

DPI

De Programmatica *Ipsium*

DE PROGRAMMATICA IPSUM

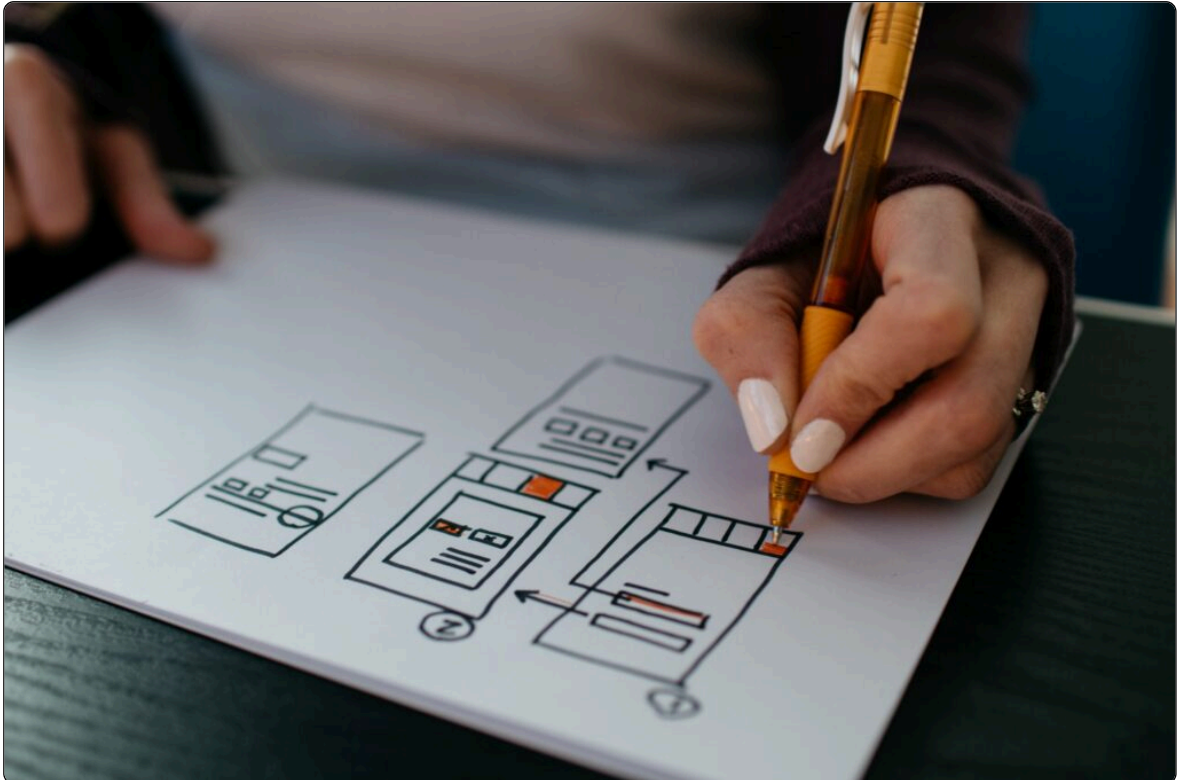
Issue 040:
Skeuomorphism

January 3rd, 2022

Table of Contents

Issue 040: Skeuomorphism	5
The Button And The Spoon	9
We Cannot Afford To Live Without It	15
Jenifer Tidwell	21

Issue 040: Skeuomorphism



January 3rd, 2022

Welcome to the fortieth issue of *De Programmatica Ipsum*, dedicated to the subject of *Skeuomorphism*. In this edition:

- Adrian calls for no more flat user interfaces¹.
- Graham argues that skeuomorphism is a necessity².
- In the Library section³, Adrian reviews “Designing Interfaces” by Jenifer Tidwell⁴.

Enjoy this issue! Please subscribe to our free newsletter⁵ to stay updated about new releases, share the articles on social media, or contribute⁶ if you would like to support our work.

ISSUE 040: SKEUOMORPHISM

Cover photo by Kelly Sikkema⁷ on Unsplash⁸.

