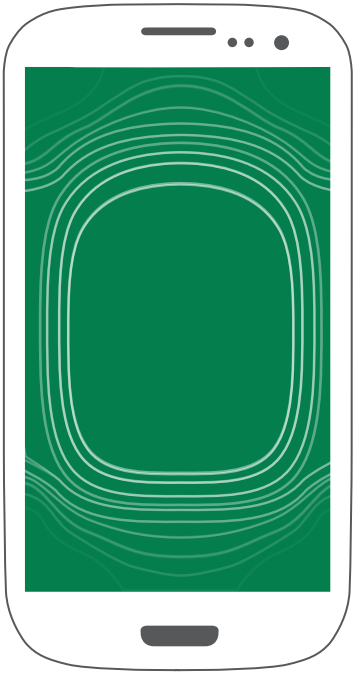
Touch Design for Mobile Interfaces

by Steven Hoober



Touch Design for Mobile Interfaces

by
Steven Hoober

Published 2021 by Smashing Media AG, Freiburg, Germany.

ISBN: 978-3-945749-97-5

Copyediting: Owen Gregory
Cover illustration: Espen Brunborg
Interior illustrations: Steven Hoober
Book design and indexing: Ari Stiles
Ebook production: Cosima Mielke
Typefaces: Elena by Nicole Dotin and Mija by Miguel Hernández.

Touch Design for Mobile Interfaces was written by Steven Hoober and edited by Alma Hoffman.

This book is printed with material from FSC® certified forests, recycled material and other controlled sources.

Please send errors to: errata@smashingmagazine.com

Contents

Foreword

by Brian Mills

What do you do right before going to bed? Or perhaps shortly after waking up? Many check email, news, weather, and/or social media… all through the omnipresent convenience of the mini-computer in our pocket, bag, or bedside.

From the moment smartphones hit the scene, it was game changing. Initially known as a personal digital assistant (PDA) or handheld PC (personal computer), the very first smartphone – the Simon Personal Communicator (SPC) created by IBM in 1992 and released for consumer purchase in 1994 – was large and cumbersome. Still, it embodied many of the hallmarks that quickly became defining features for every smartphone that followed. From a touchscreen to the ability to send/receive emails (and faxes!), the SPC also featured a calendar application, address book, native appointment scheduler, as well as standard and predictive stylus input screen keyboards.

When Apple later unveiled its first iPhone in 2007, the device featured a minimum of 4 GB of storage, a 3.5 inch screen, and deviated from prior mobile designs in eschewing most physical hardware buttons and championing a screen-based touch interface over the then-standard inde-

pendent stylus accessories. By the time Android became the most popular operating system for mobile devices in 2012, the market had exponentially diversified, in terms of not only software but both hardware as well as enhanced options for personal customization.

On average, modern smartphones now have more computing power than all of NASA did when it first started sending astronauts to the moon. Today's smartphones typically feature at least 4 GB (roughly 34,359,738,368 bits) of RAM, more than one million times the functional memory the Apollo 11 computer itself possessed. Beyond that, while the Apollo 11 computer held a processor that ran at 0.043 MHz, the latest iPhone processor is estimated to run at roughly 2,490 MHz; more than 100,000 times the overall processing power of the computer that landed humans on the moon more than 50 years ago.

As of 2021, more than half of all web traffic – almost 5.2 billion people – now takes place on mobile devices with an additional 25% increase in mobile traffic further expected by 2025, largely via both increased video consumption and streaming on mobile. While desktop usage is still predominant during the day due largely to most workplaces, smartphone and tablet traffic reigns supreme in the evening. More than just being extremely popular, touch interfaces have also become the standard for modern human-com-

puter interactions. Year-over-year, more and more "two-in-one" laptops are launched with built-in touchscreens. New foldable and/or rollable touchscreens are here (or poised to be released soon).

So, what does the future hold in terms of addressing human needs and behaviors via successful interaction design? Well, you're in for a treat: throughout this book, Steven assembles and proposes the basis for a more unified theory of information design, one with minimal caveats and exceptions while rooted in proven scientific processes and codifying what is presently known as technology continues to move beyond it.

As technology marches forward, it will continue to be tempered by human needs and behaviors, especially as the dominance of mobile and touch interfaces only continues to increase. Enjoy!

—Brian Mills

INTRODUCTION

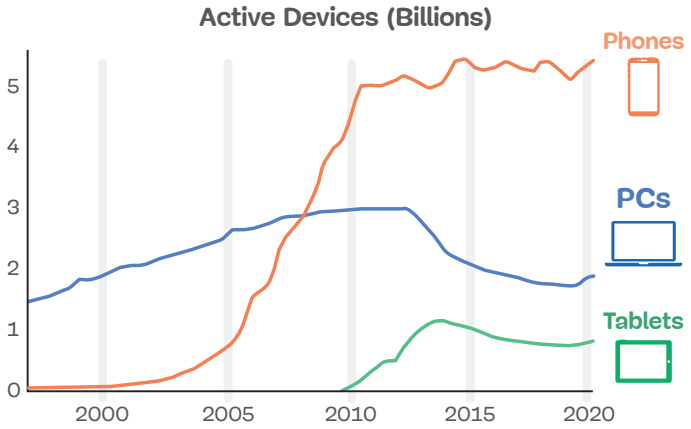
Everything Is Now Mobile

Mobile devices, especially phones, have rapidly become our main communication and connected information devices.[1] More importantly, the "traditional computer" with mouse and keyboard may have been an anomaly, a temporary circumstance that will only hang around for some types of work, but not be something we expect people to have access to every day anymore.

In the late 2000s when I started speaking and writing about mobile design, I led every talk with some charts about market shares, installed base, and use rates. I helped organize one of the first mobile-focused conferences where we all knew it was the next big thing, but when I worked or spoke elsewhere I still found myself having to explain how mobile was already a huge market and a massive opportunity.

Today it may seem that mobile devices are no longer the next big thing. There's no longer talk of the huge growth in mobile device penetration – it's already happened, as you can see in the chart below.

1. https://smashed.by/mobileuse



Approximate installed base of mobile handhelds, PCs, and tablets from 1997 through 2020.

