


CHAPTER ONE

INITIAL FORAYS INTO USER INTERFACE DESIGN

IF YOU ASK ten people for their thoughts on user interface design, you will get ten self-proclaimed expert opinions. Designing an interface for a single user grants you the luxury of just asking your customer what they want and doing it, but designing an interface for a large audience forces you to make tough decisions. Here are some stories on the subject of user interface design, starting with probably the most frequently asked question about the Windows 95 user interface.



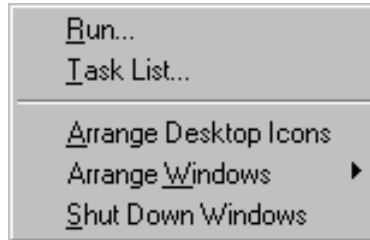
Why do you have to click the Start button to shut down?

BACK IN THE early days of what would eventually be named Windows 95, the taskbar didn't have a Start button. (Later, you'll learn that back in the early days of the project, the taskbar wasn't called the taskbar.)

Instead of the Start button, three buttons were displayed in the lower-left corner: the System button (icon: the Windows flag), the Find button (icon: an



eyeball), and the Help button (icon: a question mark). Find and Help are self-explanatory. The System button gave you this menu:



Over time, the Find and Help buttons eventually joined the System button menu, and the System button menu itself gradually turned into the Windows 95 Start menu. Some menu options such as Arrange Windows (which led to options such as Cascade Windows and Tile Windows Horizontally) moved to other parts of the user interface; others such as Task List vanished completely.

One thing kept showing up during usability tests as a major hurdle: People turned on the computer and just sat there, unsure what to do next.

That's when someone got the idea of labeling the System menu Start. It says, "Psst. Click here." With this simple change, the usability results improved dramatically because, all of a sudden, people knew what to click when they wanted to do something.

So why is Shut down on the Start menu?

When we asked people to shut down their computers, they clicked the Start button. Because, after all, when you want to shut down, you have to start somewhere.

Why doesn't Windows have an "expert mode"?

WE OFTEN GET requests like this:

There should be a slider bar somewhere, say on the Performance tab, that ranges from Novice to Advanced. At the highest level, all the advanced settings are turned on. At the Novice level, all the settings for beginners are turned on. In between, we can gradually enable stuff.

We've been trying to do something like this since even before Windows 95, and it doesn't work.

It doesn't work because those who might be whizzes at Excel will rate themselves as Advanced even though they can't tell a page file from a box of corn flakes. They're not stupid. They really are advanced users. Just not advanced at the skill we're asking them about.

And before you go mocking the beginners: Even so-called advanced users don't know everything. I know a lot about GUI programming, but I only know a little about disk partitioning, and I don't know squat about Active Directory. So am I an expert? When I need to format a hard drive, I don't want to face a dialog box filled with incomprehensible options. I just want to format the hard drive.

In the real world, people who are experts in one area are probably not experts in other areas. It's not something you can capture in a single number.

The default answer to every dialog box is Cancel

THE PROBLEM WITH displaying a dialog box is that people will take every opportunity to ignore it. One system administrator related a story in a *Network World* magazine online contest of a user who ignored a dozen virus security warnings and repeatedly tried to open an infected email attachment, complaining, "I keep trying to open it, but nothing happens." When the administrator asked why the user kept trying to open an attachment from a stranger, the answer was, "It might have been from a friend! They might have made up a new email address and didn't tell me!"¹ This story is a template for how users treat any unexpected dialog: They try to get rid of it.

We see this time and time again. If you are trying to accomplish task A, and in the process of doing it, an unexpected dialog box B appears, you aren't going to stop and read and consider B carefully. You're going to try to find the quickest path to getting rid of dialog B. For most people, this means minimizing it or clicking Cancel or just plain ignoring it.

1. "Why Some People Shouldn't Be Allowed Near Computers," *Network World*, August 23, 2003, <http://napps.networkworld.com/compendium/archive/003362.html>.



