JOEL ON SOFTWARE

I'm Joel Spolsky, a software developer in New York City. More about me.

APRIL 10, 2000 *by* JOEL SPOLSKY

Controlling Your Environment Makes You Happy

SOFTWARE DESIGNER, UIBOOK

Most of the hard core C++ programmers I know *hate* user interface programming. This surprises me, because I find UI programming to be quintessentially easy, straightforward, and fun.

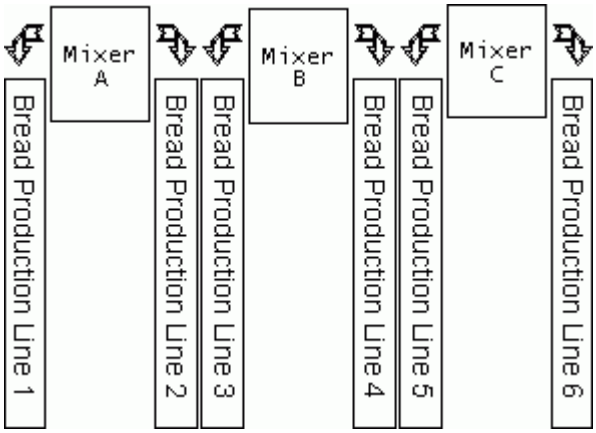
It's *easy* because you usually don't need algorithms more sophisticated than how to center one rectangle in another. It's *straightforward* because when you make a mistake, you immediately see it and can correct it. It's *fun*, because the results of your work are immediately visible. You feel like you are sculpting the program directly.

I think most programmers' fear of UI programming comes from their fear of doing UI *design*. They think that UI design is like graphics design: the mysterious process by which creative, latte-drinking, all-dressed-in-black people with interesting piercings produce cool looking artistic stuff. Programmers see themselves as analytic, logical thinkers: strong at reasoning, weak on artistic judgment. So they think they can't do UI design.

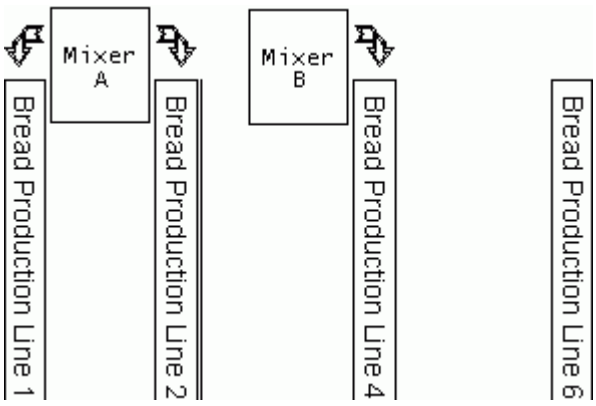
Actually, I've found UI design to be quite easy and quite rational. It's not a mysterious matter that requires a degree from an art school and a penchant for neon-purple hair. There is a rational way to think about user interfaces with some simple, logical rules that you can apply anywhere to improve the interfaces of the programs you work on.

I'm not going to give you "Zen and the Art of UI Design". It's not art, it's not Buddhism, it's just a set of rules. A way of thinking rationally and methodically. This book is designed for programmers. I assume you don't need instructions for *how* to make a menu bar; rather, you need to think about what to put in your menu bar (or whether to have one at all). There is one primary axiom I'll teach you which guides all good UI design, and it's not hard to understand at all.

My first real job was in a big, industrial bakery. The bakery was designed to have six bread production lines. For every two production lines, there was a dough mixer, which produced 180 kg lumps of dough that could be dumped to the left or the right:



Well, this was the design. In reality, Mixer C hadn't been built yet, nor had lines 3 or 5. So the arrangement was:



Alert readers will be wondering, “how did the dough get from Mixer B to production line 6?” Well, that’s where Wee Joel came in. My job, if you can believe this, was to stand on the left of Mixer B, then *catch* the giant 180 kg lumps of dough as they flew out of the mixer in a big bathtub-with-wheels, then roll the bathtub over to production line 6, and, using a winch-like device, heave the dough onto line 6. I had to do this once every ten minutes, from about 10 PM until 4 AM.

There were other complications. Line 6 couldn’t really handle 180 kg of dough all at once, so I had to slice it with a giant knife into about 10 pieces. I don’t even want to go into how absurdly difficult that was.