REFERENCES

¹ <https://deprogrammaticaipsum.com/the-button-and-the-spoon/>

² <https://deprogrammaticaipsum.com/we-cannot-afford-to-live-without-it/>

³ <https://deprogrammaticaipsum.com/category/library/>

⁴ <https://deprogrammaticaipsum.com/jenifer-tidwell/>

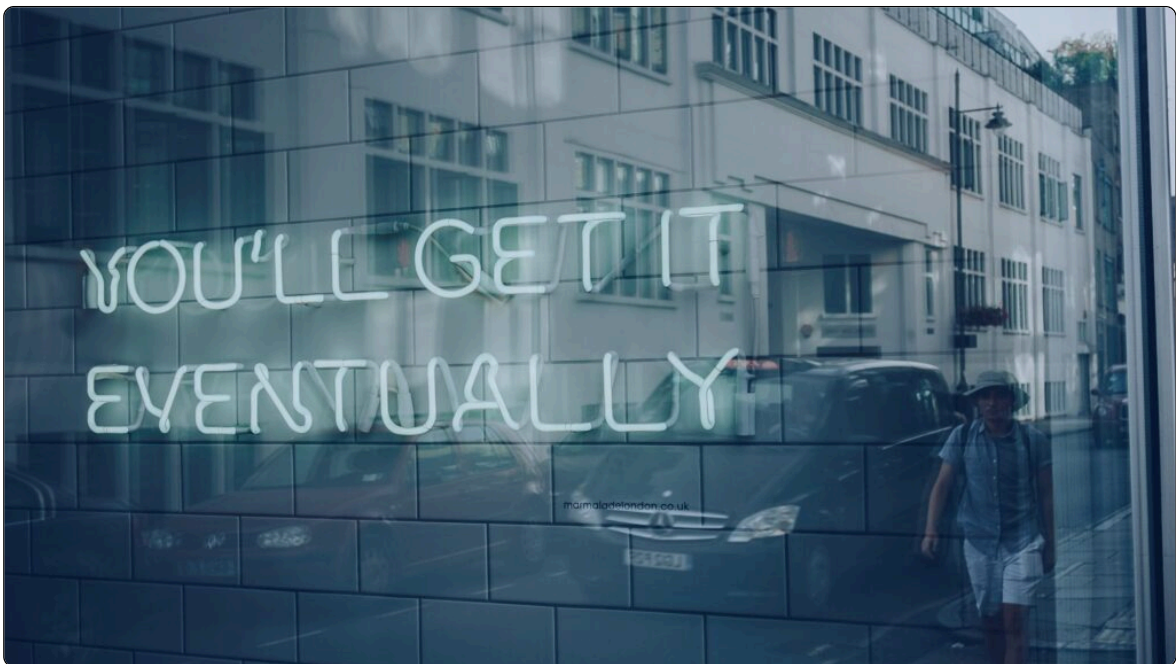
⁵ <https://deprogrammaticaipsum.com/newsletter/>

⁶ <https://deprogrammaticaipsum.com/contribute/>

⁷ https://unsplash.com/@kellysikkema?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

⁸ https://unsplash.com/s/photos/user-interface?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

The Button And The Spoon



By Adrian Kosmaczewski

As my father approaches his 80th birthday, his first contacts with the latest technologies become ever more interesting. As a pen-and-pencil architect and entrepreneur with 50 years of experience, technology is not a necessity; rather, just another useful tool. He setup his first fax machine at the end of the 90s, and got his first computer a decade after that. At some point in the 2010s he bought a smartphone, and one day asked me for help to finish some task on it. I remember his face when I told him to “touch the button that says ‘done’” and he replied, “which button? I do not see any button”.

In Spanish, just like in English, the word “button” (“botón”) can be used as both “push button” or “shirt button”. Being the son of an immigrant costume tailor who moved from Poland to Buenos Aires in the 1930s, my father’s brain first considered the latter meaning of the word “button”, before the former kicked in.

