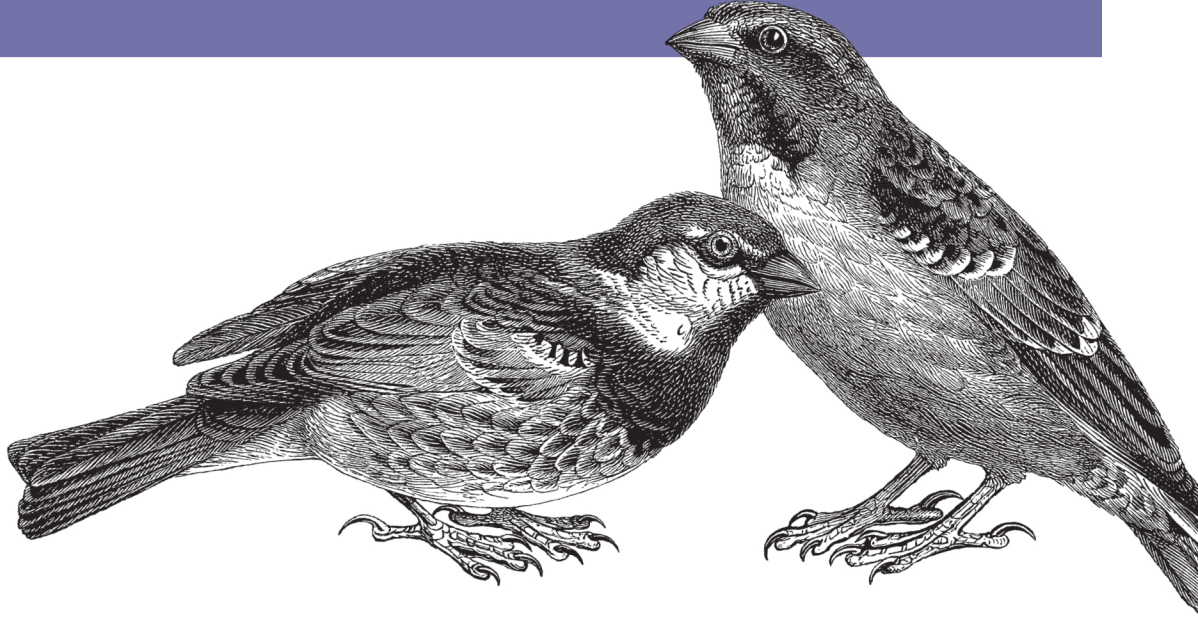
Designing with Details

Microinteractions



Dan Saffer

Foreword by Don Norman

O'REILLY®

CHAPTER 4
Feedback

A 56-year-old man punched his fist through the glass and into the electronics of the machine. "Yes, I broke the machine and I'd do it again," he told the security guards. (He was sentenced to 90 days in jail.) Another man, 59-year-old Douglas Batiste, was also arrested for assaulting a machine—by urinating on it. A woman caused $1,800 in damages to another machine by slapping it three times.[1] And 67-year-old Albert Lee Clark, after complaining to an employee and getting no satisfaction, went to his car and got his gun. He came back inside and shot the machine several times.[2]

What device is causing so much rage? Slot machines.

Slot machines are a multi-billion-dollar business. Slot machines take in $7 out of every $10 spent on gambling. Collectively, the money they generate is in the tens of billions,

1. Nir, Sarah Maslin, "Failing to Hit Jackpot, and Hitting Machine Instead," The *New York Times*, July 13, 2012.
2. "Man charged with shooting slot machine," Associated Press, February 13, 2012.

far surpassing the revenue of other forms of entertainment, such as movies, video games, and even pornography.[3] The reason that slot machines—microinteraction devices for sure—work so well at taking money from people is because of the feedback they provide. Most (read: all) of this feedback is insidious, designed specifically to keep people playing for as long as possible.

If you are the statistical anomaly who has never seen or played a slot machine, they work like this: you put coins, bills, or (in newer machines) paper tickets with barcodes into the machine. Pushing a button, tapping the touchscreen, or pulling a lever (the trigger) causes three (or more) seemingly independent "tumblers" to spin. When they stop spinning after a few seconds, if they are aligned in particular ways (if the symbols are the same on all three tumblers, for example), the player is a winner and money drops out of the slot machine. A committed player can do a few hundred (!) spins in an hour.

