

# It's the User Interface, Stupid

by Philip Chu

## Table of contents

1 Publication Information.....	2
2 You Have to Care.....	2
3 Be Consistent.....	3
4 Make It Clear.....	4
5 Make It Easy.....	5
6 But Not Too Easy.....	6
7 The Customer Is Always Wrong.....	7
8 The Computer Is Always Right.....	7
9 It's Not Just a Pretty Face.....	8
10 It's Not Just the Application.....	8
11 It's Not Just For Computers.....	9

## 1. Publication Information

Copyright ©2004-2008 by Philip Chu All rights reserved.



"It's a Unix system! I know this!" - Lex, recognizing a 3D desktop interface, in *Jurassic Park*

## 2. You Have to Care

Anyone can create a decent user interface. Designing reasonable user interfaces is not rocket science. It's not even computer science. I know there are plenty of researchers dedicated to the study of effective user interfaces and their design, but bad user interfaces occur not due to lack of training in cognitive science - bad user interfaces happen because people don't try.

- If you pay me a visit at my condo complex, it might take you a while to find my name on the phone number list pasted by the entry intercom. It's not listed alphabetically by name - instead, it's sorted by the three-digit phone code. So if you knew my phone number

already, you could look it up easily! One might suspect that the staff at our condo management company was just sticking to an unfriendly but logical system, but there are a few names, including mine, tacked on at the end of the list in seemingly random order. Those names happened to be the ones the management staff forgot to include the first time around.

Now, it's not like the names are engraved in stone. The page was obviously printed with a word processor. It's not that hard to move the text cursor when you're typing, and it's not hard (I hope) to alphabetize, especially when the list includes less than twenty numbers. This is just another case of someone not caring enough to do a good job or think about how to do a good job.

Designing a good user interface isn't always easy, but anyone can identify a bad one. We deal with interfaces all the time, from web sites to ATM machines to microwave ovens, and we have an idea what interfaces we like and which ones we don't. Even those of a contrary nature who delight in presenting puzzles, or those of an apathetic nature who just don't care, still have strong opinions on the interfaces they have to deal with.

That said, not everyone is suited to design an interface for everyone else. For example, programmers who like command-line interfaces and don't see the point of windowed desktops are probably not going to make very good GUI's. But it's a bad rap to attribute bad user interfaces as the work engineers who design products only engineers can use. Forget that left-brain/right-brain folderol. Code API's, whether internal or external, are a form of user interface - good ones are a delight to use and bad ones are painful to use, causing errors, delay and aggravation. (If you don't like Microsoft GUI's, try looking at their API's) And a "creative" graphic designer might design something that would look great in a Powerpoint presentation but completely mystify users.

The best interfaces appear in labors of love that the creators want to take home with them. The worst products are those implemented with just enough effort to satisfy checklists from the marketing department.

### **3. Be Consistent**

The most important rule in user interface design is consistency. Just as drivers rely on consistent conventions with the shape and content of road signs, the user can't navigate the application effectively if it involves constant trial and error.

- I see this problem in various Windows applications (and Windows itself), when many, but not all, operations on an item are available from a right-click menu, and some are only available from one of the menubar menus. This results in time wasted when the desired function is not immediately found, and an obviously less trustworthy user

interface model.

- A common inconsistency in computer and video games is the lack of consistency between options in the front end (at the beginning of the game) and those provided by the in-game menu. For example, if audio settings are available in-game, why should you have to exit to the front end menu to change video?

It's better for the interface to be awkward in a consistent manner than to be inconsistently intuitive in some areas and different in others.

- The Windows recycle bin is a great safety feature (although not an original idea, of course). To really delete a file you have to delete and then really delete by emptying the trash bin. On the other hand, as I discovered to my chagrin on one project, the same delete from a network-shared disk is only a one step process.

Microsoft is an easy target, but Apple isn't guiltless, either.

- I was one of the original adoring Mac fans, but the insistence (was it Steve Jobs?) on a one-button mouse was misguided. Since the resulting hack - double-clicking - was copied blindly by Microsoft along with everything else, I have spent hours explaining with torturous logic to my condo neighbors (they figured out I was a computer guy) why they're supposed to double-click on desktop icons but single-click buttons and web links, and forget control-click, shift-click, option-click or (Mac) command-click. And watch out for those laptop touchpads that turn lingering thumbs into double-clicks or mouse drags! Even the MacOSX desktop can't make up it's mind - double-click opens files in the Finder, on the Desktop and in chooser dialogs, but the Dock and the System Preferences respond to single-clicks. More than once I've accidentally changed my screen resolution by double-clicking the Display applet (the first click opens it, the second chooses an annoyingly low resolution setting)

Inconsistencies in API design slow down programmers and are sometimes downright dangerous.