Truth is, on his smartphone, neither of these things were visible. Nor a push button, nor a shirt button, not a light switch, not a calculator button, not a door knob, not a push lever, nothing. What was visible was, instead, the blue word “Done” eerily floating in the top-right corner of the screen.

As my father said, there was no button.

Skeuomorpheus

As this article hits the press, a fourth installment¹ of the Matrix franchise has just been released on cinemas worldwide. In the world of Neo and Agent Smith, a shirt button looks like a shirt button and might behave like a push button. Since The Matrix has been purposely designed as the world that has been pulled over their eyes to blind them from the truth, it *must* be the quintessential skeuomorphic environment.

Here is an example: the Keymaker² (played by Randall Duk Kim³ in the second movie⁴) appears as a... well, a keymaker, sharpening keys as if he were in a Mister Minit⁵ shop. Looking at him at work, one can forget the complexity of the software keys represented by those entities; are they symmetric key pairs? Are they based on RSA or Ed25519⁶? Is the keychain in his belt stored in the KDBX format⁷? This movie supposedly happening in the future, one can imagine that the algorithms in use are very likely to be much more complicated than those we know today. But you get the idea. However they are implemented, those keys still look like metal keys, and they open doors. Or rather, backdoors.

As the kid in the waiting room of the Oracle said to Neo while doing his Uri Geller⁸ thing, there was no spoon.

Ceci n'est pas une Pipe

Affordances on a computer screen are mechanisms designed to prevent us from becoming crazy. We need analogies to navigate a world that is very alien to most of humanity. We programmers tend to forget this fact, but the very machine I am using to write these words is a marvel of complexity, one that downright baffles and confounds the vast majority of people walking this Earth.

Thirty years ago, user interfaces had all the effects required to provide a somewhat physical appearance to that which had not any. Search for screenshots of NeXTSTEP⁹, Motif¹⁰, Windows 95¹¹, Mac OS 9¹², and you will find lots of drop shadows and shitty 3D effects, where the light almost always seems to come from the upper left corner of your laptop. User interfaces were, in a quite literal way, deeper than they are today; which is why we call “flat” the current design fashion *in vogue* among UI designers.

Current research¹³ cannot reliably determine whether flat user interfaces are more or less effective than skeuomorphic ones; but it can provide some useful recommendations:

Flatness (and minimalism) should not be considered as a cause on its own, but as a means to foster easier and aesthetically satisfying interaction.

(Spiliotopoulos, Konstantinos & Rigou, Maria & Sirmakessis, Spiros. (2018). “A Comparative Study of Skeuomorphic and Flat Design from a UX Perspective.” *Multimodal Technologies and Interaction*. 2. 31. 10.3390/mti2020031.)

What remains, beyond all logic and rational considerations of measurable effectiveness, is satisfaction. And the biggest thing that disappeared in the transition from iOS 6 to iOS 7¹⁴ was, without any doubt, my own satisfaction at dealing with this new idea of a user interface.

There is a lot of research¹⁵ and explanations¹⁶ and controversy¹⁷ and criticism¹⁸ about the pros and cons of flat vs. skeuomorphic user interfaces. This magazine being a big opinion piece, this author roots for a (hopefully, not so far away) future in which buttons on a computer screen will look like (push, not shirt) buttons again.

Beauty

When my father started his career as an architect, at the end of the 1960s, he got an internship in the office of Jorge Horacio Lestard¹⁹, one of the most famous and widely recognized Argentine architects of the 20th century. Every week, in his office at the corner of Avenida Belgrano and Perú street, Mr. Lestard would gather his staff to review the progress of design projects, pitching various teams against each other, for some new design to be added to the Buenos Aires downtown skyline.

In one of those occasions, my father told me, Mr. Lestard inspected scale models laid on the conference room table, for longer than usual. Those were proposals for a new building of the Argentine Embassy in Asunción, the capital city of Paraguay. Mr. Lestard had been staring at them in silence for minutes, transfixed and without blinking, surrounded by his staff, all enthralled and waiting for his final verdict. At some point, Mr. Lestard raised his head and broke the silence, choosing a model from the bunch: “This one!” Among sighs and deep breaths, one of the managers came to him and asked, “why this one and not the others?” Mr. Lestard, who would become tenured professor at the Universidad de Buenos Aires in 1983, and future recipient of the 2002 Konex Award²⁰, simply shrugged, smiled, and replied.

