depends on planning and the careful selection and application of controls and indicators.

If you think about it, a well-crafted GUI adds an additional challenge—not only must the front-panel elements be considered carefully, and arranged artfully, but the diagram logic also must be developed to support the interaction model. Without the requisite attention to detail, it is just as easy to build a confusing and ineffective GUI in LabVIEW as it is in any other programming language. On the other hand, it is much easier to implement a great GUI using LabVIEW, provided you attend to the preliminaries. Armed with a clearly defined interaction model and an effective design plan, building a solid LabVIEW GUI is not only easy but also enjoyable.

1.1.2 Benefits of a Well-Executed GUI

Now that the preliminaries are out of the way, here (as promised) are some of the direct benefits of enhanced GUI design:

- Improved usability leads to higher user productivity and lower long-term costs.
- Planning required for a complete, effective GUI improves the overall integrity of your application.
- A clean GUI, with clear and obvious methods of interaction, improves the reliability and safety of mission-critical applications.
- Users often perceive the level of developer commitment based on GUI "look and feel." A well-designed GUI translates to improved user confidence.
- Improvements to visual presentation and usability make software more marketable.
- A clear, attractive GUI design can open the lines of communication.

Bold statements, you say? Let's have a closer look at these points.

Improved Usability Leads to Higher User Productivity and Lower Long-term Costs. When a GUI is planned effectively and integrated properly, a number of usability benefits are realized, including

- Shorter user learning curves and lower training costs
- Better user comprehension of system capabilities
- Better user acceptance of the application

- Higher levels of user confidence-both in the capabilities of the program and in the ability to use it
- Higher levels of user productivity-increased efficiency and fewer errors

The underlying theme here is user satisfaction. No one wants to work for several hours a day with a difficult, confusing, or frustrating computer program—life is too short for that! If an operator or end user experiences a sense of satisfaction using your program, then the program is more likely to be used. Managers are less likely to hear complaints and, therefore, are less likely to look for a replacement program (or programmer!), and experienced operators are less likely to look for work elsewhere. Lower operator turnover means reduced training costs—and more experienced operators. All of this adds up to higher productivity and lower costs. But wait, there's more.

Planning Required for a Complete, Effective GUI Improves the Overall Integrity of Your Application A clean, logical, streamlined user-interaction model does not just happen; like all critical aspects of your application, an effective GUI requires careful planning and consideration throughout all stages of the development process. The good news is that advanced planning for the GUI will force you to consider carefully your application's architecture and data structures from the point of view of GUI integration.

For example, if the goal of your application is to collect and present some useful data to the user (as is the case with the majority of LabVIEW programs), it only makes sense to design the data structures with some awareness as to how the data will be displayed. A data structure that is sympathetic to the needs of the GUI will make for a cleaner diagram; output data will not need as much unbundling, indexing, and reorganization for display. Also, if your application uses large arrays or large, complex data structures, attention to the GUI before embarking on the design phase can offer additional benefits. Planning for GUI output can prevent a proliferation of large-array copies. Fewer copies of large data sets means fewer calls to the memory manager and improved overall performance.

Attention to the GUI will force you to be conscious of the front-panel implications of every element you add to the diagram—both visually and logically. To keep the panels uncluttered and comprehensible, you may discover that large diagrams can be split into smaller components. As a bonus, you are sure to discover that some of these smaller components can be generalized, improving the possibility for code reuse.

In addition, if you consider the GUI before planning the overall architecture of your application, you will have an opportunity to develop a detailed plan for when (and how) sub-VIs are called. This is the time to verify the logic and consistency of your GUI design—before the actual coding phase starts. Even informal prototyping and usability testing will reveal which GUI items are essential and which are unnecessary. By eliminating the unnecessary and focusing on the crucial, you will streamline the development process and focus your efforts. The final by-product ultimately will be a higher-quality application, one that is easier to modify, upgrade, and maintain.

A Clean GUI with Clear and Obvious Methods of Interaction Improves the Reliability and Safety of Mission-Critical Applications. Mission-critical industrial applications put additional demands on the GUI. If an operator error could lead to a forced shutdown or, worse, a potentially hazardous situation, then the accuracy of operator actions is crucial. While any properly designed software control system destined for industrial uses necessarily will have hardware interlocks to prevent a catastrophic failure, interlocks should not be depended on to compensate for an inferior or confusing GUI design.

The advantage of a clean, direct GUI design becomes most obvious in times of crisis. While operators may learn to navigate a complex GUI layout under normal circumstances, consider what happens as a critical situation develops. As the operator begins to experience stress, the regular sequence of cool, reasoned actions gives way to a series of autonomic responses. If the GUI design is clear and the important emergency response functions are readily apparent, the chance of stress-related errors will be minimized. Improved system reliability and safety will be the direct result.