What really happens is that the rules are rigged in the slot machine's favor; statistically, the slot machine will never pay out more than 90%, so the tumblers never "randomly" do anything, although the feedback makes it seem that way. If the tumblers actually worked the way they appear to work, the payback percentage would be 185% to 297% —obviously an undesirable outcome for casino owners. The outcome is "random but weighted." Blank spaces and low-paying symbols appear more frequently than jackpot symbols—that is, less frequently than they would if the tumbler were actually (instead of just seemingly) random. Thanks to the feedback they get, players have no idea what the actual weighting is; an identical model can be weighted differently than the machine next to it. Since modern slot machines are networked devices, the weighting can even be adjusted from afar, on the fly.[4]

No matter how players trigger the tumblers—by pulling the lever harder, for example —players cannot influence or change the outcome. Some slot machines also have a stop button to stop the tumbler "manually" while they spin. This too doesn't affect the outcome; it only provides an illusion of control.[5]

Not only are the tumblers weighted to prevent winning, but they are designed to incite what gambling researcher Kevin Harrigan calls the Aww Shucks Effect by frequently halting on a "near win," or a failure that's close to a success (see Figure 4-1). For example, the first two tumblers show the same symbol, but the third is blank. These near wins occur 12 times more often than they would by chance alone. Research has shown that

3. Rivlin, Gary, "The Tug of the Newfangled Slot Machines," The *New York Times*, May 9, 2004.

4. Richtel, Matt, "From the Back Office, a Casino Can Change the Slot Machine in Seconds," The *New York Times*, April 12, 2006.

5. All from Kevin Harrigan's "The Design of Slot Machine Games," 2009.

near wins make people want to gamble more by activating the parts of the brain that are associated by wins—even though they didn't win![6]

Figure 4-1. An example of a "near win." (Courtesy Marco Verch.)

When a player does win, the win is usually small, although the feedback is disproportionate to the winning, so that players think they've won big. Lights flash, sounds play. And the sounds! In the *New York Times* profile of slot machine designer Joe Kaminkow, it notes:

> Before Kaminkow's arrival, [slot machine manufacturer] I.G.T.'s games weren't quiet—hardly—but they didn't take full advantage of the power of special effects like "smart sounds"—bright bursts of music. So Kaminkow decreed that every action, every spin of the wheel, every outcome, would have its own unique sound. The typical slot machine featured maybe 15 "sound events" when Kaminkow first arrived at I.G.T. [in 1999]; now that average is closer to 400. And the deeper a player gets into a game, the quicker and usually louder the music.[7]

6. Clark, L, Laurence, A., Astley-Jones, F., Gray, N., "Gambling near-misses enhance motivation to gamble and recruit brain-related circuitry," *Neuron* 61, 2009.
7. Rivlin, Gary, "The Tug of the Newfangled Slot Machines." The *New York Times*.

The slot machine microinteraction is so addictive because it provides, via feedback, *intermittent reinforcement of behavior*. Slot machine players keep performing the same behavior until they are eventually rewarded. With slot machines, if payout was predictable—if the player won every other time, for example—players would quickly get bored or annoyed. What keeps people playing is the very unpredictability of the payouts, plus the promise that very rarely there will be a big jackpot. In general, this is *not* the kind of reinforcement you want for most microinteractions, where you want consistent feedback with positive reinforcement (via feedback) of desirable behavior. Predictability is desirable.

Slot machines teach us that feedback is extremely powerful and can make or break a microinteraction. Visuals and sound combine to make an engaging experience out of what could be a repetitive, dull activity of pulling a lever over and over. Obviously, they do this to their mind-blowingly lucrative benefit and you certainly don't want every microinteraction being like a flashing, noisy slot machine, but the lesson is the same: feedback provides the character, the personality, of the microinteraction.

Feedback Illuminates the Rules

Unlike slot machines, which are designed to deliberately obscure the rules, with microinteractions the true purpose of feedback is to help users understand how the rules of the microinteraction work. If a user pushes a button, something should happen that indicates two things: that the button has been pushed, and what has happened as a result of that button being pushed (Figure 4-2). Slot machines will certainly tell you the first half (that the lever was pulled), just not the second half (what is happening behind the scenes) because if they did, people probably wouldn't play—or at least not as much. But since feedback doesn't have to tell users how the microinteraction *actually* works—what the rules actually are—the feedback should be just enough for users to make a working mental model of the microinteraction. Along with the affordances of the trigger, feedback should let users know what they can and cannot do with the microinteraction.