The steep demand for mobile devices, and their usage rates, leveled off because everyone on the planet already seems to have one or more mobile phones.[2] [3] [4]

The mobile market is not just huge, it is the default for connectivity and communication. However, don't think of mobile users as different from desktop users, or as something new or unique. The sharp growth of mobile started almost 20 years ago as I write this. Mobile adoption is so widespread now that people across all generations have internet access on their smartphone or tablet.

2. https://smashed.by/movrmobileuse
3. https://smashed.by/usmobileuse
4. https://smashed.by/pc2019

In contrast, consider the personal computer (PC) line. The installed base – the number of computers actually in use in the world – of desktop and laptop computers has been dropping for years. Currently, in the US more people use mobile devices than computers for everyday tasks.[5] [6]

In this book we will discuss a body of research on how people really hold, look at, touch, and generally use their mobile devices, as well as the foundations of touch and the technology of touchscreens. In addition, I will provide tips and tricks for better design for mobile devices, based on this data, and bust some assumptions about older technologies, such as trying to apply desktop and mouse design paradigms to mobile design.

The first chapter, "**Defining Mobile Devices,**" describes different devices in the context of their unique attributes of portability, connectivity, and awareness. We'll come to understand how important it is to understand mobile technology and use patterns, and become aware that mobile paradigms are influencing more traditional computing platforms.

5. https://smashed.by/internetaccess
6. https://smashed.by/xmasdata

We disregard history at our peril; yet the tech industry seems to do this willfully. Chapter 2, "**The History and Technology of Touch**," reviews the evolution of direct screen interaction dating back to the 1950s, and the development of commercialized touch since the 1980s. Today's devices can be better understood by learning how touch technology advanced before achieving ubiquity through "Capacitive Touch," the subject of chapter 3, which addresses how capacitive touch intersects with human behavior and impacts our design work.

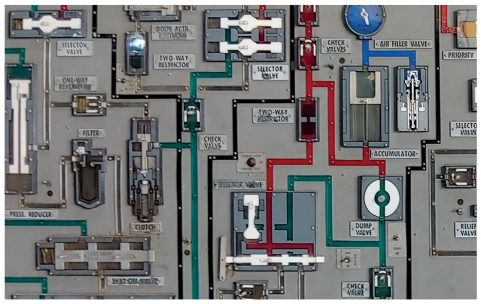
Machine Era Lessons: Air Logic

When consumer products like cars first needed complex controls systems like central locking and power windows, electronics were in their infancy. Electro-mechanical relays were large, hot, loud, expensive and unreliable. This was solved instead mostly by the use of "air logic," also known as fluidics, along the same lines as the term *electronics*. (smashed.by/fluidicapps)

Compressed air, (or sometimes vacuum or other fluid) systems use miniature three- and four-way valves to fulfill the needs of the logic components *and*, *or*, *not*, *yes*, and *flip-flop*, as well as timers and delay mechanisms. Users turn dials or push buttons

In chapter 4, we'll learn how various "**Standards, Assumptions, and Problems**" can be problematic for designers of today's mobile touchscreens. The specifications, norms, and principles of earlier times are sometimes not to be trusted as they are too often based on technological assumptions that no longer apply.

Chapter 5, "**Finding Out How People Hold and Touch**," covers the observational research I conducted to discover how people actually manipulate their mobile phones and



Portion of a training panel depicting the bleed air, pressurized air, and hydraulic fluidic logic and operations for the F-100D fighter.

that feed data into the system via air instead of electricity. The control system actuates power controls, which cause air or electrically driven accessory controls to function.

(continued overleaf)

tablets in everyday use. Through debunking some widely accepted but incorrect assumptions, we'll learn most of all to change the way we think about designing for touch, because while there are many ways to hold a device, everyone uses them all, constantly shifting from one to another.

To understand touchscreen performance, I undertook several studies and found that popular notions of touch accuracy and preferred touch regions are wrong. I discuss this in chapter 6, "**Touch Accuracy and the Center-Out Preference**," and show that people favor the middle of the screen for both reading and touching. Every mobile device user has experienced "**How Fingers Get In the Way**," and in chap-

Air Logic (continued)

Some even used these to control engine accessories, like air conditioning, fuel timing, and emissions controls. (smashed.by/projectwater)

These days, air logic has only specialized industrial and military uses. From the 1970s, the cheap and reliable integrated circuit began replacing air logic, and cars were really the vanguard of the all-electronic world we live in.

When I encounter an air logic system, I often imagine a long-retired engineer with a shelf of industry awards and

ter 7 I explain how our designs can take into account the ways people adapt their touch to perform different actions like tapping and scrolling, and to overcome problems of visibility and interactivity.

Some missed touches can't be avoided, however. Chapter 8 covers the issues around "**Imprecision and Probability**" and shows how we need to design systems, interactions, and processes that prevent mistakes – especially avoiding catastrophe when mistaken taps are made.

We mustn't forget that mobile devices are different from desktop computers not only because of touch or their size,

books; the creator of air logic, now forgotten owing to the march of technology.

What I say in this book is as accurate as it can be at the time of writing, and while much of it is about human behavior, it is presented in the context of how touchscreen mobile phones, tablets, and to some degree computers, work in 2021. We must keep ourselves educated, keep up to date with trends and technology, and never be so in love with a particular solution, or way of working that we lose sight of the changing world.

but because they are used in all kinds of locations and people handle them constantly. People and their environments can be confusing, confounding, and unpredictable. Chapter 9, "**Phones Are Not Flat**," describes ways we can consider likely problems when planning the design of our apps and websites.

We start to move into tactics – with a little less theory, data, research, and background info – in "**People Only Touch What They See**" (chapter 10). I will cover best practices in how the UIs of interactive elements are designed to attract the eye, afford action, be readable, and inspire confidence that they can be safely tapped.

In chapter 11, "**1, 2, 3: Designing By Zones**," I'll introduce the concept of information design, describe how human vision is not what it appears to be, and then turn all that we've learned so far into a simple formula we can all use to create well-organized, usable templates for touchscreen design.

Shifting from template theory to template creation practice, in Chapter 12, "**Progressive Disclosure**," I review the pros and cons of some of the most important page design elements, such as menus, lists, floating bars, and tabs, to see how they can integrate with the concept of information design for center-out touchscreen products.