This manifests itself in many ways, but the basic idea is, “That dialog box is scary. I’m afraid to answer the question because I might answer it incorrectly and lose all my data. So I’ll try to find a way to get rid of it as quickly as possible.”

Here are some specific examples, taken from conversations I have had with real customers who called the Microsoft customer support line:

- “How do I make this error message go away? It appears every time I start the computer.”

“What does this error message say?”

“It says, ‘Updates are ready to install.’ I’ve just been clicking the X to make it go away, but it’s really annoying.”

- “Every time I start my computer, I get this message that says that updates are ready to install. What does it mean?”

“It means that Microsoft has found a problem that may allow a computer virus to get into your machine, and it’s asking for your permission to fix the problem. You should click on it so the problem can be fixed.”

“Oh, that’s what it is? I thought it was a virus, so I just kept clicking ‘No.’”

- “When I start the computer I get this big dialog that talks about automatic updates. I’ve just been hitting Cancel. How do I make it stop popping up?”

“Did you read what the dialog said?”

“No. I just want it to go away.”

- “Sometimes I get the message saying that my program has crashed and would I like to send an error report to Microsoft. Should I do it?”

“Yes, we study these error reports so we can see how we can fix the problem that caused the crash.”

“Oh, I’ve just been hitting Cancel because that’s what I always do when I see an error message.”

“Did you read the error message?”

“Why should I? It’s just an error message. All it’s going to say is ‘Operation could not be performed because blah blah blah blah blah.’”

When most people buy a car, they don’t expect to have to learn how an engine works and how to change spark plugs. They buy a car so that they can drive it to get from point A to point B. If the car makes a funny noise, they will ignore it as long as possible. Eventually, it may bother them to the point of taking it to a mechanic who will ask incredulously, “How long has it been doing this?” And the answer will be something like, “Oh, about a year.”

The same goes for computers. People don’t want to learn about gigabytes and dual-core processors and security zones. They just want to send email to their friends and surf the Web.

I myself have thrown out a recall notice because I thought it was junk mail. And computers are so filled with pop-up messages that any new pop-up message is treated as just another piece of junk mail to be thrown away.

Those who work at an information desk encounter this constantly. People ignore unexpected information. For example, even when a sign on a door says that “XYZ is closed today,” you can bet that people will walk on in and ask, “Is XYZ open today?”

“No, it’s closed today. Didn’t you see the sign on the door?”

“Hmm, yeah, now that you mention it, there was a sign on the door, but I didn’t read it.”

Automobile manufacturers have learned to consolidate all their error messages into one message called “Check engine.” Most people are conditioned to take the car in to a mechanic when the “Check engine” light goes on, and let the mechanic figure out what is wrong. Is it even possible to have a “Check engine” light for computers? Or would people just ignore that, too? How can a computer even tell whether a particular change in behavior is *normal* or *unintended*?



The best setting is the one you don't even sense, but it's there, and it works the way you expect

ONE SOLUTION THAT many people propose to the issue of “How should something be designed” is “Design it in every imaginable way, then let the end users pick the one they want with an option setting somewhere.” This is a cop-out.


Computers need to be made simpler. This means fewer settings, not more. One way to reduce the number of settings is to make them implicit. You'll see more of this trend as researchers work on ways to make computers simpler, not more complicated.

Your toaster has a slider to set the darkness, which is remembered for your next piece of toast. There is no Settings dialog where you set the default darkness, but which you can override on a slice-by-slice basis.

Yes, this means that if you spent three weeks developing the perfect toaster slider position for Oroweat Honey Wheat Berry, and then you decide for a change of pace to have a slice of rye bread instead, you're going to have to move the slider and lose your old setting. People seem not to be particularly upset by this. The toaster works the way they expect.

Perhaps, you, the power-toaster-user, would want all toasters to let you save up to ten favorite darkness settings. But I suspect most people don't even sense that there are “missing options.” If you started adding options to toasters, people would start wishing for the old days when toasters were simpler and easier to use.

“When I was a kid, you didn't have to log on to your toaster to establish your personal settings.”



