Some thoughts on touch screen user interface design

Creating intuitive natural interfaces

Willem L. Middelkoop

June 24, 2020

What would you like to do?

write without interference

I hold Microsoft Word responsible for inhibiting the flow of language, for imposing arbitrary standards upon the nature of text, for pushing technologies of editing and revision where they don't belong

Past weeks I have been working on a new app involving personal health. The main challenge is to come up with an intuitive, natural interface that works well on small touch screen devices. But what is it that makes an touch screen interface any good? Read along for some best practices.

Mouse and keyboard heritage

Traditional PC application interfaces have been primarily designed and developed with distinct input devices in mind, like the keyboard and mouse. Many conventions, best practices and design patterns have their link to this past. This is all fine when you are desiging an app for traditional PC's, but chances are that your target audience is using modern touch screen devices these days. It would be a mistake to recreate the old UI on modern touch screen devices!

I hold Microsoft Word responsible for inhibiting the flow of language, for imposing arbitrary hazards upon the nature of text, for pushing technologies of editing and revision where they don't belong

Recreating old UI patterns on a modern touch screen device is not going to win you any usability awards... (although, who doesn't miss Clippy?)

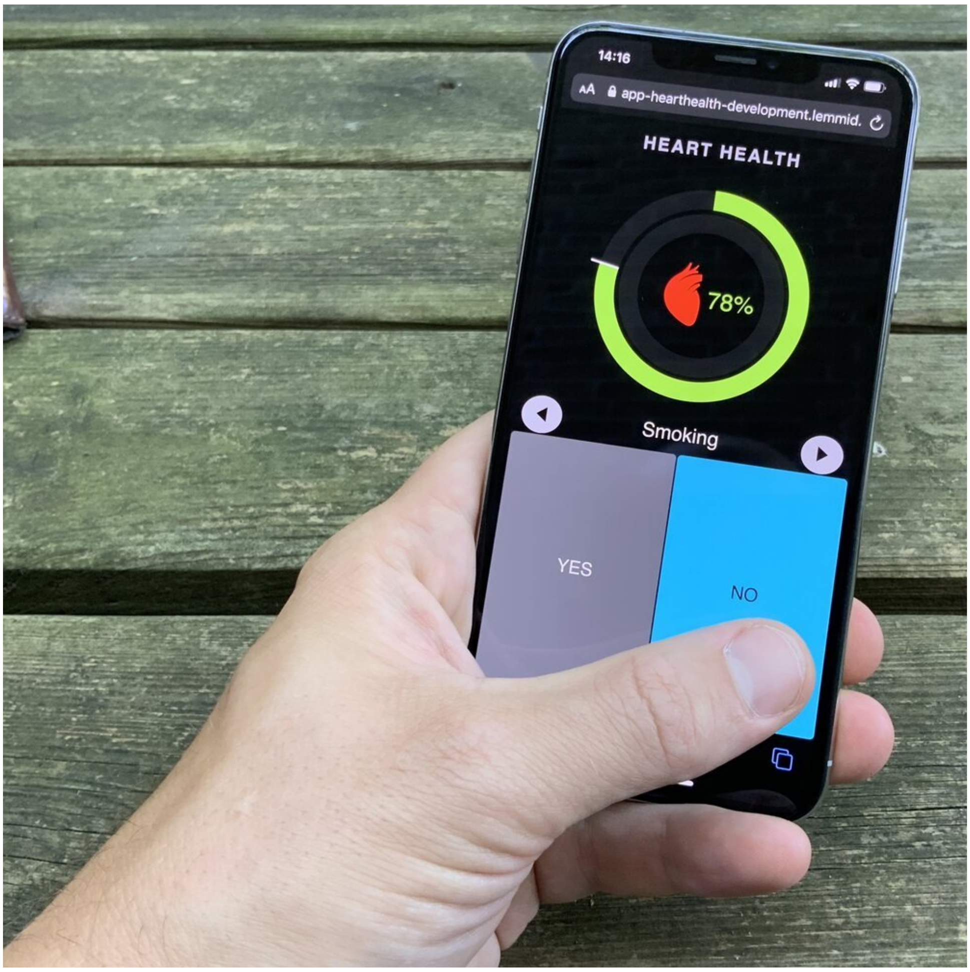
Natural interactions

Modern touch screen devices enable interactions to become more natural as they eliminate the need for input devices such as the mouse. The device *is* the input button: on screen elements can be easily manipulated for direct and immediate interactions, using touch and gestures.