“Because it is sooooooo pretty!”

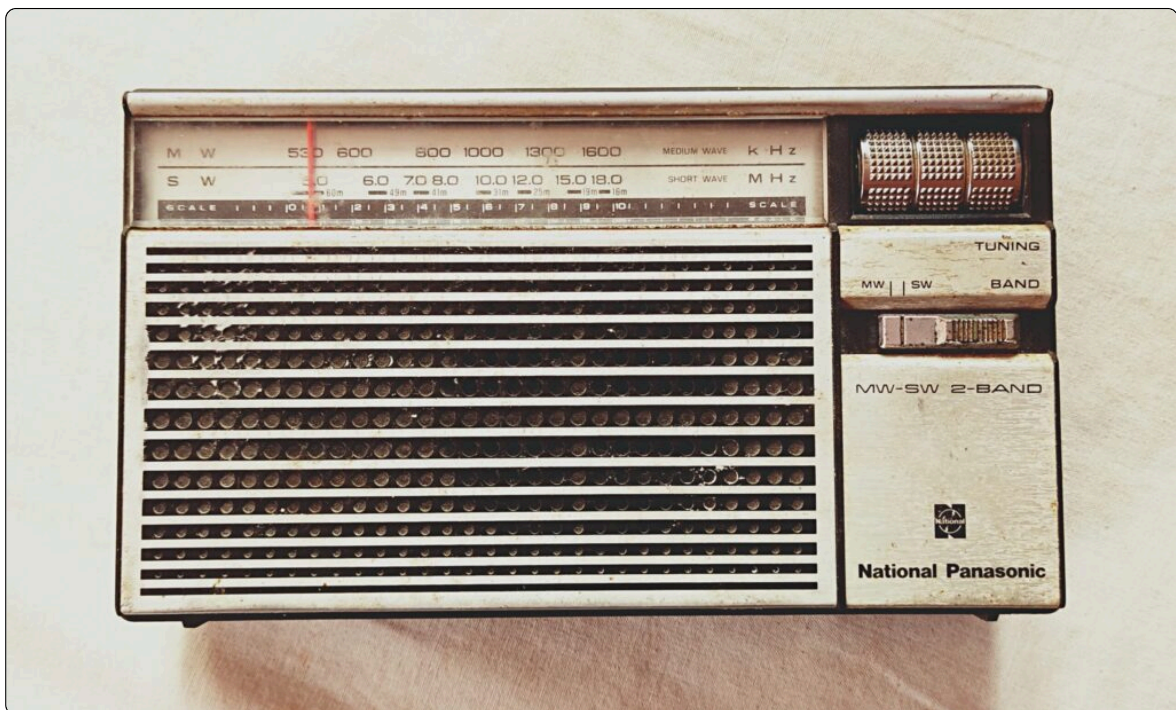
Google can run as many user groups and statistics and gather as much data as they want to take design decisions, but at the end of the day, the only thing that remains in our minds and hearts is the satisfaction provided by beauty. Computer screens, in that respect, should be treated as artisanal pieces²¹, both with purpose *and* aesthetics at the same time. The disconnection between designers and users has let user interfaces to drift off to a space where only designers are satisfied by them. It is time to remember the users trying to find those buttons, trying to bend those spoons, and to take *their* satisfaction into account, too.

Cover photo by Nigel Tadyanehondo²² on Unsplash²³.

