blink

By Jake Fleisher

Designing system response and feedback for these devices requires different consideration than “traditional” user interface systems. As these project types proliferate, we’re establishing a growing library of best practices, tips, and tricks. Here are five things we think about, discuss, and practice when we’re working in the domain of hardware + NUI.

Does the project involve dedicated hardware? Great! Hold on, it’s going to be a fun ride

When a client wants to develop a project with “dedicated hardware,” we get excited. Some devices are great at the one-size fits all (e.g., cell phone, laptop, etc.), but there are other devices that are hyper-focused and shouldn’t be any other way (e.g., very specific tools like stud finders and circuit testers, my stovetop espresso maker, my favorite kitchen knife, etc.). Working on a use- or scenario-specific device allows research and design to really focus on creating a great experience unencumbered by having to wear “many hats.” Guaranteed, walking the path of dedicated hardware and NUI is more difficult, with more variables, more unknowns, a longer timeline, and with greater chance for stumbles and mid-course “adjustments.” But, it is much more interesting to have those problems. It comes with its own problems and challenges that are unique to that mechanical and physical design, and they have to be worked out by you. Make sure you have access to people (designers, researchers, engineers, etc.) who have physical human factors (a.k.a. ergonomics) and cognitive human factors experience. This is how you can slay the hardware dragon challenges with alacrity and skill.

The 50 millisecond rule

Speaking of physical and cognitive human factors, our research has found that generally, latencies below 50 milliseconds (five hundredths of a second), are perceived as instant. It turns out this is pretty important, especially when considering elements with heavy influence on user experience such as input modality, system response time and type, battery life requirements, thermal management, and a host of others. We get the chance to apply first-hand knowledge about latency in specifying hardware performance (which has a direct corollary to user experience).

For some types of experiences, it is crucial that users not detect latency, while simultaneously allowing the hardware to maximize latency. Minimizing latency comes with a direct cost: chip speed requirement, power consumption, heat generation, and battery life are all affected negatively. Based on extensive usability testing we are able to make specific, quantitative performance recommendations that allow our clients to optimize the hardware design and user experience.

System input and feedback: More ≠ better

The desire to add more stuff to a product remains strong for many, even in the presence of good reason and evidence that says otherwise. When adding more stuff extends to slapping on additional system feedback or behavior, the user experience becomes really inelegant, really fast. Think about it: does that microwave really need to beep for each keypad press when the display indicates the system is receiving your input?

We sometimes encounter an engineering team who want to add an indicator to show that the system is “registering” the user’s input and responding appropriately. It is well-intentioned, of course, but it can be superfluous at best and highly annoying at worst. The idea would be something like adding an indicator light to a lamp to “show” you that it’s working: totally redundant and unneeded. Just because you can doesn’t mean you should.

Beware the delta between prototype performance and production behavior

It’s difficult or just downright impossible to get valid feedback data from users if the prototype doesn’t accurately reflect what production behavior will be like. This is important for all types of products, from the purely digital (think: apps) to purely physical (think: bicycle seats) and everything in-between. The more “physical” a product is, the more important it is to be aware of any differences between your prototype and what the final product will be like.

Balancing speed of prototype creation and revision against its performance accuracy is what it all boils down to. It’s critically important for the project team to have solid experience in materials and manufacturing processes and just as crucial to have developers to guide implementation of digital products.

Interestingly, the problem that the prototype performs better than a production version is at least as common as the other way around. In a case where your prototype presents better performance than the production version will, nasty surprises lurk when the real product starts coming off the line. “Designing in” tolerances and “faults” in a prototype that accurately reflect any limitations or compromises that will be part of the manufactured product is critical to sound feedback and valid data.

We once gathered feedback for a client using a prototype that involved a spring-loaded button.

The user was required to hold down this button while using the product. We established the users' acceptable range of motion and input effort for this critical button using several versions of our client's prototypes. These specifications turned out to be unsupportable using the materials and processes our client had hoped to use. Our client was able to make some changes and also made the difficult decision whether to "flex" on some of the specifications we developed for a great user experience. They made an informed decision, and as a result of their expertise and our research, there were no surprises in the production version of the product.

There's never enough time to do it right, but there's always time to fix it afterwards

This is a well-worn design and engineering aphorism, but it never goes out of date.

Roughly, it means when you insist on an unreasonable development schedule (because, you know, it has to be done now), you'll probably spend all that time you saved (and then some) after the project ships when the whole thing is going south because it's unusable, uncomfortable, breaks, and people hate it. Just think of all the "what were they thinking?" experiences you've had with products, services, or messages out there: I guarantee that all of them, to some extent or more, can be summed up with this old, but accurate little ditty.

It's one thing to go live with a digital design that you can subsequently fix (e.g., changing a call-to-action button location, size, or color on a website) – that's okay and it's a great way to do the best testing possible: with real users in real scenarios. It's much more difficult to adjust or change hardware components after the production process has started. The desire to keep to a schedule can overshadow sound decision-making (pushing ship date) and bite you later on.

The best way to address this is in the very beginning of a program: is there enough time to codify user needs and expectations? Is there enough time to define minimum user experience performance requirements? If it's a brand-new product and mechanism, it's going to take longer than you think or want it to. Account for this.

The wrap

Purpose-specific digitally-enabled hardware can provide superior user experiences over a more general, one-size-fits-all hardware device, and developing hardware is straight-out fun because it's a unique challenge that has never been solved before. It can be gratifying if you know where the potential pitfalls are and how they differ from products that are digital-only. Surround yourself with people who have experience and give the program a realistic schedule to make sure your end result is useable, useful, and desirable. You'll save time, money, and heartache in the long run.

> *Jake does strategy work at Blink. In his spare time he's a gearhead, hack musician, and fan of nighttime bike rides.*