The first few days, of course, I was terrible at this job. It seemed nearly impossible. Every bone in my body ached. My blisters had blisters. I had aches in places where I didn’t know I had places.

At first I just couldn’t keep line 6 supplied with dough. Every time they had an interruption in the dough, this caused a big gap on the assembly line. When the gap rolled into the oven, the oven (expending a constant amount of energy over a reduced amount of dough) started to heat up more, which burnt the bread. Sometimes, line 6 would get gummed up and stop production, but the mixer went right on ahead producing dough for me, and I would run the risk of running out of bathtubs-with-wheels to store the dough in. When this happened, I actually had to clean and oil the floor and dump the dough on the floor to be scraped up later. Not that this would work very well, because if the dough got older than about 30 minutes it would ferment and wouldn’t make good bread. If this happened, you had to chop it up into 5 kg pieces and put one piece into the mixture for each future batch.

After a week or so, I got good enough at the routine that I actually had, if I remember correctly, 2 minutes free for every 10 minute dough-cycle to rest. I figured out a precise schedule and learned how to tell the mixer to skip a batch when the production line stopped.

And I started to think about why, as the beer commercial asks, *some days are better than others*.

One day, thinking about this problem, I noticed that one of the bathtubs-with-wheels had pretty lousy wheels that wouldn't turn well. Sometimes this bathtub did not go where I pushed it, and bumped into things. This was a small frustration. Sometimes, as I was pulling the chain to winch up the bathtub, I scraped myself — just a little bit — on a splinter of metal on the chain. Another small frustration. Sometimes, as I ran with an empty bathtub to catch a dough emission about to fly out of the mixer, I slipped on a little bit of oil on the floor. Not enough to fall, mind you, just a tiny, small frustration.

Other times, I would have tiny victories. I learned to time the dough production perfectly so that fresh dough would arrive just seconds before the previous batch ran out. This guaranteed the freshest dough and made the best bread. Some of the victories were even tinier: I would spot a tiny blob of dough that had flung off of the mixer and attached itself to the wall, and I would scrape it off with a paint scraper I carried in my back pocket and throw it in the trash. YES! When slicing the dough into pieces, sometimes it just sliced really *nicely* and *easily*. Tiny moments of satisfaction, when I managed to control the world around me, even in the smallest way.

So that's what days were like. A bunch of tiny frustrations, and a bunch of tiny successes. But they *added up*. Even something which seems like a tiny, inconsequential frustration affects your mood. Your emotions don't seem to care about the magnitude of the event, only the quality.

And I started to learn that the days when I was happiest were the days with lots of small successes and few small frustrations.

Years later, when I got to college, I learned about an important theory of psychology called Learned Helplessness, developed by Dr. Martin E. P. Seligman. This theory, backed up by years of research, is that a great deal of depression grows out of a feeling of *helplessness*: the feeling that you cannot control your environment.

The more you feel that you can control your environment, and that the things you do are actually working, the happier you are. When you find yourself frustrated, angry, and upset, it's probably because of something that happened that you could not control: even something small. The space bar on your keyboard is not working well. When you type, some of the words are stuck

together. This gets frustrating, because you are pressing the space bar and *nothing is happening.* The key to your front door doesn't work very well. When you try to turn it, it sticks. Another tiny frustration. These things add up; these are the things that make us unhappy on a day-to-day basis. Even though they seem too petty to dwell on (I mean, there are people *starving* in Africa, for heaven's sake, I can't get upset about *space bars*), nonetheless they change our moods.

Let's pause for a minute and go back to computers.

We're going to invent a typical Windows power user named Pete. When you're thinking about user interfaces, it helps to keep imaginary users in mind. The more realistic the imaginary user is, the better you'll do thinking about how they use your product. Pete is an accountant for a technical publisher who has used Windows for six years at the office and a bit at home. He is fairly competent and technical. He installs his own software; he reads PC Magazine, and he has even programmed some simple Word macros to help the secretaries in his office send invoices. He's getting a cable modem at home. Pete has never used a Macintosh. "They're too expensive," he'll tell you. "You can get a 700 Mhz PC with 128 Meg RAM for the price of…" OK, Pete. We get it.