One caveat: you can have legitimate, nondeceitful reasons for not wanting users to know how the rules work; for example, users may not need to know every time a sensor is triggered or every time the device goes out to fetch data, only if something significant changes. For example, you don't often need to know when there is no new email message, only when there is a new one. *The first principle of feedback for microinteractions is to not overburden users with feedback*. Ask: what is the least amount of feedback that can be delivered to convey what is going on (Figure 4-3)?

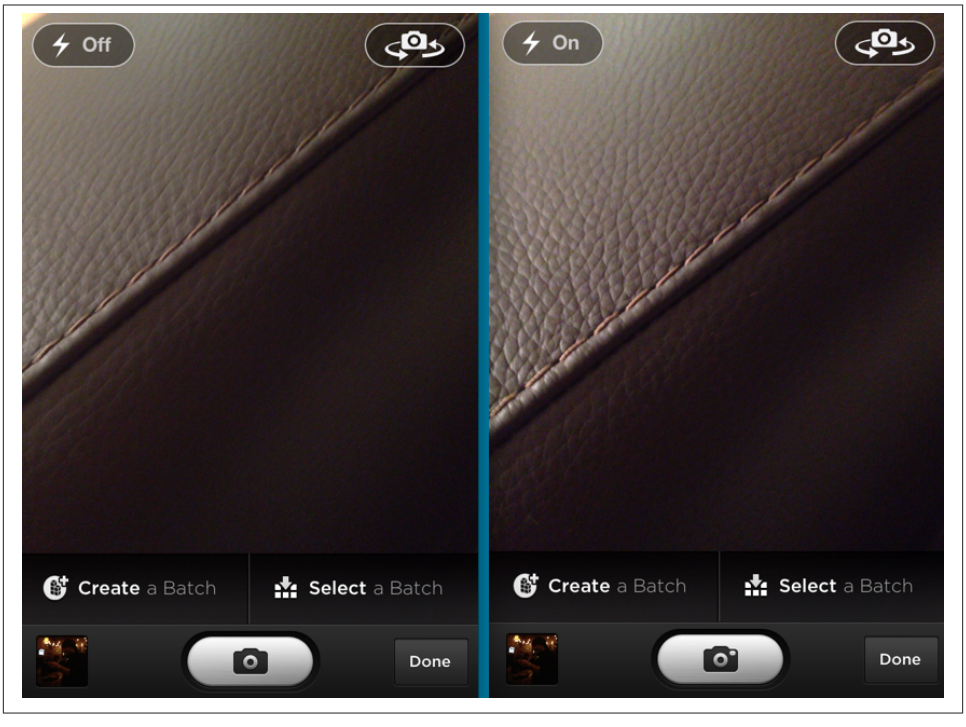


Figure 4-2. In Batch, when the flash is on, the camera icon on the shutter button gets a white flash indicator. (Courtesy Little Big Details.)

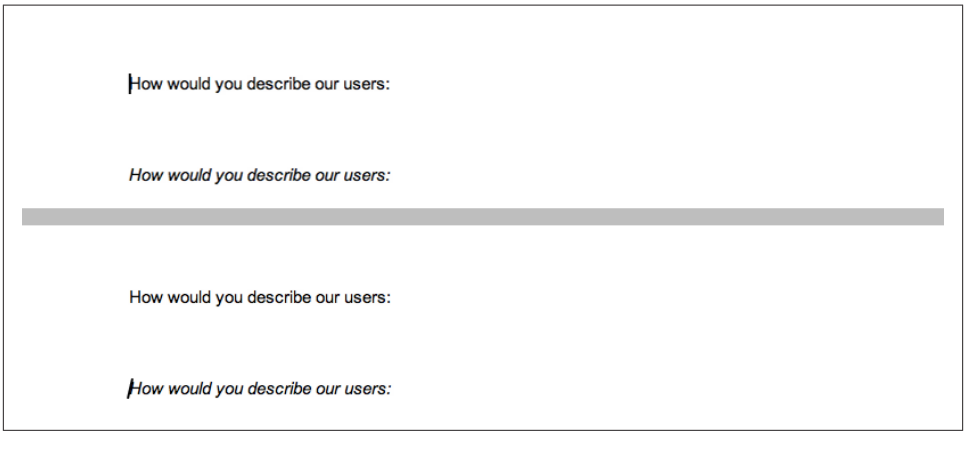


Figure 4-3. Google Docs slants the cursor when you're typing in italics. Microsoft Word does this as well. (Courtesy Gregg Bernstein and Little Big Details.)