REFERENCES

- ¹ <https://www.imdb.com/title/tt10838180/>
- ² <https://neoencyclopedia.fandom.com/wiki/Keymaker>
- ³ <https://www.imdb.com/name/nm0453641/>
- ⁴ <https://www.imdb.com/title/tt0234215/>
- ⁵ <https://misterminit.co/>
- ⁶ <https://en.wikipedia.org/wiki/EdDSA>
- ⁷ https://keepass.info/help/kb/kdbx_4.html
- ⁸ https://en.wikipedia.org/wiki/Uri_Geller
- ⁹ <https://en.wikipedia.org/wiki/NeXTSTEP>
- ¹⁰ [https://en.wikipedia.org/wiki/Motif_\(software\)](https://en.wikipedia.org/wiki/Motif_(software))
- ¹¹ https://web.archive.org/web/20081022103457/http://www.sigchi.org/chi96/proceedings/desbrief/Sullivan/kds_txt.htm
- ¹² https://en.wikipedia.org/wiki/Mac_OS_9
- ¹³ https://www.researchgate.net/publication/325563195_A_Comparative_Study_of_Skeuomorphic_and_Flat_Design_from_a_UX_Perspective
- ¹⁴ <https://akos.ma/blog/12-years-of-iphone-a-developers-perspective/#2013>
- ¹⁵ <https://designmodo.com/flat-design-principles/>
- ¹⁶ <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/affordances>
- ¹⁷ <https://www.nngroup.com/articles/flat-ui-less-attention-cause-uncertainty/>
- ¹⁸ <https://www.webfx.com/blog/web-design/when-flat-design-falls-flat/>
- ¹⁹ <https://www.modernabuenosaires.org/arquitectos/jorge-horacio-lestard>
- ²⁰ <https://www.fundacionkonex.org/b2404-jorge-lestard>
- ²¹ <https://deprogrammaticaipsum.com/a-brief-history-of-programming-artists/>
- ²² https://unsplash.com/@nxvision?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
- ²³ https://unsplash.com/collections/1372174/susan%E2%80%99s-confusion?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

We Cannot Afford To Live Without It



By Graham Lee

Skeuomorph is a word with two Greek bits in. “Morph” we see all the time, and it means “shape”. “Skeuo” means “tool”. So the word means “tool-shape”, but tools are not themselves considered skeuomorphic. An object is skeuomorphic, or has skeuomorphic features, if it has stylistic or design components reminiscent of an earlier necessity.

To what extent are any features in a computer UI skeuomorphic? If a notes application displays a yellow canvas with rules, is that a skeuomorphic aspect of the design, or is it just design? We could say that it is reminiscent of a legal notepad, which is an

idea Americans have and those pads are typically yellow. They are typically yellow because they always used to be yellow, and most yellow legal pads that Americans have used have been dyed. A long time ago they would have been yellow because the paper was unbleached. So is yellow legal notepaper skeuomorphic, or just “a design choice”? Is a yellow notes app skeuomorphic, if the thing it is copying did not need to be yellow? If it is not really a tool shape?

Please do not write in, the answer is completely inconsequential. I am interested in the design inheritances that are consequential, where the inherited design tells us how to use something by analogy with something else we know how to use.

Sometimes the things that we need and the things that we do not need are closely connected. We need the instrument cluster in a motor car to give us feedback about the state of the machine. It is usually mounted on the dashboard, which was introduced to stop pebbles and nails kicked up from the horses' hooves from hitting the driver. We do not particularly need that any more.

The kinds of interface analogies we benefit from in the computer world are analogies. Computer interfaces are completely unlike anything else we use. I mean, some of them are, they do not have to be. A modern motor car's controls are a computer interface, providing input to the engine control unit, anti-lock brake system, traction control system, and potentially other computers. But the analogue to a pre-computer motor car's controls is complete. But things like the desktop computer interface and the touch screen phone interface are completely novel, and any analogy we can grasp is useful.

Some of the analogies make sense in isolation, but fall apart when they are combined in one place on a computer. I have windows onto files in folders, said windows all being on top of the wallpaper that is decorating my desktop. I also keep my trash can in a dock on the desktop.

The point of these analogies is not just vestigial reminders of tools that preceded the computer: the only people who previously read my files through a window were my competitors. No, the point is to give affordances: entry points into cognition of the new tool through metaphor with existing, understood concepts. A window is a window in that it shows a view onto a scene (in our case, a document) which

may be complete or partial. A folder is a folder in that it is a place where you can group zero or more files.