In order to demonstrate our superior intellect, we will now ask you a question you cannot answer


DURING THE DEVELOPMENT of Windows 95, a placeholder dialog was added with the title “In order to demonstrate our superior intellect, we will now ask you a question you cannot answer.” The dialog itself asked a technical question that you need a brain the size of a planet to answer. (Okay, your brain didn’t need to be quite that big.)

Of course, there was no intention of shipping Windows 95 with such a dialog. The dialog was there only until other infrastructure became available, permitting the system to answer the question automatically.

But when I saw that dialog, I was enlightened. As programmers, we often find ourselves unsure what to do next, and we say, “Well, to play it safe, I’ll just ask users what they want to do. I’m sure they’ll make the right decision.”

Except that they don’t. As we saw earlier, the default answer to every dialog box is Cancel. If you ask the user a technical question, odds are that they’re just going to stare at it blankly for a while, then try to cancel out of it. The lesson they’ve learned is this: Computers are hard to use.

So don’t ask questions the user can’t answer. It doesn’t get you anywhere, and it just frustrates the user.



Why doesn’t Setup ask you if you want to keep newer versions of operating system files?

WINDOWS 95 SETUP would notice that a file it was installing was older than the file already on the machine and would ask you whether you wanted to keep the existing (newer) file or overwrite it with the older version.

Asking the user this question at all turned out to have been a bad idea. It's one of those dialogs that asks users a question they have no idea how to answer.

Suppose you're installing Windows 95 and you get the file version conflict dialog box. "The file Windows is attempting to install is older than the one already on the system. Do you want to keep the newer file?" What do you do?

Well, if you're like most people, you say, "Um, I guess I'll keep the newer one," so you click Yes.

And then a few seconds later, you get the same prompt for some other file. And you click Yes again.

And then a few seconds later, you get the same prompt for yet another file. Now you're getting nervous. Why is the system asking you all these questions? Is it second-guessing your previous answers? Often when this happens, it's because you're doing something bad and the computer is giving you one more chance to change your mind before something horrible happens. Like in the movies when you have to type Yes five times before you can launch the nuclear weapons.

Maybe this is one of those times.

Now you start clicking No. Besides, it's always safer to say "No," isn't it?

After a few more dialogs (clicking No this time), Setup finally completes. The system reboots, and ... it blue-screens.

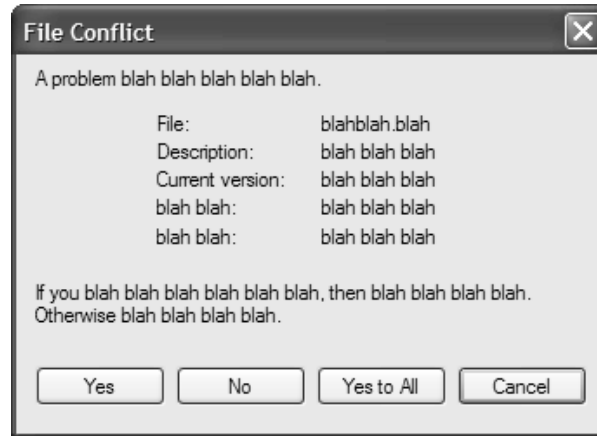
Why?

Because those five files were part of a matched set of files that together form your video driver. By saying "Yes" to some of them and "No" to others, you ended up with a mishmash of files that don't work together.

We learned our lesson. Setup doesn't ask this question any more. It always overwrites the files with the ones that come with the operating system. Sure, you may lose functionality, but at least you will be able to boot. Afterward, you can go to Windows Update and update that driver to the latest version.

Some have suggested that expanding the dialog with more explanatory text would solve the problem, but this misses the fact that people don't want to be bothered with these dialogs to begin with, as well as the fact that more information doesn't help anyway because the user doesn't have the background knowledge necessary to make an informed decision in the first place.

To a user, the dialog looks like this:



Making the dialog longer just increases the number of blahs. It's like trying to communicate with someone who doesn't speak your language by repeating yourself louder and more slowly. Users just want to surf the Web and send email to their grandchildren. Whatever you put in the dialog, they simply won't read it. Giving the dialog more buttons merely increases the paralysis factor.