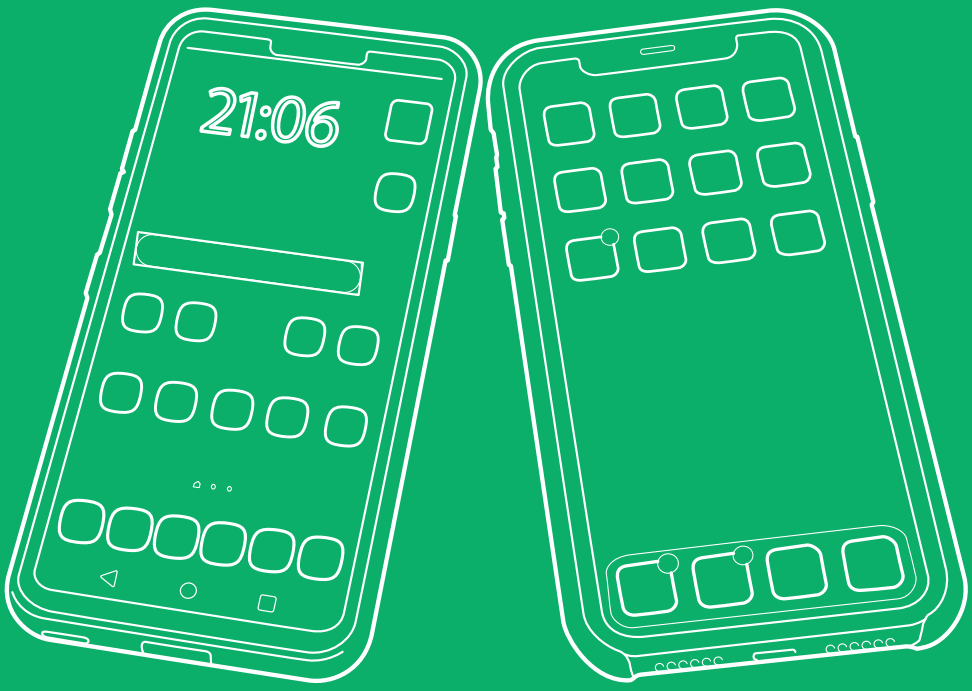
In the final chapter, "**Practical Mobile Touchscreen Design**," we'll finish off by skimming lightly over more or less the entire process of designing digital touchscreen products, from teams to strategy and onward. From these resources we can start building a reference library of how to pursue each aspect of the design process.

This Is All True, and It All Works

This book is a comprehensive overview of my work on the topic of touchscreen use and designing for touchscreens, but it isn't the first time I have shared the information or the design guidelines.

Many others have used these insights in their work and have found the same results, significantly improving their product design through deeper understanding of human behavior and physiology. I have lived and worked through most of the rise of the mobile – and especially the smartphone – but in writing this book, I took my recollections and my points of view, and undertook research to grasp the wider story and determine if what I saw was true.

But first, let's start at a very high level, with some definitions of the sorts of devices we're talking about, and the history of touch.



CHAPTER ONE

Defining Mobile Devices

CHAPTER ONE

Defining Mobile Devices

This book is about designing for mobile devices based on their unique attributes of portability, connectivity, and awareness. But before we can talk about how people interact, much less how to take advantage of that knowledge and design, it's important to understand a bit about the history, the technology, and what today counts as a mobile touchscreen device at all.

The personal computer (PC) is still assumed to only be used at a desk-like workstation, in discrete sessions of work with the user focused entirely on the computer.

Mobile devices have always been different from "computers" in that they are:

- always on: there's no need to turn them on to start or end work
- always with us: not just close at hand, but also personal devices
- aware: by being connected, and full of sensors

Mobile phones are rapidly becoming touchscreens and touchscreen phones are increasingly all-touch, with the

largest possible display area and fewer and fewer hardware buttons. Today, about half of mobile devices are smartphones, and some of the remaining feature phones are also touchscreen. (*Feature phones* are mobile phones with extra features. Usually today this means internet connectivity, cameras, GPS, and so on, but they are distinct from *smartphones*.) As shown in the trend chart in the introduction, with the majority of internet access via mobiles, in just another decade almost everyone will use touch as their primary interaction method, worldwide.

To many designers and developers the process of designing for mobile assumes that touch is natural, so we don't need to pay any particular attention to the design of touch systems. This is not true. As children we all had to learn – for years – how to touch, feel, and manipulate real-world objects. Touchscreens and our standard paradigms of interaction are not the same as the real physical world; touchscreen behaviors are as learned as the use of a mouse or a doorknob.

Touch is also not a direct analogue of "traditional" pointing devices like a mouse or trackpad, and there is not one type of touchscreen. Changes in the technology of touch over time mean that many assumptions and standards of how to design for touch from just a few decades ago are no longer relevant – and may be actively misleading or dangerous.

Let's start by defining what a mobile device is and examining the scope of how understanding mobile technology and use patterns is important when designing for the huge number of mobile devices in the world, as well as because the world is changing.

Smartphone

One of the more common assumptions is that mobile means a touchscreen smartphone. Something like one of these in the picture below is certainly in your pocket, or lying next to you. Some of you have more than one of them.

Android and iOS smartphones.

When you encounter a smartphone anywhere in the world, it is probably an Android phone. It barely matters which manufacturer, in the same way you probably barely care who made the Windows PC on your desk. The hardware matters, but the underlying OS is the same, and pretty much all apps will run on any device of the same age. Android holds a bit over half the US market, and closer to 75% of the installed base worldwide.[1]

Building a Device Lab

I am very often asked what my favorite phone is. I have always told everyone that, professionally, I have no opinion. I make a point of having a lot of devices and switch between them regularly – not because I am indecisive or must always have the newest and best thing, but so I can stay familiar with the variations between devices and operating systems. I need to understand how real people use digital products and services.

If you design for mobile devices, you really need to create or get access to a device lab, a small collection of functional phones and tablets that address the range of the most likely devices your users employ. My choices in a lab should not be your choices.

1. https://smashed.by/movrmobileuse

This simple Android/iOS market distinction was not always true and took a long time to settle into its current shape. It was a full six years after the iPhone launched before the then-ubiquitous Nokia Symbian S60 OS was overtaken by iOS. And perfectly good BlackBerry and Windows devices hung on with good market share in some regions for years longer. Why? Because the smartphone didn't burst onto the scene fully formed, but it transitioned and offered multple

My mobile device lab, right next to my computer.