These ideas of windows, desktops, files and folders are affordances for the kinds of people who had familiarity with the existing objects. The market for personal computers sold by the likes of Commodore, Apple, and IBM in the 1980s was affluent Westerners, typically those in white-collar occupations. They will not have trouble with folders on their desktops, but probably have somebody else to do their typing so may not be comfortable with the QWERTY keyboard.

But the reach of the computer has vastly outstripped the reach of the lifestyle depicted in *Mad Men* and the *Dilbert* comic strip. Both in space – people have computers who never had office jobs – and time – people who have office jobs now get computers and do not get paper filing systems. The affordances of the desktop-and-files metaphor are now the affordances of familiarity, not the affordances of metaphor.

Let me give a different example. I have a digital radio. It receives MPEG layer 2 data streams over the airwaves and lets me choose one of them to decode and listen to. I pick which channel I hear through a rotary knob on the front panel: not as the easiest way to select from hundreds of channels; not as a throwback to the cat's whisker radiogram from the 1920s; but because that is now the expected user interface for radios. So it is with files and folders.

It is tempting to think that we can do away with these affordances now because nobody uses the original object from which the affordance takes form. Just as the floppy disk is now the “3D-printed save icon”, so the filing cabinet is now the Society for Creative Anachronism's USB stick. But we have to be careful. These affordances are not only analogies for people new to the computer, they are now increasingly analogies to existing computers for people familiar with the computer.

Innovation¹ in user interface is of course possible, and can be beneficial. The now ubiquitous “pull to refresh”² gesture was introduced by Loren Brichter³ in Tweetie⁴ 2.0 over a decade ago. But it also relies on analogy to explain its operation: you are looking at a list of tweets, arranged chronologically (back in the days before *The Algorithm*) from newest downward. So anything newer than the most recent

tweet you can see must involve scrolling up from the top: pull down to see newer things.

If you confuse affordance with skeuomorph then you will throw away useful hints because they look like vestigial anachronisms. A button has bevels both so that it looks like a button on a 1970s transistor radio, and so that it is distinguishable from non-interactive UI. Multicoloured icons increase distinguishability as well as fidelity.

A flat UI becomes an expensive, inflexible sheet of paper, when it can be so much more. Beware of discarding utility in the name of novelty.

Cover photo by Chandan Chaurasia⁵ on Unsplash⁶.