Directly manipulating the on screen elements feels natural - a mouse in this use case would be redundant

Declutter and size elements adequately

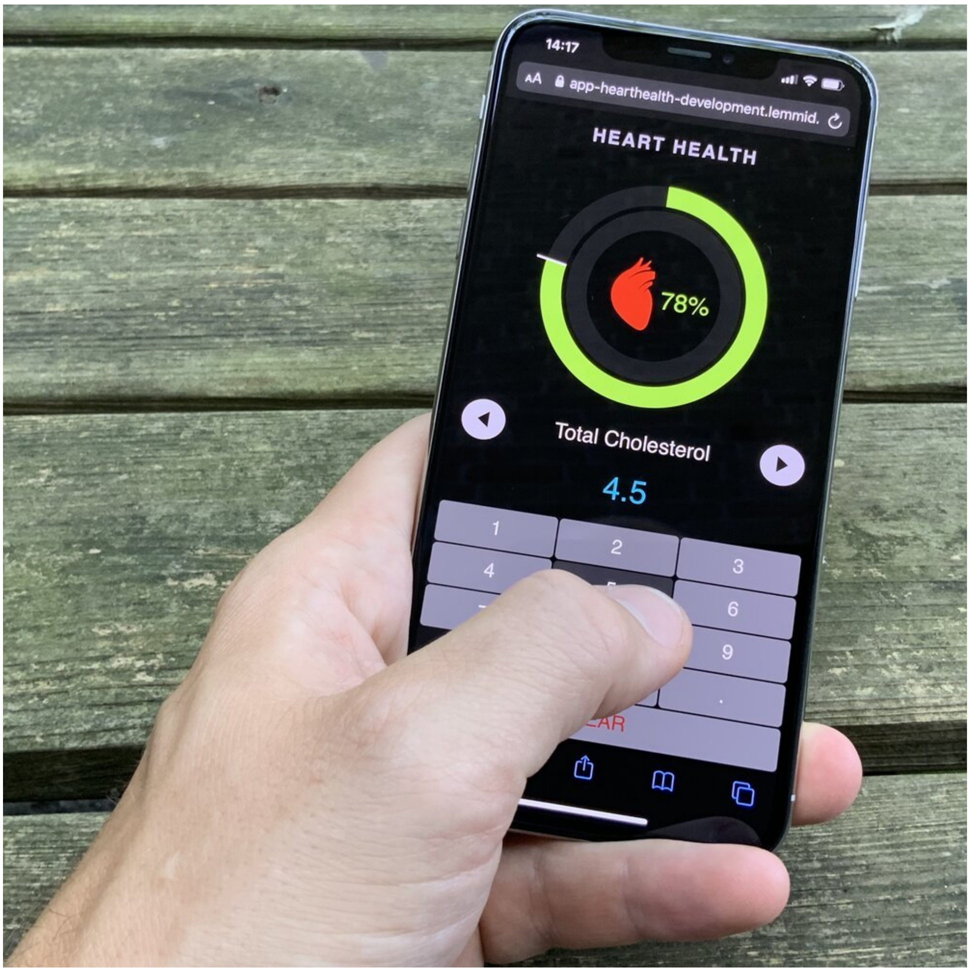
A finger is a lot larger than a pointy mouse cursor, this requires a different approach to sizing interface elements. Interactive elements should be big enough to be touched, with ample margins between them. Targets to close together increase the likelihood of error.



The big 'yes/no' buttons reduce the likelihood of error as they do not require mouse pointer pixel precision