In the past, when there were four or five major OSes, I maintained a very large library of devices and switched which one was my personal phone at least twice a year. Right now as I write this, I mostly carry Android, but pick up the iPhone a lot.

(continued overleaf)

solutions to people's information and communications needs. The first smartphone was more often called a *PDA phone*, because at that time everyone knew what a PDA was.

The personal digital assistant was a touch device – though most were used with a stylus or pen – with an interface much like we have today on smartphones. But PDAs were not connected and had to be docked to a computer to sync. When the first mobile phones became smart and acquired the PDA pieces, they quickly became marketed as smartphones to differentiate them from other device types.

Device Lab (continued)

If you're building products for foreign or global use, don't just go to the phone store and pick up what is popular wherever you live. Find the most common phones for your target market or audience, and try to get one of those. Yes, many foreign market devices are available in other countries, and work well.

Remember that mobile networks are not the same as Wi-Fi, so carry a spare SIM and switch that out occasionally, especially during tests of whether your app or website operates properly. Likewise, I have both a Mac and a PC on my desk. My current tablet PC is employer-issued, but when I work for clients who don't provide one, I buy a

The Kyocera 6035, the second PDA phone sold in the US.

PC of my own. And I have a Chromebook. Yes, mostly the toddler watches TV on it, but this is one way that I keep a library on a budget and experience firsthand how they really work. I use these devices in my day-to-day life.

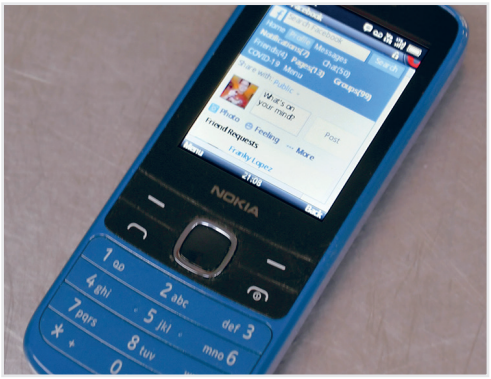
Yes, this will cost something, so with any luck you can get your employer to fund it, or maybe find an open device lab, a shared pool that you can use free or for a nominal fee. I have also known designers who share with one another to create ad hoc labs, so ask around the community and see what resources are already available.

More at smashed.by/devicelab

Feature Phone

As shown in the introduction, today almost everyone in the world has a mobile device, but fully half of them are still not smartphones. Early cellular mobile phones only made phone calls. Fairly rapidly, text messaging (SMS: short message service) was added, as it was baked into the network itself, originally as a way to send messages internally or for testing.

By 2002, mobile phones had access to the internet with what became perfectly good web browsers, had cameras, and within a few years would get additional features such as GPS receivers, faster internet, Bluetooth, and Wi-Fi.



A typical feature phone, displaying the Facebook feed.

These were called feature phones to differentiate them from plain old phones (they had more *features* – get it?) or what would later be derisively called *dumbphones*. While some feature phones are still clamshell or *flip phones*, the candy bar or slider (as shown opposite) are at least as common.

Feature phones all use proprietary and little known operating systems developed by the makers of the phones, as opposed to prevalent ones like Android or iOS. Though usually unable to be meaningfully upgraded, they can install apps. In many ways, the feature phone app ecosystem is simpler than the smartphone ecosystem we have today as almost every app works on almost every phone. Many phones come preinstalled with common apps such as social networking, email, and maps, or with shortcuts to the browser.

Users of feature phones are not the left behind, and can be connected through their mobile phones like any smartphone user. Most project teams, designers, and even governments are quite dismissive of this half of the world, but we shouldn't be. These devices tie the world together, and have formed the foundation of many very large-scale and profitable projects. Your product could probably benefit from working at least in some way on feature phones.

Tablet

iPads are considered an entirely distinct market from phones, but I consider them clearly mobile as well, owing to the way they are used. People carry them around, work with them while standing, unlike how they use laptops. And the way they interact is a direct extension of mobiles.

Notice I just said iPads, because – as every expert says – iOS won that market and there are almost no Android tablets.[2] At least that's what I hear from all tech writers discussing how awesome the new iPad keyboard is, or when I go try to buy one at the local discount computer bodega.[3]

An Android tablet, set up for toddler streaming and gaming.

2. https://smashed.by/tabletwar
3. https://smashed.by/ipad

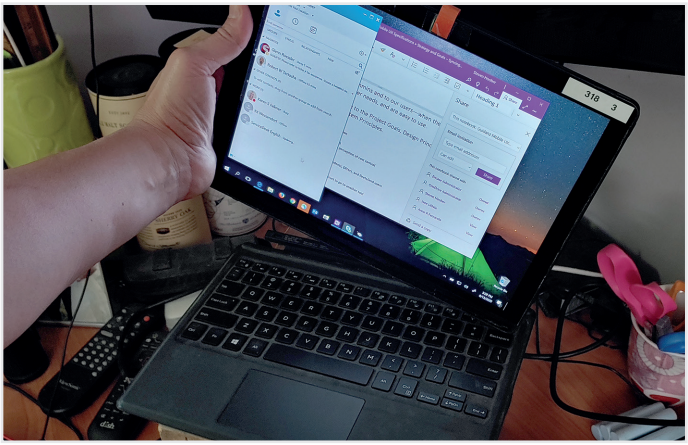
Very often, however, what our gut instinct tells us and what "everyone knows" is flat wrong. Android is installed on a huge number of tablets, and there's a whole other class of devices called Chromebooks. Most industry-tracking sources classify them as computers. But the majority are in tablet form factors, with maybe a dockable keyboard.

Chrome OS is also not really a different OS, much less a computer one, but simply a very minor offshoot of Android. When counted as computers, as they commonly are, they make up 10–20% of all PCs sold, and this figure is rising rapidly. These are big numbers, and if considered part of the tablet market, since they are touch-first, mostly Android machines, Google would own over 70% of that.[4]

Personal Computer

With the Chrome numbers included, "laptops" are no longer any such thing. But even for Windows, many have touchscreens and undock or fold up to turn into tablets or other types of touch-first devices. The picture below shows my corporate-issue Windows PC being re-docked to its magnetic keyboard.