Feedback should be driven by need: what does the user need to know and when (how often)? Then it is up to the designer to determine what format that feedback should take: visual, audible, or haptic, or some combination thereof (see Figures 4-4 and 4-5).

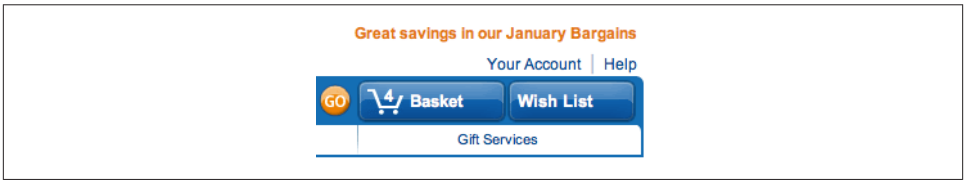


Figure 4-4. Amazon puts the item counter inside the shopping cart button. (Courtesy Matthew Solle and Little Big Details.)

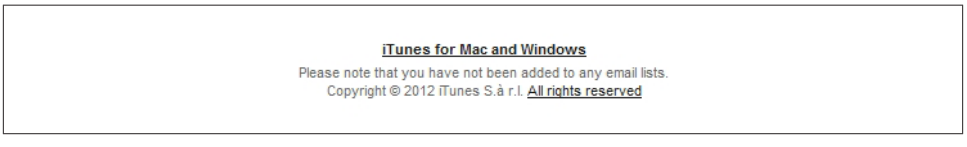


Figure 4-5. Sometimes it's important to indicate what didn't happen. When recommending an app via email, Apple's App Store tells you that you haven't been added to any email lists. (Courtesy Little Big Details.)

Feedback should occur:

- *Immediately after a manual trigger or following/during a manual adjustment of a rule*. All user-initiated actions should be accompanied by a system acknowledgment (see Figure 4-6). Pushing a button should indicate what happened.
- *On any system-initiated triggers in which the state of the microinteraction (or the surrounding feature) has changed significantly*. The significance will vary by context and will have to be determined on a case-by-case basis by the designer. Some microinteractions will (and should) run in the background. An example is an email client checking to see if there are new messages. Users might not need to know every time it checks, but will want to know when there are new messages.
- *Whenever a user reaches the edge (or beyond) of a rule*. This would be the case of an error about to occur. Ideally, this state would never occur, but it's sometimes necessary, such as when a user enters a wrong value (e.g., a password) into a field. Another example is reaching the bottom of a scrolling list when there are no more items to display.
- *Whenever the system cannot execute a command*. For instance, if the microinteraction cannot send a message because the device is offline. One caveat to this is that multiple attempts to execute the command could occur before the feedback that

something is amiss. It might take several tries to connect to a network, for example, and knowing this, you might wait to show an error message until after several attempts have been made.

- *Showing progress on any critical process, particularly if that process will take a long time*. If your microinteraction is about uploading or downloading, for example, it would be appropriate to estimate duration of the process (see Figure 4-7).

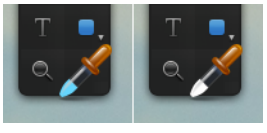


Figure 4-6. Pixelmator's eyedropper tool shows you the color you've chosen inside the pipette. (Courtesy Little Big Details.)

Feedback could occur:

- *At the beginning or end of a process*. For example, after an item has finished downloading.
- *At the beginning or end of a mode or when switching between modes* (Figure 4-8).

Figure 4-7. Transmit 4 shows in one progress bar both the total transfer and individual transfers. (Courtesy Stef van der Feen and Little Big Details.)

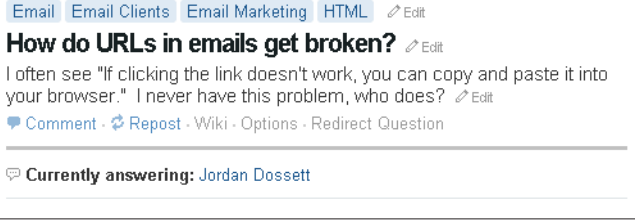


Figure 4-8. On Quora, you can see if someone is answering the question you're looking at. (Courtesy Allison Ko and Little Big Details.)