- I recently helped a client resuscitate a legacy Windows 98 product and get it running on Windows XP. Tracking down some odd artifacts in the GUI, I was horrified to discover that all of the MFC classes took an attribute mask in the same parameter position except one, and the previous engineers on the product hadn't noticed the exceptional case because the bits they passed as the wrong argument just miraculously happened to work - seven years ago.

## **4. Make It Clear**

Clarity seems like an obvious goal in user interface design, yet how many programs allow

you to sit down and just get started?

- I thought after the whole hanging chad business that the new electronic voting systems would at least have simple user interfaces. And the ones I've encountered at my local polling location do have a very simple controller - a clickable track dial for moving the cursor around the screen and selecting answers. However, the system allows you to position the cursor anywhere on the screen, including the question fields, so it's not clear what you're supposed to select, and if you do select a field that's not selectable, well, nothing happens.

API's also can suffer from tack-on-itis.

- If you program with the Microsoft Win32 API, you have to remember to use the newer versions of some functions that are simply extended with an "Ex" suffix and additional arguments, e.g. CreateWindowEx instead of CreateWindow. Try using Win32 without constantly referring to the documentation. Just try it.

## **5. Make It Easy**

The one and only time I tried to order from a certain MacDonald's near Johns Hopkins University in Baltimore, the cashier responded "We don't have any right now." After trying a couple of other items, she finally informed me that they weren't serving lunch, yet. But as I ordered a McMuffin the minute hand advanced from 10:59 to 11 AM and the cashier duly informed me that breakfast was no longer being served.

User interfaces can be bureaucratic, too. Some force you to jump through a specific sequence of hoops, and sometimes you have to start all over if you get one thing wrong. Voice mail is pretty much designed to do this. But often this results from implementors taking the least-effort approach and just tacking on functionality.

- High-end 3D content creation systems tend to have feature-rich interfaces - I worked on one that just chained together sequences of generic popups because that was the easiest way to program it. To use a texture, bring up a file chooser to select the image file, a color chooser for each of several shading options, and popup numeric entry dialog for each numeric option. Contrast this with the Macintosh control panel (old or new) - everything in one dialog.

Put the related options together on one dialog, and don't make the user leave until done. And think about whether you even really need a dialog.

- In Windows Explorer, you have to right click on a drive icon and select the Properties dialog to see how much space is available. In the Macintosh Finder, it's always displayed at the bottom of the window.

Jumping through hoops on a desktop application is annoying enough, but on web sites it's excruciating.

- I've found travel planning on the American Airlines reservation web site terribly annoying if I'm trying browse. Type in your travel start and return dates, hit Submit, and if you don't find something you like, you have to start all over. If you don't "Search by Fare", the resulting list of flights doesn't show the respective fare, and you have to choose a flight and proceed to the reservation to see how much it'll cost.

The sites for other airlines (Jet Blue and Southwest, for example) allow you to conveniently browse forward and backward a day at a time to compare flights. Moreover, they immediately show results for surrounding days so there's a good chance you'll see all the options you want.

Obviously, it should be obvious what to do. If you have to explain it, there's room for improvement.

## **6. But Not Too Easy**

On the other hand, there are some functions that you don't want to make too accessible.

- I worked on one rather large and flakey application that had a "Reset" button prominently displayed on the user interface. Users would click on that button as if it was an accelerator pedal anytime the application seemed unresponsive, i.e. anytime the user became impatient. The reset function was useful, but should have had either a confirmation step or been an option within a menu. As it was, I suspect the reset probably interrupted more operations than it recovered, and it surely made debugging more difficult.

Emergency and safety features, like fire extinguishers or the emergency break in your car, should be obvious in function but inconvenient enough to prevent accidental use. But in keeping users out of trouble, don't be patronizing.

- I worked on a chart display that had all kinds of unnecessary "features" that tried to avoid displaying anything that was not deemed presentable. For example, if the user zoomed out enough (using a scale slider) so that smaller items on the chart did not fully enclose their text labels, then those items simply disappeared from the chart. This was a bad solution to a non-existent problem - it's easy enough to zoom back in if the display is offending the user's esthetic sensibilities, and magically omitting data risks confusing or, worse, misleading the user.

Don't presume to tell users what they want to do or see. Just give them the tools to do it.

## **7. The Customer Is Always Wrong**

Well, not really. But if customers really knew what they wanted, they could start their own company.

- I worked on a computer graphics tool that featured a somewhat atypical menu system - rather than go to a menu bar or an ever-present vertical list of all the menu options on the side of the screen (common among our high-end competitors), our users would shift-click inside our 3D modelling window to get a popup with cascading menus. When we took a survey among potential customers, they said they wanted a vertical side menu.

Of course we implemented the side menu, but that was a lot slower for professional users who don't want to drag the mouse across the screen every time they need select a new operation or mode, so we had another mouse click warp the mouse the side menu - the type of design decision that only makes sense in the context of a previous bad decision. And then of course the heavily populated side menu didn't fit on the screen when we ported the product from the high-end Silicon Graphics workstations to Windows NT machines. And guess what - anyone who actually bought and used the product for serious work used the original menus.