One day Pete's friend Gena asks him for some computer help. Now, Gena has a Macintosh iBook, because she loves the translucent boxes. When Pete sits down and tries to use the Macintosh, he quickly gets frustrated. "I hate these things," he says. He is, finally, able to help Gena, but he's grumpy and unhappy. "The Macintosh has such a clunky user interface."

Clunky? What's he talking about? *Everybody knows* that the Macintosh has an elegant user interface, right? The very *paradigm* of ease-of-use?

Here's my analysis of this mystery.

On the Macintosh, when you want to move a window, you can grab any edge with the mouse and move it. On Windows, you must grab the title bar. If you try to grab an edge, the window will be reshaped. When Pete was helping Gena, he tried to widen a window by dragging the right edge. Frustratingly, the whole window moved, rather than resizing as he expected.

On Windows, when a message box pops up, you can hit enter *or* the space bar to dismiss the message box. On the Mac, space doesn't work. You usually need to click with the mouse. When Pete got alerts, he tried to dismiss them using the space bar, like he's been doing subconsciously for the last six years. The first time, nothing happened. Without even being aware of it, Pete banged the space bar harder, since he thought that the problem must be that the Mac did not register his tapping the space bar. Actually, it did — but it didn't care! Eventually he used the mouse. Another tiny frustration.

Pete has also learned to use Alt+F4 to close windows. On the Mac, this actually changes the *volume*. At one point, Pete wanted to click on the Internet Explorer icon on the desktop, which was partially covered by another window. So he hit Alt+F4 to close the window and immediately double-clicked where the icon would have been. The Alt+F4 raised the volume on the computer and didn't close the window, so his double click actually hit the Help button in the toolbar on the window which he wanted closed anyway, which immediately started bringing up a help window, so now, he's got *two* windows open which he has to close.

Another small frustration. But, boy, does it add up. At the end of the day, Pete is grumpy and angry. When he tries to control things, they don't respond. The space bar and the Alt+F4 key "don't work" — for all intents and purposes, it's as if those keys were broken. The window disobeys him when he tries to make it wider, playing a little prank where it just moves over instead of widening. Bad window. Even if the whole thing is subconscious, the subtle feeling of being out of control translates into helplessness, which translates into unhappiness. "I like my computer," Pete says. "I have it all set up so that it works exactly the way I like it. But these Macs are clunky and hard to use. It's an exercise in frustration. If Apple had been working on MacOS all these years instead of messing around with Newtons, their operating system wouldn't be such a mess."

Right, Pete. We know better. His feelings come *despite* the fact that the Macintosh really is quite easy to use — for Mac users. It's totally arbitrary which key you press to close a window. The Microsoft programmers, who were, presumably, copying the Mac interface, probably thought that they were adding a cool new feature by letting you resize windows by dragging any edge. The MacOS 8.0 programmers probably thought they were adding a cool new feature when they let you move windows by dragging any edge.

when they let you move windows by dragging any edge.

Most flame wars you read about user interface issues focus on the wrong thing. Windows is better because it gives you *more ways* to resize the window. So what? That's missing the point. The point is, does the UI respond to the user in the way in which the user *expected* it to respond? If it didn't, the user is going to feel helpless and out of control, the same way I felt when the wheels of the dough bathtub didn't turn the way I pushed them, and I bumped into a wall. Bonk.

UI is important because it affects the feelings, the emotions, and the mood of your users. If the UI is wrong and the user feels like they can't control your software, they *literally* won't be happy and they'll blame it on your software. If the UI is smart and things work the way the user expected them to work, they will be cheerful as they manage to accomplish small goals. Hey! I ripped a CD! It *just worked! Nice software! Wooooooooooo!*

To make people happy, you have to let them feel like they are in control of their environment. To do this, you need to *correctly* interpret their actions. The interface needs to behave in the way they are expecting it to behave.

Thus, the cardinal axiom of all user interface design:

> **A user interface is well-designed when the program behaves exactly how the user thought it would.**

As Hillel said, everything else is commentary. All the other rules of good UI design are just corollaries.

ABOUT THE AUTHOR.

In 2000 I co-founded Fog Creek Software, where we created lots of cool things like the FogBugz bug tracker, Trello, and Glitch. I also worked with Jeff Atwood to create Stack Overflow and served as CEO of Stack Overflow from 2010-2019. Today I serve as the chairman of the board for Stack Overflow, Glitch, and HASH.

← PREVIOUS POST
2000/04/10

NEXT POST →
2000/04/11