Always look for moments where the feedback can demystify what the microinteraction is doing; without feedback, the user will never understand the rules.

Feedback Is for Humans

While there is certainly machine-to-machine feedback, the feedback we're most concerned with is communicating to the human beings using the product. For microinteractions, that message is usually one of the following:

- Something has happened
- You did something
- A process has started
- A process has ended
- A process is ongoing
- You can't do that

Once you know what message you want to send, the only decisions remaining are how these messages manifest, as in Figures 4-9 through 4-11. The kind of feedback you can provide depends entirely upon the type of hardware the microinteraction is on. On a mobile phone, you might have visual, audible, and haptic feedback possible. On a piece of consumer electronics, feedback could only be visual, in the form of LEDs.

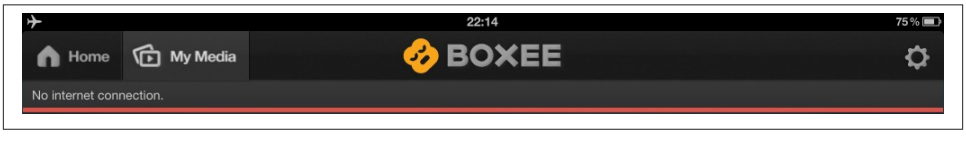


Figure 4-9. Humans respond to faces. The Boxee logo turns orange and "goes to sleep" when there is no Internet connection. (Courtesy Emil Tullstedt and Little Big Details.)

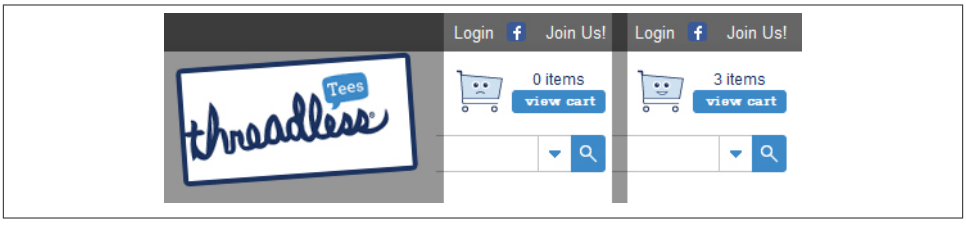


Figure 4-10. The Threadless shopping cart face turns from frowning to happy when you put items in it. (Courtesy Ahmed Alley and Little Big Details.)