Do you know the name of your printer driver? Or whether you should keep version 4.12.5.101 or downgrade it to 4.12.4.8? I sure don't.

Thinking through a feature

EVERYONE HAS A suggestion for a taskbar grouping feature. It's just a little bit of code; why not just do it?

Writing the code is the easy part.

Designing a feature is hard.

You have several audiences to consider. It's not just about the alpha geeks; you have to worry about the grandmothers, the office workers, the IT departments. They all have different needs. Sometimes a feature that pleases one group offends another.

So let's look at some of the issues surrounding the proposed feature of allowing users to selectively ungroup items in the taskbar.

One issue with selective grouping is deciding the scope of the feature. Suppose the user ungroups Internet Explorer, then closes all the Internet Explorer windows, and then opens two new Internet Explorer windows: Do the new ones group?

If so, you now have an invisible setting. How do you configure grouping for programs that aren't running? (How do you configure something that you can't see?)

Suppose you've figured that out. That's fine for the alpha geeks, but what about Grandma?

"The Internet is all disorganized."

"What do you mean?"

"My Internet windows are all disorganized."

"Can you explain a little more?"

"My taskbar used to be nice and organized, but now the Internet parts are disorganized and spread out all over the place. It used to be nice and neat. I don't know how it happened. I hate the Internet. It's always messing up my computer."

What is the user interface for selective ungrouping? Anything that is on a context menu will be executed accidentally by tens of thousands of people due to mouse twitching. Putting the regroup onto the context menu isn't necessarily good enough because those people don't even realize it was a context menu that did it. It was just a mouse twitch.

Mouse twitches cause all sorts of problems. Some people accidentally dock their taskbar vertically; others accidentally resize their taskbar to half the size of the screen. Do not underestimate the havoc that can be caused by mouse twitching.

Soon people will want to do arbitrary grouping. "I want to group this command prompt, that Notepad window, and this Calc window together."

What about selective ungrouping? "I have this group of ten windows, but I want to ungroup just two of them, leaving the other eight grouped together."

When you have selective/arbitrary grouping, how do you handle new windows? What group do they go into?

Remember: If you decide, “No, that’s too much,” thousands of people will be cursing you for not doing enough. Where do you draw the line? And also remember that each feature you add will cost you another feature somewhere else. Manpower isn’t free.

But wait, the job has just begun. Next, you get to sit down and do the usability testing.


Soon you’ll discover that everything you assumed to be true is completely wrong, and you have to go back to the drawing board. Eventually, you might conclude that you overdesigned the feature and you should go back to the simple on/off switch.

Wait, you’re still not done. Now you have to bounce this feature off corporate IT managers. They will probably tear it to shreds, too. In particular, they’re going to demand things such as remote administration and the capability to force the setting on or off across their entire company from a central location. (And woe unto you if you chose something more complicated than an on/off switch: Now you have to be able to deploy that complex setting across tens of thousands of computers, some of which may be connected to the corporate network via slow modems.)

Those are just some of the issues involved in designing a feature. Sometimes I think it’s a miracle that features happen at all!

(Disclaimer: I’m not saying this is how the grouping feature actually came to be. I just used it as an illustration.)

Curiously, when I bring up this issue, the reaction of most people is not to consider the issue of trade-offs in feature design but rather to chip in with their vision of how the taskbar should work. “All I want is for the taskbar to do X. That other feature Y is useless.” The value of X and Y changes from person to person; these people end up unwittingly proving my point rather than refuting it.



When do you disable an option, and when do you remove it?

WHEN YOU'RE DISPLAYING a menu item or a dialog option, and the option is not available, you can either disable it or you can remove it. What is the rule for deciding which one to do?

Experiments have shown that if something is shown but disabled, users expect that they will be able to get it enabled if they tinker around enough.

Therefore, leave a menu item shown but disabled if there is something the user can do to cause the operation to become available. For example, in a media playback program, the option to stop playback is disabled if the media file is not playing. When it starts playing, however, the option becomes available again.