Users Often Perceive the Level of Developer Commitment Based on GUI "look and feel," and a Well-designed GUI Translates to Improved User Confidence. The degree of planning, attention to detail, and quality of the underlying code often are reflected in the GUI. When a developer is truly committed to quality, all aspects of the application reflect this commitment. If the GUI is considered carefully during all phases of planning and development, user interaction flows smoothly and logically. If the visual presentation is consistent and professional, it usually indicates an organized and professional approach throughout. Alternatively, an inconsistent GUI with poor attention to detail may reflect inadequate planning or a lack of commitment to quality—it is unlikely that the user will assume the "invisible" aspects of the application were crafted meticulously if the "visible" aspects exhibit a lack of attention, inconsistencies, or other problems.

Improvements to Visual Presentation and Usability Make Software More Marketable. Answer this: What kind of first impression does your application make? The development cycle for large applications can be long, with a number of twists and turns along the way. It can be difficult to remain subjective throughout this process, so sometimes the best way to gauge the impact of your application is through the eyes of a nonpartial third party.

Enlist the help of a friend or colleague, and while your "appointed critic" is having a look, get a candid answer to this question: Does the visual appearance inspire an immediate sense of quality? Remember, purchasing decisions generally come early on; customers may not have time or opportunity to dig too deeply into the finer details of your offerings, so decisions to purchase your product or service may depend heavily on initial impressions of quality. Because large companies devote extensive resources to the packaging and appearance of products, pleasing visual design is a pervasive expectation for all types of products. If you want your application to present a professional image, the GUI panels will have to look good.

Properly managed, the visual aspects of your GUI can provide a competitive advantage. Unusually distinctive visual presentation enhances product differentiation. If all your competition offers the same, drab, unimaginative interface design, a visually exciting alternative will be a breath of fresh air to potential customers and clients. If being remembered is half the battle in a competitive marketplace, a striking GUI can be your secret weapon.

Extra effort devoted to the "look" of your panels can provide other spin-off benefits you may not have considered. For example, screen captures of well-crafted, clean user-interface panels will be welcome additions to your marketing materials—brochures, catalogs, and of course, the Web site.

Great looking, user-friendly software will inspire positive comments and "word-of-mouth" reviews between friends and colleagues. Word-of-mouth endorsements have been shown to be many times more effective than conventional advertising for generating new customers, so if your application can elicit good reviews, invariably this will have a positive effect on the bottom line.

And while creating an immediate sense of quality is mandatory for the success of commercial applications, its value to in-house projects should never be overlooked either. As pointed out previously, a sense of quality will ensure user acceptance and confidence in the application.

Clear, Attractive GUI Design Can Open the Lines of Communication. Another area where a high-quality GUI can be useful is in the area of internal communications. Many technical people find it challenging to communicate sophisticated or abstract technical concepts to managers with limited technical experience or training; a background in business management or accounting is not the most suitable prerequisite for understanding closed-loop feedback control strategies. A clearly focused GUI can be helpful in these instances. Stepping through your well-designed GUI with nontechnical decision makers will help them to "visually" comprehend the essence of your message. Even without a complete understanding of the technical terminology and underlying scientific principles, this "show and tell" approach can be substantially more effective than verbal communications alone.

1.1.3 Return on Investment

All these benefits do not come without a price. The additional effort devoted to GUI design and planning likely will increase the time it takes you to finish each project —particularly if GUI planning has not been a high priority in the past. Extra development time invariably equates to higher development cost. On the other hand, if long-term costs are factored into the equation, the scales quickly tip in your favor.

While no comprehensive test data have been compiled specifically for LabVIEW development, several UI[JKM2] cost-benefit studies have been completed for conventional software development. For example, in one study focusing on in-house software development and deployment, accrued cost savings were calculated to be more than US $65,000. Other studies on commercial products produced substantially larger savings (Eberts, 19XX[JKM3], pp.12-18). According to sources at IBM, every dollar invested in ease of use returns $10 to $100. IBM goes on to say: "In certain cases, training sessions have been shortened from 1 week to 1 day or 1 hour, saving the companies thousands or even millions of dollars. Significant savings of help-desk calls and service costs are another added bonus when products are made to meet user needs" (IBM Web site, May 2000; URL: **http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/79**).

These claims recognize one of the key realities of running any business: Human costs represent the highest proportion of a company's operating expenses. Assuming that your application will be run several hundred times during its useful life span, saving even a couple of minutes of the operator's time per run will quickly add up to several hours. If the program is intended to run continuously under human operator control, consider the value of even a modest usability improvement of 10 percent. Obviously, cost savings multiply as the number of users increases. If several users will operate the program concurrently or you are developing a networked application that serves data to people throughout a large organization, the aggregate human cost savings over the life span of the application could easily equate to tens or even hundreds of thousands of dollars. And because easy-to-use software is more likely to have a long and useful life, the extra investment in your GUI will save an additional, often overlooked hidden cost-the cost of developing a replacement program.