REFERENCES

¹ <https://deprogrammaticaipsum.com/issue/issue-036-innovation/>

² <https://en.wikipedia.org/wiki/Pull-to-refresh>

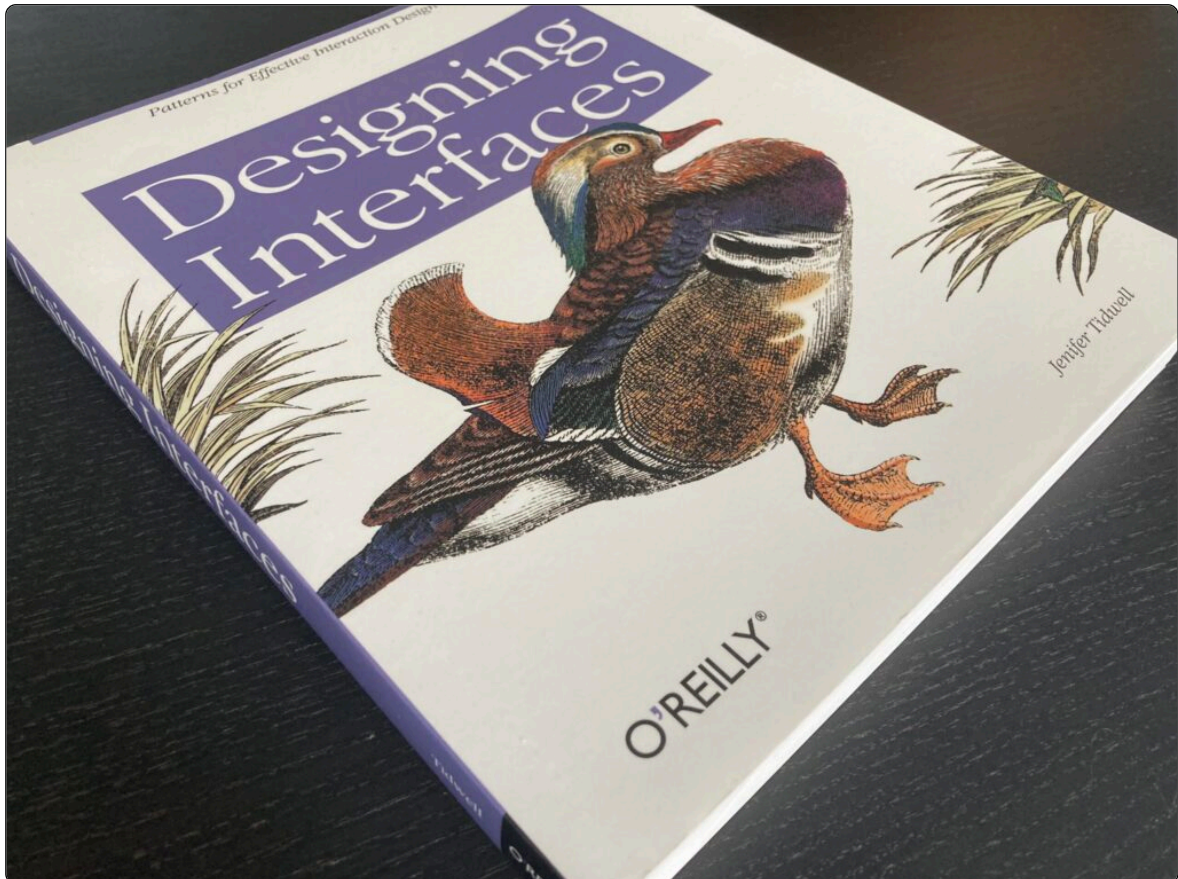
³ https://en.wikipedia.org/wiki/Loren_Brichter

⁴ <https://en.wikipedia.org/wiki/Tweetie>

⁵ https://unsplash.com/@chaurasia?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

⁶ https://unsplash.com/s/photos/transistor-radio?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Jenifer Tidwell



By Adrian Kosmaczewski

If there is one thing that computer books are most definitely not usually praised for, it is their visuals. Thankfully, books about user experience and user interface design are usually, indeed, worthy of such acclaim. In this case, however, limiting a review to such criteria would be short-sighted, poor, and unjust. The truth is that most important literature works are multi-layered, profound, and suitable for multiple relectures.

In 2007, when Jenifer Tidwell¹ published the first edition of her book “Designing Interfaces” (its third edition² was published January 2020), we had been, as an in-

dustry, making visual user interfaces on bitmap displays for around 45 years. Well, maybe not for *that* long. Yes, it all started with Ivan Sutherland's "Sketchpad"³ PhD thesis at the MIT, but GUIs remained a niche curiosity for two decades, while Douglas Engelbart⁴, Alan Kay⁵, and Jef Raskin⁶ worked on it. It was not until the Mac in the 1980s, and later Windows and the Web in the 1990s, that GUIs became what they are today.

So it is fairer to say that Ms Tidwell's book encapsulates roughly the first quarter century of GUI knowledge ever produced by the human race; quite an accomplishment in itself.

All of this knowledge is organized as a collection or catalog of "patterns"; by far one of the most hyped⁷ words in programming literature between 1995 and 2007. Starting with the overall architecture of a user interface in panels and windows, Ms Tidwell drilled down to the most common elements, such as preview panes, button groups, autocompletion, breadcrumbs, treemaps, and more. All in all, almost 100 different patterns; 94 to be exact. Each profusely illustrated with examples from the Motif, Xerox Star, Mac ("Classic" and OS X), and Windows galaxies, but also showcasing PDA screens, and even examples of popular cellphones of the pre-iPhone era.

The first layer of contact with the book, the visual production, hits your brain as a high-speed train as soon as you have it in your hands; it is hard to browse this gem and not dream of designing the next laureate of the Apple Design Awards⁸. Few books in our craft produce such effect. It is refreshing.