4. https://smashed.by/chromebook



Removing the tablet portion of a detachable Windows PC.

The classic clamshell laptop or notebook PC is losing ground to the convertible (fold around to turn into a flat device) and detachable form factors. Over 40% of laptops sold in 2020 were of one of these 2-in-1 designs.[5]

As early as 2013, 10% of laptops had a touchscreen.[6] Today (in 2021) it is hard to find an exact number, but it appears over half of all laptops sold include touchscreens. And their use is also very tablet-like. People carry them and use them in the hand, tapping the screen to interact and only placing them on a flat surface to type when very long forms have to be filled out.

5. https://smashed.by/pcdforecast
6. https://smashed.by/touchlaptops

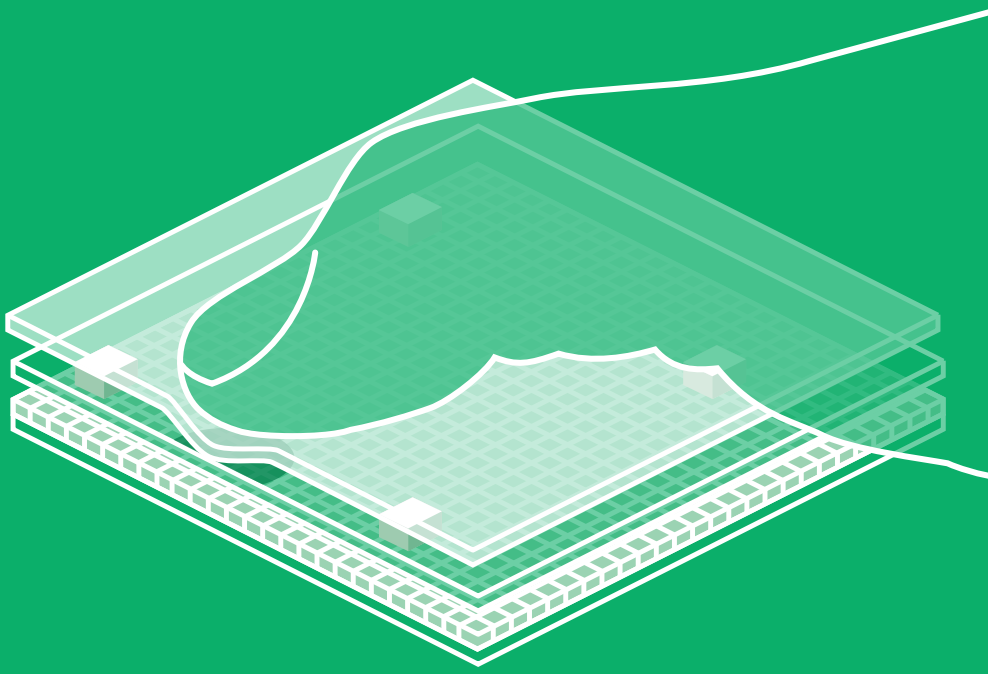
A plumber at my house, interacting with the touchscreen on his laptop.

Even while sitting at a desk, people will tab their way through a spreadsheet, mouse into the ribbon to do some formula selection, then reach over and tap the screen to dismiss a dialog, or switch to another app.

In addition, a number of desktop all-in-one computers also include touchscreens, and are used much the same way as any convertible PC when on a desk. Touch and computers generally designed around mobile principles are everywhere. How did this happen?

It's helpful to have a clear understanding of what a mobile device is. We can all define them better for our projects, and be more aware of how far broadly mobile methods have reached into other more traditional computing platforms.

Knowing what we mean by touch is equally important, and essential to that is an appreciation of the many methods and history of touchscreens and other pointing devices in computing. In the next chapter we'll take a tour through what came before today's ubiquitous capacitive touch.



CHAPTER TWO

The History and Technology of Touch

CHAPTER 2

The History and Technology of Touch

Now that we have some working definitions of device classes, it's time to dive into what we mean when we refer to a touch or touchscreen device. It is a lot more complex than you might think, with many device types and methods of use, and a history going back some 70 years.

Touch Is Older Than the Mouse

SAGE – Semi-Automatic Ground Environment – was a giant network of radars, radios, and command centers all coordinated by a computer. The US Air Force used it to sense and coordinate the response to an expected enemy attack by bombers and, later, missiles. It was, in fact, the first real full-time working computer, turned on and left on for decades. Developing it created the entire field of software engineering, analysis, and project management.

The semi-automatic part meant the computer did many tasks previously done by hand, like plotting positions of radar-tracked items. When an operator selected an item as

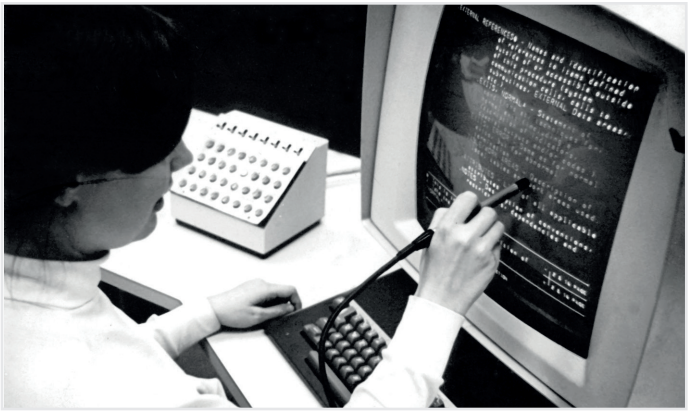
shown here, the computer provided more information, and it could perform contextual actions, such as allow a fighter plane to be vectored to that target without manually copying or reading back the data.

A SAGE controller selecting items on screen in 1958. (Used by kind permission of The MITRE Corporation.)