Directly measurable human cost savings are only part of the whole cost-benefit picture, however. Improvements in usability also can contribute in other indirect but equally dramatic ways to the bottom line. Improved efficiency, faster time to market, enhanced customer satisfaction, improved employee moral-these qualities and others are the universal goals of modern business. With the current trend toward open markets and a global economy, it is not difficult to see why. Quality issues are becoming increasingly important to business survival, and well-designed, efficient, easy-to-use software can play an important role in acquiring and preserving a competitive advantage.

Granted, differences do exist between LabVIEW graphic programming and conventional code-based methods. When reviewing the results of cost-benefit studies compiled in connection with conventional development, these differences should be kept in mind. However, similarities between the underlying user goals-regardless of the development system—indicate that similar long-term benefits and cost savings can be realized in the context of LabVIEW application design. After all, when a frustrated user is staring blankly at the screen, trying to figure out what to do next, it does not make much difference whether the programmer typed the code into a text editor or wired a LabVIEW diagram. The clock is ticking, and from the user's point of view, the details of construction are neither relevant nor helpful. What is important here is how easy it is for the user to find a solution and get back to being productive. The point where the goals of user, employer, and GUI designer intersect is the point where the user is experiencing maximum productivity and efficiency. It is the responsibility of the GUI designer to make this point plainly visible to the user or to provide useful signposts when it is obscured. Challenging? Unquestionably, but the payoff is well worth the investment.

1.1.4 Make Your Own Case

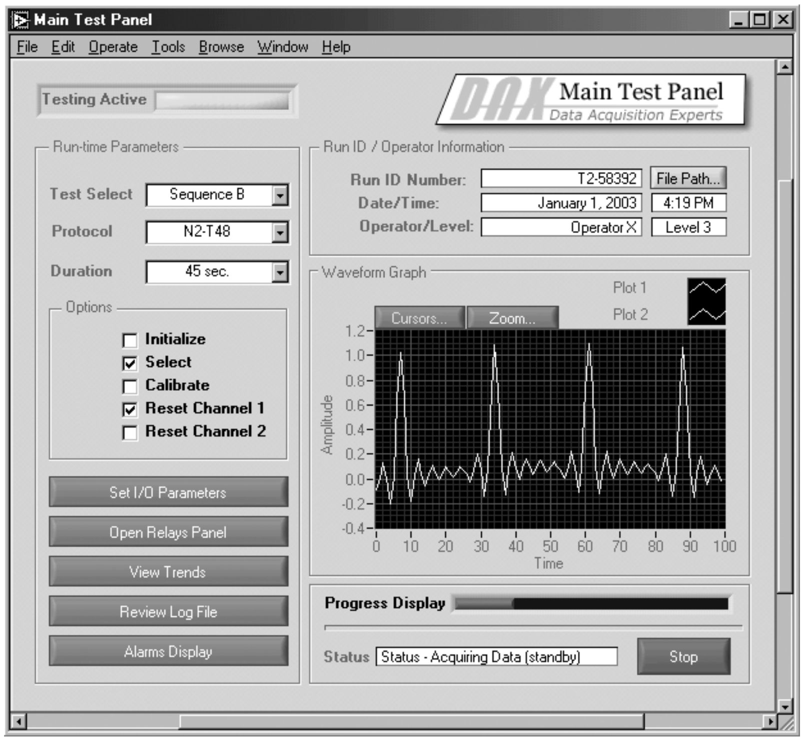
The preceding section summarized a number of key benefits realized through better GUI planning and integration. With further consideration, you may come up with some additional ideas of your own. And while some of these points may resonate more than others—depending on your own experience and biases—it all boils down to a single underlying theme: Quality GUI design is important. Critically important. As you devote more energy to quality GUI design, you will discover rewards on a personal level as well. Watching a new operator, with little or no training, quickly and efficiently navigating your interface inspires a sense of pride and accomplishment. This is, after all, the noble goal of the interface designer: to provide a comfortable, supportive, and efficient virtual environment so that people can get some work done.

1.2 What Defines a Superior GUI?

Understanding the benefits of a high-quality GUI is only one part of the complete picture. While it is great to see what can be expected as the payoff for additional effort, there is a detail that must be addressed: What exactly is a superior GUI? How will you know it when you see it, and—more important—what steps can be taken to ensure that your own GUIs are superior GUIs?