Optimise textual inputs

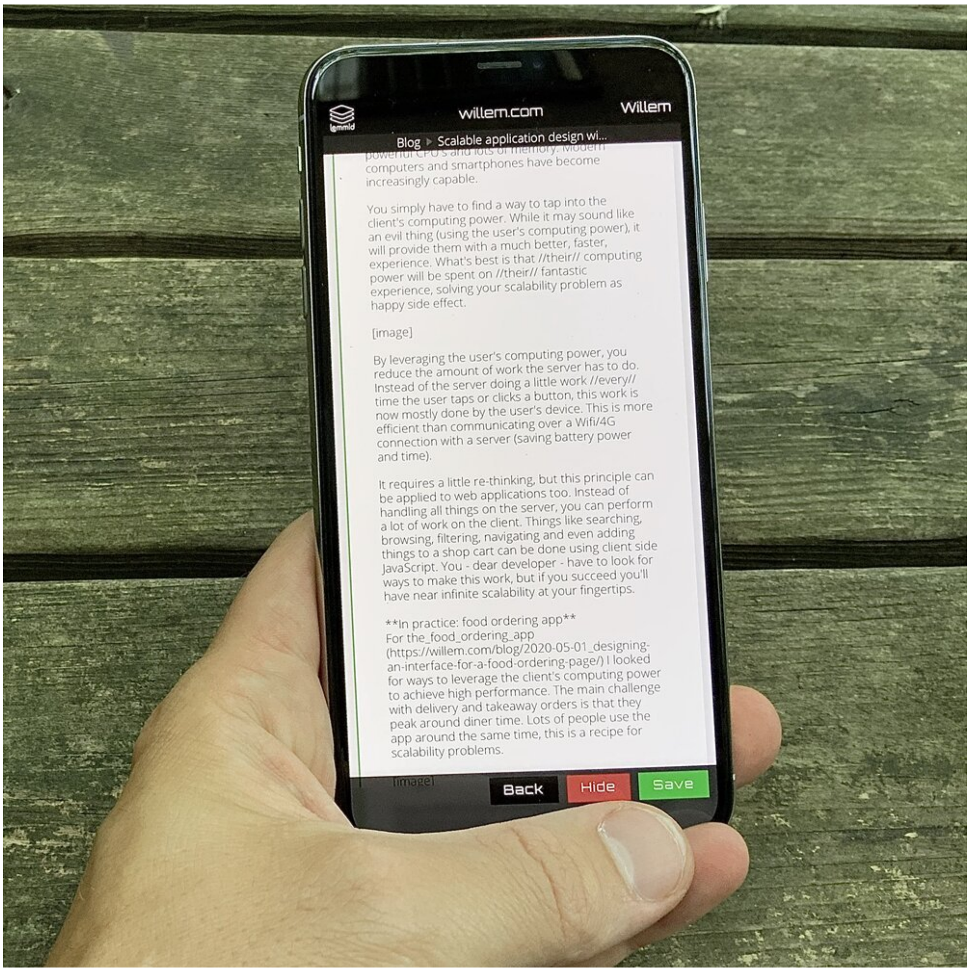
Depending on what you're building, chances are that you will need some kind of text or numeric input at some stage. The easy (and wrong) way to do this is to simply show a complete virtual keyboard. Instead of recreating a virtual recreation of the past (the physical keyboard) you should take the opportunity to rethink *what* you're asking the user. Depending on this *input context* you should then show a minimal keyboard that only offers those keys that are relevant.



Contextual keyboards provide a better way to enable textual input. It makes no sense to show a full QWERTY-keyboard if you only ask for a numeric value.

Rethinking positions

Where traditional PC applications often have their toolbars on top of the windows, replicating this on a touch screen device would cause the hands to obstruct the remainder of the screen. In addition to keeping the contents of the screen visible, you should also consider the ergonomics of handling the device. Frequently used buttons should preferably be placed near the natural position of the fingers.



powerful CPU's and lots of memory. Modern computers and smartphones have become increasingly capable.

You simply have to find a way to tap into the client's computing power. While it may sound like an evil thing (using the user's computing power), it will provide them with a much better, faster, experience. What's best is that //their// computing power will be spent on //their// fantastic experience, solving your scalability problem as happy side effect.

[image]

By leveraging the user's computing power, you reduce the amount of work the server has to do. Instead of the server doing a little work //every// time the user taps or clicks a button, this work is now mostly done by the user's device. This is more efficient than communicating over a Wifi/4G connection with a server (saving battery power and time).

It requires a little re-thinking, but this principle can be applied to web applications too. Instead of handling all things on the server, you can perform a lot of work on the client. Things like searching, browsing, filtering, navigating and even adding things to a shop cart can be done using client side JavaScript. You - dear developer - have to look for ways to make this work, but if you succeed you'll have near infinite scalability at your fingertips.

In practice: food ordering app
For the_food_ordering_app (https://willem.com/blog/2020-05-01_designing-an-interface-for-a-food-ordering-page/) I looked for ways to leverage the client's computing power to achieve high performance. The main challenge with delivery and takeaway orders is that they peak around diner time. Lots of people use the app around the same time, this is a recipe for scalability problems.

[image]

The Back / Hide / Save buttons are near the thumb: they are within range and reaching for them does not cover the contents of the screen

Forget hovering and tooltips

Forget interface elements, such as navigation structures, that rely on hovering. Most touch screens do not register fingers floating above the glass. Neither can you rely on tool tips to explain functionality of your interface. A touch screen UI should be clear to the user at first glance.

Navigation structures that rely on hovering are NOT going to work on touch screens

Be creative

Designing for touch interaction is something you should - above all - approach as something creative. Instead of recreating *the old*, you should be thinking of all the possibilities *the new* offers.

Snake '97 - It's fun, but it's UI design is arguably bad

When I created the Snake 97 game for touch screen devices I recreated the old input mechanism with painstaking attention to detail. While I think this is fun, it is arguably bad UI design as it totally ignores all the possibilities offered by the new technology. It's no surprise then that games like Angry Birds are so successful as they take *full* advantage of the natural interaction made possible by touch.

Catapulting birds is fun because of the natural interaction that is key to the Angry Birds game

Conclusion

One influential professor once asked me: ”But is it *fun*, Willem?”. When you are designing for touch, you should really consider this question. If you answer it with *”YES!”* then you should consider yourself on the right track.

You should be reimagining interaction instead of recreating old things. Sure, not everything old is bad; but it would be a mistake to ignore the potential good that modern technology enables!