How did the operator select the target? By direct screen manipulation. The trackball was used as early as the 1940s in secret military or specialized places, and was developed independently several times over decades. The mouse would not be seen in public until Douglas Engelbart's "Mother of All Demos" in 1968.[1]

1. https://smashed.by/mousehistory

Instead, the SAGE operators used light pens or, for the first few years, light guns. Over time, this evolved and by the late 1960s light pens were usable on reasonably modern-looking systems to do tasks we're familiar with on touchscreens today. The Hypertext Editing System (below) was not just a prototype, but shipped and was used to do work like editing Apollo program documents in the 1960s. Light pens were so expected to be the pointing device, integral to the display, that there are still reserved system calls on the PC for their control, included as part of the display monitor controls at interrupt 10.[2] Mouse controls were added much later, as their own command set at interrupt 33. [3]



Using a pen to select text with the Hypertext Editing System on a computer terminal located at Brown University in 1969.

2. https://smashed.by/int10
3. https://smashed.by/int33

In 1983, the HP-150 computer workstation was released by Hewlett-Packard with a touchscreen and the same ability to be used as a text editing system with basic pointing. Touchscreens began to make their way into consumer products, like automotive infotainment systems starting with the 1986 Buick Riviera.

The mouse remained obscure and expensive until it was shipped as the default pointing device with the first Macintosh in 1984. It wasn't until 1992, when the release of Windows 3.1 resulted in the mass adoption of the PC, that computer users really began to switch from command line interfaces.

The WIMP (windows, icons, menus, pointer) interface has been mainstream barely longer than the web. Throughout that time, direct manipulation systems of pen and touch have been doing good work, and have improved and become more and more available.

All these devices, over decades, worked in a variety of ways. While now we're only really concerned with how capacitive touch works (see chapter 3), the best way to understand how and why that works is to know a little about the earlier technologies, and how they evolved into the modern smartphone.

The Technology of Touch

There have been numerous technologies that support touch input.[4] Here I am going to outline all of those, because there are issues that arise with assumptions about technology in old designs, standards, and publications. I'll talk about those problems in chapter 4, but first, we need to understand the baseline technology, and how modern capacitive touch systems are very different from older technologies.[5]

LIGHT GUNS

The earliest systems with light pens or light guns, as described above, worked through a very unusual method that is broadly inapplicable today. I will discuss this briefly so you know why, and also so you understand how the first direct screen manipulation tool you may have used is not technology we use today.

Nintendo's Duck Hunt game used a light gun, with exactly the same technology as that first employed in 1950s aircraft and missile control systems. If this makes you want to plug in your old Nintendo, brace yourself: it won't work. The scanning method is specific to the way cathode ray tube (CRT) TVs work (the big high-voltage glass tube technology) and won't operate properly on any flat panels like the LCD TVs we have today.

4. https://smashed.by/touchtypes
5. https://smashed.by/touchtechnology

Nintendo's Duck Hunt game controller during play.

Keyscan

Another underlying technology that can aid our understanding is how keyboards, keypads, and sensing grids (such as touchscreens) work.

Most people who think at all about how keyboards and keypads work presume that each key is also a button; that is, a key is wired to a momentary contact switch and directly sensed by the computer the moment it is pressed. But that would be extremely cumbersome and expensive.

Instead, a matrix circuit is used, where all the keys are arranged in a grid. The state of each column of switches is scanned in turn to identify if any

The technology for pointing detection was essentially the reverse of what we might think about modern touchscreen interaction. When the trigger is pressed (or the pen makes contact with the screen) a signal is sent to the display; in turn, the display flashes a series of squares across the screen.

Ideally, these flash fast enough to escape the user's notice, but in practice they are just barely visible. Each square appears for just a moment and at a very specific time. The gun or pen has a camera tightly coupled to the display timing. When it sees one of these squares, the exact time it sees it

particular switch is closed, meaning that a specific key has been pressed: smashed.by/keyboardmatrix

Touchscreens – no matter the technology – work the same way. All of them have a grid of sensors and scan the grid inputs to sense touch. This matrix scan, or *keyscan*, is performed many times a second for the entire array. For keyboards this is easy, and fairly low scan rates perform adequately, as even very fast typists cannot strike more than 20 keys per second.

For touchscreen sensor grids, there are many more points to scan, and users do not just tap but gesture as well. Since mobile hardware is always

(continued overleaf)

allows the computer to tell which square it is looking at and register a selection, or hit.[6]

Non-CRT displays don't scan like this, so modern flat-panel displays simply cannot use the technology. In addition, the need for a camera and cable to keep the timing working makes this entirely unsuitable for general purpose selection today.

Keyscan (continued)

slower owing to size, heat, and power management, slow scan rates caused some of the problems of early touchscreens, even through the mid 2000s. Slow scanning on touchscreens leads to dropped points, which is especially visible when trying to use handwriting input, which was supposed to be the killer app of those early touch and pen devices.

In 2020, touchscreen sample rates are as high as 240 times per second, or multiple times faster than the screen refresh rate: smashed.by/touchsamplingrate

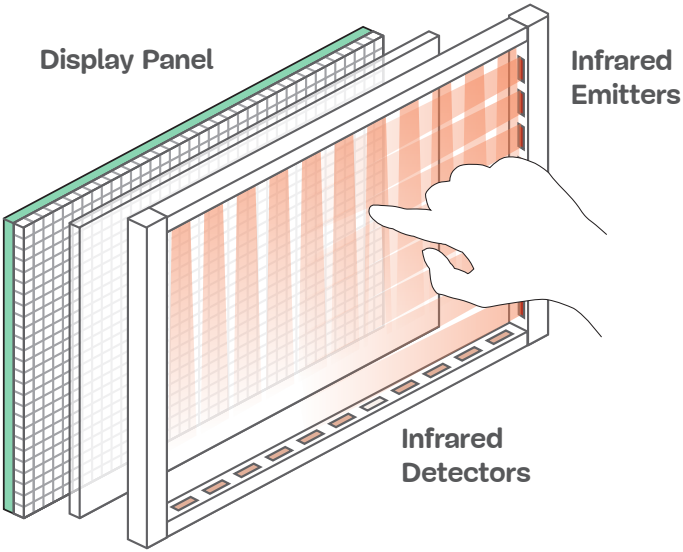
This allows the touchscreen response to be precalculated, avoiding any perceived lag or jumpiness.

6. https://smashed.by/lightpen

INFRARED BEAM

The next oldest direct input technology, the infrared beam, didn't have any special requirements for the pointer, so it could use absolutely anything, including fingers.

A grid of infrared (IR) beams and emitters is placed in a raised bezel around the edge of the flat display area. When one beam is interrupted, the computer knows a selection has been made.