On a second appreciation, comes to the reader the size and breadth of the catalog. Similarly to other books⁹ built around the "pattern" moniker, Ms Tidwell crafted a reference book that has its place next to the keyboard and mouse of anyone creating software for a living or for pleasure. The logic behind each visual element is clearly explained, and documented with examples coming from the most radically different environments, some of which are of historical relevance at this point.

The third, and in the opinion of this author, the most important takeaway from the book is not the (gorgeous) visual design, nor the immense wisdom of the pattern catalog; it is rather in the following statement in the opening pages:

The first step in designing an interface is figuring out what its users are really trying to accomplish.

Ah, the most important question, the one that many still fail to answer before 1.0 hits the road.

2007 was a time of brushed metal¹⁰ interfaces on Aqua¹¹; of translucent Windows Vista title bars on Aero¹²; of Web 2.0 candy-like UIs with shadows and gradients; and of the CSS Zen Garden¹³. It was also the year of the unveiling of the iPhone, arguably¹⁴ the most successful product of all time, and one that had quite an impact in the design of space-constrained user interfaces. The first edition of this book, of course, does not mention it at all, and understandably so.

Fifteen years later, we live in a world where Atwood's Law¹⁵ became the norm, where interfaces are reactive (whatever that means), and where "flat" is a shiver-inducing visual trend. Here is this author hoping that the creators of the next Electron app will get a copy of Ms Tidwell's book, dive into it, and find out that there is indeed an answer for pretty much every single question they might have. User interfaces do not require any more shallow, mindless "innovation"¹⁶ at this point.

For those programmers interested in visual design, this author recommends coupling this book with Jeff Johnson's funny "GUI Bloopers"¹⁷ (1999), David Kadavy's excellent "Design for Hackers"¹⁸ (2011), and most importantly, with the review¹⁹ in volume 40, issue 2 of the IEEE Annals of the History of Computing²⁰ of the work of Susan Kare²¹.

Cover photo by the author.

REFERENCES

- ¹ <https://www.linkedin.com/in/jenifertidwell/>
- ² <https://www.oreilly.com/library/view/designing-interfaces-3rd/9781492051954/>
- ³ <https://dspace.mit.edu/handle/1721.1/14979>
- ⁴ https://en.wikipedia.org/wiki/Douglas_Engelbart
- ⁵ <https://deprogrammaticaipsum.com/what-smalltalk-was-not/>
- ⁶ <https://deprogrammaticaipsum.com/jef-raskin/>
- ⁷ <https://deprogrammaticaipsum.com/issue/issue-001-hype/>
- ⁸ https://en.wikipedia.org/wiki/Apple_Design_Awards
- ⁹ <https://deprogrammaticaipsum.com/kathy-sierra/>
- ¹⁰ [https://en.wikipedia.org/wiki/Brushed_metal_\(interface\)](https://en.wikipedia.org/wiki/Brushed_metal_(interface))
- ¹¹ [https://en.wikipedia.org/wiki/Aqua_\(user_interface\)](https://en.wikipedia.org/wiki/Aqua_(user_interface))
- ¹² https://en.wikipedia.org/wiki/Windows_Aero
- ¹³ https://en.wikipedia.org/wiki/CSS_Zen_Garden
- ¹⁴ <http://www.asymco.com/2019/05/16/the-pivot/>
- ¹⁵ <https://deprogrammaticaipsum.com/innovationscript/>
- ¹⁶ <https://deprogrammaticaipsum.com/issue/issue-036-innovation/>
- ¹⁷ <https://www.amazon.com/gp/product/1558605827/>
- ¹⁸ <https://www.amazon.com/Design-Hackers-Reverse-Engineering-Beauty-ebook/dp/B005J578EW>
- ¹⁹ <https://www.computer.org/csdl/magazine/an/2018/02/man2018020048/13rRUx0xPNH>
- ²⁰ <https://www.computer.org/annals>
- ²¹ https://en.wikipedia.org/wiki/Susan_Kare