On the other hand, if the option is not available for a reason the user has no control over, remove it. Otherwise the user will go nuts looking for the magic way to enable it. For example, if a printer is not capable of printing color, don't show any of the color management options, because there's nothing the user can do with your program to make that printer a color printer.

By analogy, consider a text adventure game. The player tries something clever, such as "Take the torch from the wall," and the computer replies, "You can't do that, yet." This is the adventure game equivalent to graying out a menu item. The user is now going to go nuts trying to figure out what's happening: "Hmm, maybe I need a chair, or the torch is too hot, or I'm carrying too much stuff, or I have to find another character and ask him to do it for me."

If it turns out that the torch is simply not removable, what you've done is send the user down fruitless paths to accomplish something that simply can't be done. For an adventure game, this frustration is part of the fun. But for a computer program, frustration is not something people tend to enjoy.

Note that this isn't a hard-and-fast rule; it's just a guideline. Other considerations might override this principle. For example, you may believe that a consistent menu structure is more desirable because it is less confusing. (A media playback program, for example, might decide to leave the video-related options visible but grayed when playing a music file.)

When do you put ... after a button or menu?

SAVE AS... APPEARS on some menus. You'll also find plenty of Customize... buttons. What is the rule for dots?

Many people believe that the rule for dots is this: "If it's going to display a dialog, you need dots." This is a misapprehension.

The rules are spelled out in the Windows User Interface Design Specifications and Guidelines (what a mouthful) in the section titled "Ellipses."

You should read the guidelines for the full story, but here's the short version: Use an ellipsis if the command requires additional information before it can be performed. Sometimes the dialog box is the command itself, such as About or Properties. Even though they display a dialog, the dialog *is the result*, as opposed to commands such as Print, where the dialog is *collecting additional information prior to the result*.

User interface design for vending machines

HOW HARD CAN it be to design the user interface of a vending machine? You accept money, you have some buttons, users push the buttons, and they get their product and their change.

At least in the United States, many vending machines arrange their product in rows and columns. To select a product, you press the letter of the row and the number of the column. Could it be any simpler?

It turns out that subtleties lurk even in something this simple.

If the vending machine contains ten items per row, and you number them 1 through 10, a person who wants to buy product C10 has to push the buttons C and 10. But in our modern keyboard-based world, there is no 10 key. Instead, people press 1 followed by 0.

What happens if you type $C + 1 + 0$? After you type the 1, product C1 drops. Then the user realizes that there is no 0 key. And he bought the wrong product.

This is not a purely theoretical problem. I have seen this happen myself. How would you fix this?

One solution is simply not to put so many items on a single row, considering that people have difficulty making decisions if given too many options. On the other hand, the vendor might not like that design; their goal might be to maximize the number of products.

Another solution is to change the labels so that the number of button presses needed always matches the number of characters in the label. In other words, no buttons with two characters on them (for example, a 10 button).

You could switch the rows and columns so that the products are labeled 1A through 1J across the top row and 9A through 9J across the bottom. This assumes you don't have more than nine rows, however. Some vending machines have many more selections on display, resulting in a very large number of rows.

If you have exactly ten items per row, you can call the tenth column 0. Notice, however that you also should remove rows I and O to avoid possible confusion with 1 and 0.

Some vending machines use numeric codes for all items rather than a letter and a digit. For example, if the cookies are product number 23, you punch $2 + 3$. If you want the chewing gum (product code 71), you punch $7 + 1$. What are some problems with having your products numbered from 1 to 99?

Here are a few problems. You may have come up with others:

- Products with codes 11, 22, 33, and so on may be selected accidentally. A faulty momentary switch might cause a single key-press to register as two, or a user may press the button twice by mistake or frustration.
- Product codes less than ten are ambiguous. Is a 3 a request for product number 3, or is the user just being slow at entering 32? Solving this by adding a leading zero will not work because people are in the habit of ignoring leading zeros.

- Product codes should not coincide with product prices. If there is a bag of cookies that costs 75 cents, users are likely to press 75 when they want the cookies, even though the product code for the cookies is 23.

User interface design for interior door locks

HOW HARD CAN it be to design the user interface of an interior door lock?