A diagram showing how infrared beam sensors work.

The HP-150 touchscreen computer pictured earlier used this method. The first models revealed that the real world is full of dust, and they required regular and tedious cleaning of the detector and emitter ports. This issue was rapidly overcome, with conformal covers that empowered touchscreens on industrial applications and ATMs.

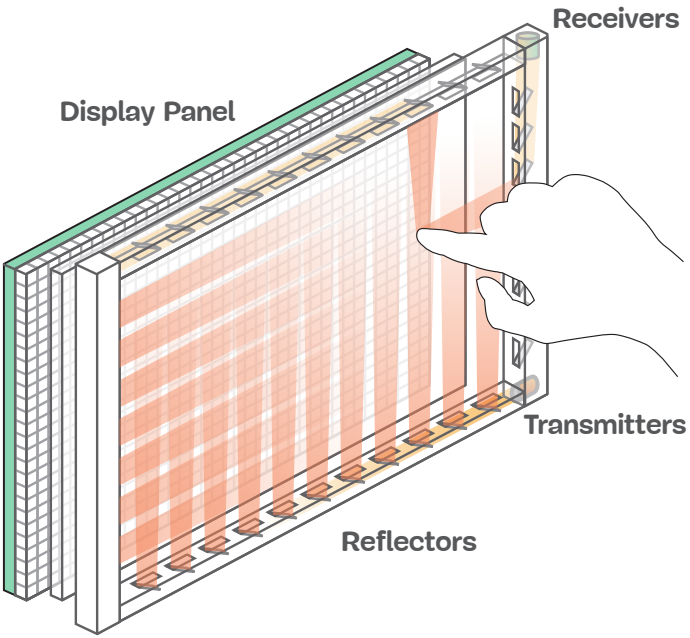
IR-beam sensors are bulky enough that they were never really used on any portable devices, but they continue to be used in other applications. They can be installed over entire walls, be made very rugged, work with any pointing devices, and don't even need to be over a screen, so they can be used for selection on printed or projected surfaces. They are capable of multitouch sensing, respond very quickly, and provide no resistance themselves to the moving finger, so can be used to paint and manipulate objects freely. The latest fighter jet the US Air Force promotes as being very high-technology, the F-35 strike fighter, relies almost exclusively on touchscreens instead of switches, but they are IR-beam touchscreens that work in all environments.[7]

SURFACE ACOUSTIC WAVE

An alternative to infrared light is sound. Like IR-beam screens, a raised bezel surrounds the display area, but most of it is taken up by reflectors; transmitters and receivers are

7. https://smashed.by/capacitivetouch

placed in the corners. The transmitters send out ultrasonic waves, configured to "stick" to the display surface, hence *surface acoustic wave* (SAW).



A diagrammatic layout of the operation of a SAW touch sensor.

When something soft, like a human finger, touches the screen, it blocks the waves and appears as a dead spot in the area, whose position is found by the receivers.[8]

8. https://smashed.by/acousticwave

These are similar in use to IR-beam systems, but even less used now. While they too can be made very rugged and possess high precision, so they can use contact size as a proxy for pressure sensing, they are easily confounded by water and oils – two things humans have on their hands all the time.

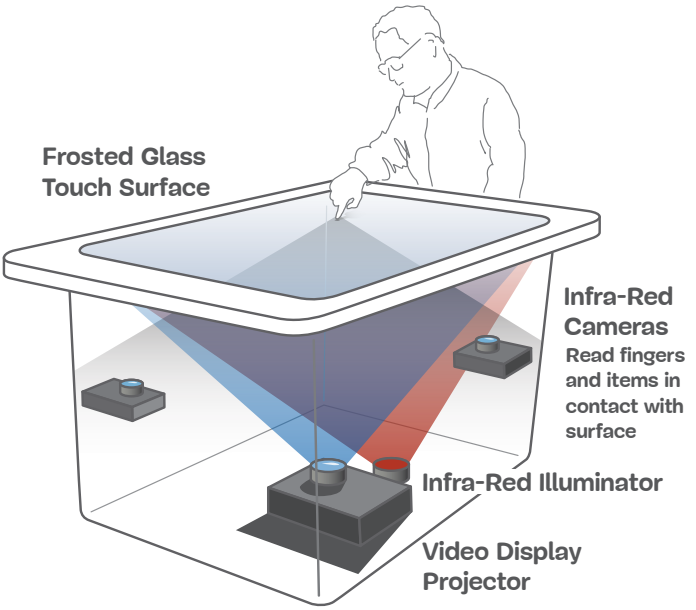
In the 2000s, there was a brief resurgence in a related acoustic technology using pens or pen-holders that emitted their own sound to offer portable pen-based capture devices. An office I worked at used one of these to good effect, but even those are no longer made, and other systems meet the needs instead.[9]

MACHINE VISION

Many of the earliest experiments in touch used optical systems of one sort or another. While these are usually in the IR band, they should not be confused with the IR-beam systems outlined above.

While some research projects have used several cameras pointing at the screen, and image recognition and multilateration to identify the position, these are complex and rarely seen in the wild. A simpler method is used on rear-projection devices such as the Microsoft Surface table (later renamed PixelSense, and discontinued around 2013).

9. https://smashed.by/mimeo



How a typical machine vision table works.

Detecting touch on rear-projection or transparent-display systems simply involves a camera and lighting behind the display screen. *Machine vision* uses digital signal processing to recognize shapes and use them to control processes.[10] As you may have seen when looking through frosted glass partitions or a fogged-up window, items even a few inches away can be vague or invisible, but items in contact with the glass are perfectly sharp. This, along with IR lights to illuminate the fingers in contact makes the image recognition simple and reliable.

10. https://smashed.by/machinevision

Machine vision systems can be very high-resolution, support multitouch, and have useful additional features, such as recognizing individual hands to differentiate multiple users, or recognizing other objects, allowing users to merge digital and physical objects together. However, these sorts of touch sensors are very thick and heavy, and require an empty space behind for the cameras to view the screen.