If it's as easy as just asking customers or your salespeople, then hire them to do your job. Relying on user surveys, focus groups, marketing checklists and the like is a cop out. There's no substitute for observation, analysis and innovation. And you don't need huge resources to get it done.

- I spent one project listening to a user interface programmer complain that we needed user testing. Meanwhile, he was oblivious or dismissive of every stumble, inconvenience and point of confusion encountered by the testers and project manager when trying out the product.

You don't need to make user testing a big formal deal. Just do it. Watch, listen, learn.

## **8. The Computer Is Always Right**

User interface specialists often ridicule programmers for presenting a user-interface that models what the program is really doing instead of a simpler user model. But they are at best half-right. Hiding complexity is usually a good thing, but presenting a user model that does not match the the application's internal model is a recipe for trouble.

- When I was a user-interface programmer for a large video game project, the game designers came to me with all kinds of game features that they just assumed involved cosmetic tweaks to the HUD, since they just considered how those features would look.

More often than not, they involved fundamental changes to the game engine.

If the application model is too difficult to use, or just plain wrong, then that's what needs to be fixed.

## **9. It's Not Just a Pretty Face**

User interfaces are getting better looking all the time, but looks aren't everything.

- I once worked for a neat freak who went around the office at night arbitrarily tossing anything she found on desktops into drawers. One book went missing for months. Worse, she instructed her lackeys to wire up the computers so it was nice and tidy, and utterly inconvenient. The test staff had to move desks in order to switch test equipment, and I spent an afternoon testing video game memory functions by repeatedly crouching and blindly reaching behind a box under the desk, because gosh, it would have looked too ugly to have those slots facing forward!

Good looks make a good first impression, but that's all you're getting if the interface is painful to use and everything is hard to find.

- I worked for one manager who had a minor in visual arts and insisted on designing the user interface himself, mainly by drawing pictures. That's not bad in itself, and the pictures looked nice, but he seemed to expect that to be the whole process. There was no description of the user flow, not much thought to consistency, user model, or grouping controls by function rather than appearance. And despite this being a Mac app, no reference (or reading) or the Apple Human Interface Guidelines, which is the industry gold standard.

Artistic skill is useful (I'm afraid I can't design an icon to save my life), but user interface design is about analysis, empathy, and diligent research. It may not be rocket science, but it is engineering.

## **10. It's Not Just the Application**

User interface design is about user experience, and the application interface is just one part of that.

- Apple gets it right. Buying my Macbook Pro at the Apple Store was like buying a luxury car. I spent some time gazing at it in the showroom (the Apple Store), purchased it in a few minutes from an Apple Store rep who immediately retrieved it from the back and handed it to me, carried it out in a nifty box with handle, carefully unfolded the quality cloth that wrapped the laptop, and zipped through the setup process. And now, passersby

at Starbucks often comment how cool my laptop looks.

Contrast this with my previous computer purchases. Every time I bought a computer from Fry's, it turned out they gave me a previously-returned box (in the case of a Powerbook, the registration had been partially filled out and the MacOS install was only halfway complete - as it consisted of a dozen floppy disks, I guess the previous purchaser got tired of swapping floppies). Best Buy aggressively pushed an extended care warranty on me, left me in line for fifteen minutes to complete the purchase, and somehow I got billed by Microsoft for their Internet service (I wasn't the only one - I discovered years later that I missed out on a class action lawsuit). At CompUSA I watched a sales associate hunt-and-peck an old terminal for fifteen minutes so he could print out the paperwork on an equally old dot-matrix printer. Not quite as bad as the buying experiences for both my Toyotas (nice cars, slimy salesmen), but I don't have to upgrade my car every couple of years.

It doesn't matter how well you design your application if the user is turned off before even using the product. Whenever possible, I start all of my projects in designing the installer. And don't forget support - all of that goodwill engendered by smooth installation and a nice interface can be cancelled out by one lousy call center.

## **11. It's Not Just For Computers**

While it's easy to complain about the state of computer user interfaces, and it's all warranted, at least there are plenty of people complaining about it. If only that much attention was paid to everyday things.

- Take my bills (please). Like GUI's, they keep getting glitzier, as my phone and utility providers proudly announce their "new look". But I still have bills that are folded and creased just far enough that when I detach the tear-out stub to include with the check, they don't fit in their envelopes without folding them down a quarter inch. My utility bill doesn't list the account number on the stub, so I'll detach it, fill in the balance paid, start writing the check, and realize I have to go back to the other half to find the account number. My doctor's bill is worse - it doesn't even show the balance due on the stub. And while the main part of the bill shows the balance (of course), it only lists account changes in the month the bill was issued - any other charges made since my last visit to the doctor are included in the balance but will never, ever be listed anywhere! My question, why bother listing any charges at all?