Locking or unlocking the door from the inside is typically done with a latch that you turn. Often, the latch handle is in the shape of a bar that turns.

Now, there are two possible ways you can set up your lock. One is that a horizontal bar represents the locked position, and a vertical bar represents the unlocked position. The other is to have a horizontal bar represent the unlocked position and a vertical bar represent the locked position.

For some reason, it seems that most lock designers went for the latter interpretation. A horizontal bar means unlocked.

This is wrong.

Think about what the bar represents. When the deadbolt is locked, a horizontal bar extends from the door into the door jamb. Clearly, the horizontal bar position should reflect the horizontal position of the deadbolt. It also resonates with the old-fashioned way of locking a door by placing a wooden or metal bar horizontally across the face. (Does no one say “bar the door” any more?)

Car doors even followed this convention, back when car door locks were little knobs that popped up and down. The up position represented the removal of the imaginary deadbolt from the door/jamb interface. Pushing the button down was conceptually the same as sliding the deadbolt into the locked position.

But now, many car door locks don't use knobs. Instead, they use rocker switches. (Forward means lock. Or is it backward? What is the intuition there?) The visual indicator of the door lock is a red dot. But what does it mean? Red clearly means *danger*, so is it more dangerous to have a locked door or an unlocked door? I can never remember; I always have to tug on the door handle.

(Horizontally mounted power window switches have the same problem. Does pushing the switch forward raise the window or lower it?)



The evolution of mascara in Windows UI

THE LOOK OF the Windows user interface has gone through fashion cycles.

In the beginning, there was Windows 1.0, which looked very flat because screen resolutions were rather low in those days, and color depth was practically nonexistent. If you had 16 colors, you were doing pretty well. You couldn't afford to spend very many pixels on fluff such as borders, and shadows were out of the question because of lack of color depth.

The *flat look* continued in Windows 2.0, but Windows 3.0 added a hint of 3D, with a touch of beveling in push buttons.

Other people decided that the 3D look was the hot new thing, and libraries sprang up to add 3D shadow and outlining effects to nearly everything. The library CTL3D.DLL started out as just an Excel thing, but it grew in popularity until it became the standard way to make your dialog boxes *even more 3D*.

Come Windows 95, and even more of the system had a 3D look. For example, beveling appeared along the inside edge of the panes in the Explorer window. Furthermore, 3D-ness was turned on by default for all programs that marked themselves as designed for Windows 95. For programs that wanted to run on older versions of Windows as well, a new dialog style DS_3DLOOK was added, so that they could indicate that they wanted 3D-ization if available.

And if the 3D provided by Windows 95 by default wasn't enough, you could use CTL3D32.DLL to make your controls *even more 3D than ever before*. By this point, things started getting really ugly. Buttons on dialog boxes had so many heavy black outlines that it started to look like a really bad mascara job.

Fortunately, like many fashions that get out of hand, people realized that too much 3D is not a good thing. User interfaces got flatter. Instead of using 3D effects and bold outlines to separate items, subtler cues were used. Divider lines became more subdued and sometimes disappeared entirely.

Microsoft Office and Microsoft Money were two programs that embraced the *less-is-more* approach. The beveling is gone, and there are no 3D effects. Buttons are flat and unobtrusive. The task pane separates itself from the content pane by a simple gray line and a change in background shade. Even the toolbar has gone flat. Office 2000 also went largely flat, although some simple 3D effects linger (in the grooves and in the scrollbars, for example).

Windows XP jumped on the *flat-is-good* bandwagon and even got rid of the separator line between the tasks pane and the contents pane. The division is merely implied by the change in color. “Separation through juxtaposition” has become the new mantra.

Office XP and Outlook 2003 continue the trend and flatten nearly everything aside from the scrollbar elements. Blocks of color are used to separate elements onscreen, sometimes with the help of simple outlines.

So now the pendulum of fashion has swung away from 3D back toward flatness. Who knows how long this school of visual expression will hold the upper hand? Will 3D return with a vengeance when people tire of the starkness of the flat look?