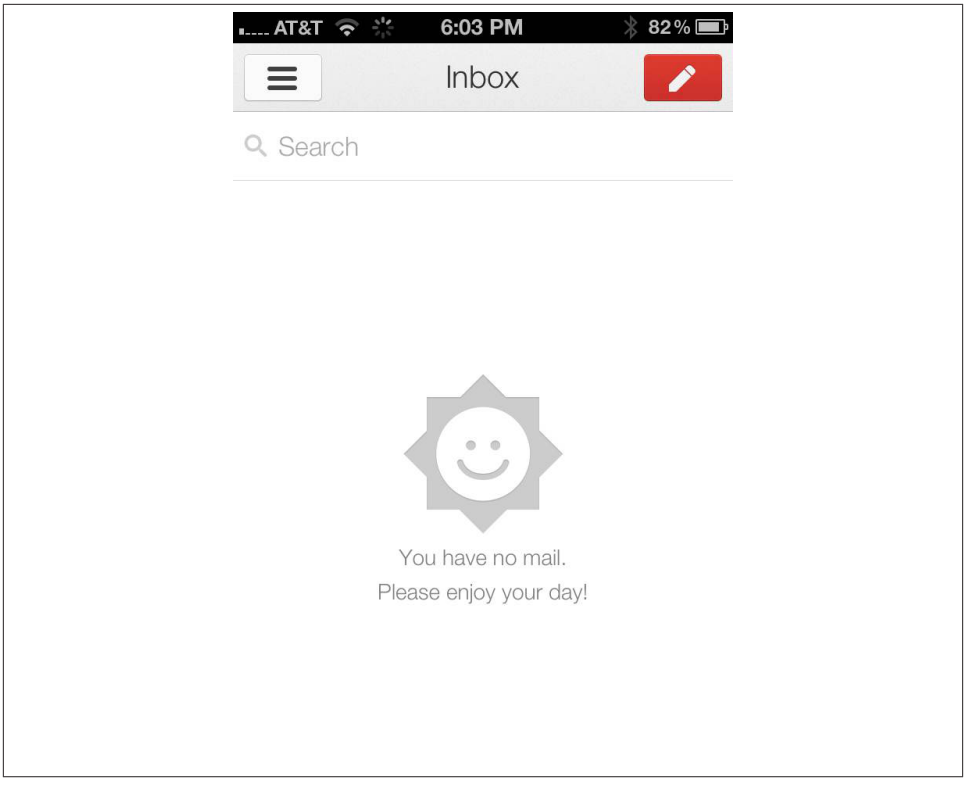


Figure 4-11. The Gmail iPhone app shows what not to do: randomly put a smiley face for a message that isn't necessarily a happy one. (Courtesy Steve Portigal.)

Let's take a microinteraction appliance like a dishwasher as an example. The dishwasher process goes something like this: a user selects a setting, turns the dishwasher on, the dishwasher washes the dishes and stops. If someone opens the dishwasher midprocess, it complains. Now, if the dishwasher has a screen, each of these actions could be accompanied by a message on the screen ("Washing Dishes. 20 minutes until complete."). If there is no screen, there might be only LEDs and sounds to convey these messages. One option might be that an LED blinks while the dishwasher is running, and a chime sounds when the washing cycle is completed.

Text (written) feedback is not always an option (for example, if there is no screen or simply no screen real estate). Once we move past actual words—and let's not forget that a substantial portion of the planet's population is illiterate: 793 million adults, according to the Central Intelligence Agency—we have to convey messages via other means: sound, iconography, images, light, and haptics. Since they are not text (and even words can be vague and slippery), they can be open to interpretation. What does that blinking LED mean? When the icon changes color, what is it trying to convey? Some feedback is clearly

learned over time: when that icon lights up and I click it, I see there is a new message. The "penalty" for clicking (or acting on) any feedback that might be misinterpreted should be none. If I can't guess that the blinking LED means the dishwasher is in use, opening the dishwasher shouldn't spray me with scalding hot water. In fact, neurologically, errors improve performance; how humans learn is when our expectation doesn't match the outcome.

The second principle of feedback is that the best feedback is never arbitrary: it always exists to convey a message that helps users, and there is a deep connection between the action causing the feedback and the feedback itself. Pressing a button to turn on a device and hearing a beep is practically meaningless, as there is no relationship between the trigger (pressing the button) or the resulting action (the device turning on) and the resulting sound. It would be much better to either have a click (the sound of a button being pushed) or some visual/sound cue of the device powering up, such as a note that increases in pitch. Arbitrary feedback makes it harder to connect actions to results, and thus harder for users to understand what is happening. The best microinteractions couple the trigger to the rule to the feedback, so that each feels like a "natural" extension of the other.

Less Is More

The more methods of feedback you use, the more intrusive the feedback is. An animation accompanied by a sound and a haptic buzz is far more attention getting than any of those alone. *The third principle for microinteractions feedback is to convey the most with the least*. Decide what message you wish to convey ("Downloading has begun") then determine what is the least amount of feedback you could provide to convey that message. The more important the feedback is, the more prominent (and multichannel) it should be (Figure 4-12).

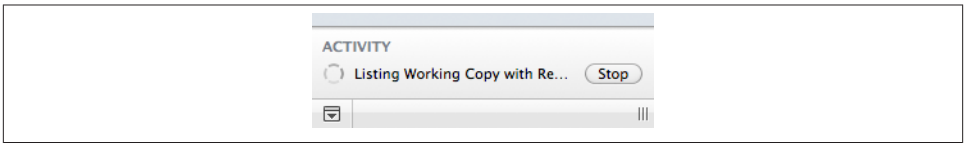


Figure 4-12. In Cornerstone, the number of segments in the spinning activity wheel are equal to the number of processes happening in the background. (Courtesy Yusuf Miles and Little Big Details.)

The fourth principle of feedback is to use the overlooked as a means of message delivery. Many microinteractions contain conventional parts of any interface—as they should. These overlooked parts of the UI—scrollbars, cursors, progress bars, tooltips/hovers, etc.—can be used for feedback delivery. This way, nothing that isn't already there will get added to the screen, but it can communicate slightly more than is usual