Answers to these questions, as you may have guessed, will take the remaining pages of this book to develop. Unfortunately, there are no shortcuts. As a point of departure, however, consider the following empirical observations:

Certainly, aesthetic considerations must figure into the equation—it is hard to imagine a truly superior GUI that is visually unappealing (see Figure 1-1). However, visual presentation is only one part of the whole story. As we will soon discover, "look and feel" elements become less critical as the user gains more experience with the application. Unquestionably, the effectiveness of any user interface extends far deeper than the "look and feel" of the GUI panel. The true measure of an effective GUI lies in the concepts underlying the methods of interaction—the stuff "below the surface" that determines how quickly and easily a user can realize the intended goals of the application. We will look at this more closely in the next chapter.



Testing Active

Test Select: Sequence B
Protocol: N2-T48
Duration: 45 sec.

Options
- [ ] Initialize
- [✓] Select
- [ ] Calibrate
- [✓] Reset Channel 1
- [ ] Reset Channel 2

Run ID Number: T2-58392
Date/Time: January 1, 2003 4:19 PM
Operator/Level: Operator X Level 3

Progress Display
Status: Status - Acquiring Data (standby)

Figure 1-1 Clean, organized panels inspire confidence and add a professional veneer to your applications.

To achieve a transparent, intuitive interface, energy must be devoted to seeing the problem through the eyes of the intended user. All aspects of the user's problem and how to solve it must be understood: the steps that must be taken, the desired output, what information is needed by the user, how to clearly present information on the screen, the relative frequency and importance of tasks, and so on. Only through a complete understanding of the needs of users can you hope to build a truly effective interface.

So then, our definition of a superior GUI is simply: a GUI that the user finds both visually appealing and functionally capable. Fair enough. But what about the notion that "beauty is in the eye of the beholder"? Surely, evaluating the appearance of a GUI is problematic—after all, what you find "visually appealing" may not fit with the user's definition. Certainly, in matters of personal taste, there is no universal definition.

While it cannot be denied that the aesthetic quality of any GUI is highly subjective, it is important to keep your goal in perspective. Your design does not have to be the GUI equivalent of the *Mona Lisa* to be acceptable to the user—it just has to present a clean, professional image. By applying a few basic principles of graphic design, not only will you avoid the embarrassing design mistakes frequently made by novices, but you also may discover that it is relatively easy to build professional-looking panels. (The issue of graphic design will be covered in Chapter 6, "Graphic Design for Engineers 101—A Crash Course in Layout and Design.")

And what about functionality? Aren't evaluations of the functionality extremely subjective as well?

This question raises an important point about the universality of user interfaces in general. What exactly defines ease of use, and why does one design seem more intuitive to some people than it does to others? The answer to this question will surface in Chapter 5, "From Task Definition to GUI Design," but before we can effectively delve with this thorny issue, the mental process of human-computer interaction must be explored. In the next chapter we will begin our journey by exploring the psychological aspects of human-computer interaction, but first, there are a few personal questions.

1.3 Assessing Your Own GUI Abilities

How convincing are your existing GUIs designs? What are the strengths and weaknesses of your best VI panels? Before diving into specific GUI improvement strategies and techniques, take a moment to evaluate the current state of your GUI design awareness.

Start by selecting a few of the VIs you have developed in the past. Try to choose your best GUI VIs—the ones that represent the current peak of your GUI design abilities. Take a critical look at each of these VIs. Examine them and note their strengths and weaknesses. For example: What do you like about the VIs? What do you dislike and why? Are there any problems? Can you identify the source of problems? Try to be as objective as possible and record any observations that occur to you.

Evaluate your designs with regard to the following qualities:

The "look"

- Aesthetic quality of the organization and layout
- Color choices
- Font choices
- Graphic elements
- Overall visual impression
- Consistency

The “feel”

- GUI object choices—Does behavior match functionality?
- GUI object location, proximity—How do these affect task flow?
- Access to critical and frequently used items
- Navigation and freedom of movement
- Appropriate feedback for each user action
- Interface text: button, menu, and dialog labels and messages

Conceptual elements

- System metaphors
- Ease of use
- The power of each GUI action
- Depth of GUI
- Flexibility and the capacity for growth and change

Take a few moments to jot down your impressions. If there is something about a VI panel that you cannot quite put your finger on—a general perception, either good or bad—make a note of this too. Next, save these notes in a notebook or enter them into a word processor file and save them in a directory with a copy of each of the VIs you have analyzed.

At the end of this book, you will be invited to have another look at these preliminary notes. The understanding acquired through reading the intervening material should provide you with a finely tuned critical eye for spotting flaws and some practical steps for correcting problems and improving the overall design. By comparing these preliminary notes with the notes prepared at the end of this book, you will have a very real measure of the understanding you have gained. So take a moment now to make some notes. Chapter 2 will be waiting when you are finished.