Digitizing Tablets

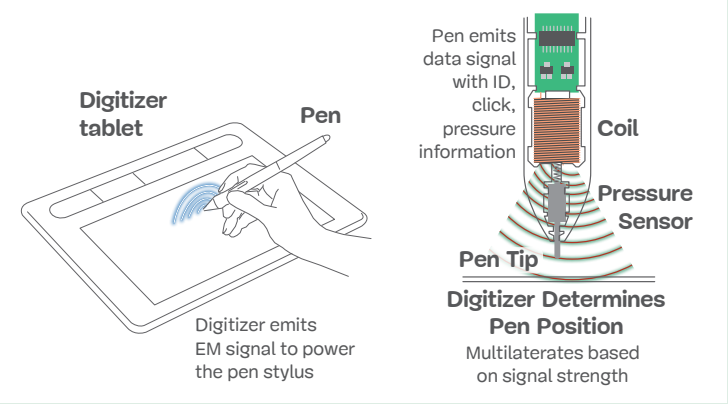
There is an entirely other way that stylus devices work, mostly used by device makers such as Wacom for its digitizing tablets and monitors, but also encountered in many touchscreen laptops and some mobile phones, such as the Samsung Galaxy Note line. They use *electromagnetic resonance*, which interacts with the stylus to determine not only position on the surface, but also distance above the surface, angle, and azimuth. Usually the stylus includes a moving tip with a pressure sensor to send pressure telemetry data as well: smashed.by/emrstylus

This is a very interesting technology, and one I have used every day for over 30 years. But it requires very specific styluses and does not work with fingers, which puts it beyond the

RESISTIVE TOUCH

You have certainly used a resistive touch panel.

While rapidly losing ground to capacitive touch, resistive was very common for a long time, and it is cheap and rugged enough to persist or dominate in certain areas,

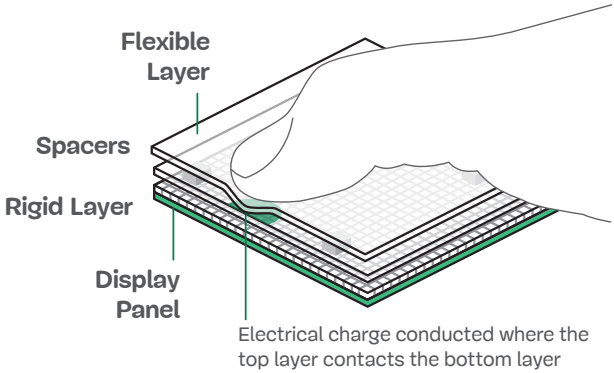


How a digitizing stylus interacts with the tablet.

scope of this book. Many of these devices are also touch, but these contain two technologies: a projected capacitive grid, and a resonance grid behind that. They interfere with each other, so some method to switch – whether manual or automatic – is always included.

such as point-of-sale terminals, airline seatback entertainment centers, and industrial controllers.

Resistive refers to physical or mechanical resistance, not electrical. The functionality also tells us why it has fallen by the wayside. The top layer of the display is a flexible plastic layer. When touched, it makes contact with a rigid layer below and closes an electrical path. The grid position can then be sensed.



A cross-section with exaggerated scale showing how resistive touch works.

Since about 2010, gesture-sensing and multitouch resistive panels have been available as well.[11]

Resistive touch was the standard for a long time, and found in tablet PCs, PDAs, and smartphones. It is reliable, and at

11. https://smashed.by/resistivepanels

the time when glass displays were much more fragile was very rugged – resistant to environmental changes, dropping, or impact damage – compared with capacitive touch.

The other key upside to resistive touch is stylus use. On today's capacitive touch devices a stylus or pen with a touchscreen is considered an admission of defeat, and except for a handful of devices with specialized sensor panels, such as the Samsung Galaxy Note series, pen support is poor.

But resistive devices just need touch, so any pen that doesn't scratch the screen will work, and almost all devices come with a small pen tucked away inside the device case.

The onboard pen also provided for something that capacitive still doesn't really do well: supporting input in adverse environments. Resistive doesn't care about dust, dirt, snow,

A resistive touch smartphone and passive stylus covered in snow, but still being successfully used to log data (c.2007).

or water on the screen, or what is being used to touch it. As long as the user can touch the screen accurately, whether with fingers, gloves, or stylus, the device works.

The downsides to resistive touch are largely due to the technology being mature and improvements to capacitive touch. The flexible plastic top layer means resistive touch devices cannot be as bright or clear, regardless of the display underneath. Compared to early capacitive touch, they were very sturdy: the plastic does not crack when dropped; they can be worn, scratched or cut if misused, or items like a ballpoint pen are employed as a stylus – a common problem in point-of-sale terminal applications.

They also vary a lot in ease of use. The more rugged devices inherently have stiffer and thicker top layers, which are harder to deflect to register a touch. These often require a pen or stylus to use properly, making them less suitable for touch applications, but which does permit higher-density design and much more flexibility in applications than the typical alternatives with push buttons only.

Sometimes there are serious mismatches in the design, and rugged display units with stiff touch are used in environments where no pen is offered, so they are simply hard to use, as well as suffering damage from ad hoc stylus use. I have encountered new installs in the late 2010s like this.

A portable point-of-sale terminal with a resistive screen, as currently used by an airline. The stylus is stowed at the top.

While resistive touch is now considered hopelessly out of date for consumer products like mobile phones and tablets, it is still selected for certain new applications. If you have to design for a system using resistive touch, most of the touch guidelines outlined in the rest of the book apply to these screens. Be sure to understand where the technology differs, and try the display panel you are required to use so you understand how much pressure is needed and if it is truly touch capable or requires a stylus.[12]

KINESTHETIC GESTURE

The successor to the Duck Hunt gun is really the concept of control by moving your body to control systems directly.

12. https://smashed.by/resistivetouchscreens

While simply waving at the computer to get it to do things has been a dream since the earliest days, it has been elusive, unreliable, and is still somewhat of a niche.

These technologies are often available on modern smartphones for fairly direct interactions. You can probably wave in a certain way to trigger a selfie to be taken on your phone, for example.[13] [14] What gesture do you use for that? I can't tell you, because it varies. This lack of affordance – making it clear the control is available at all – and of standards for how these controls work makes them fairly poorly used and hard to plan for in the design of digital products on mobile devices.[15] [16]

To help you understand what these concepts and technologies mean here's a quick overview.

Kinesthetics is the ability to detect and understand movements of the body. Though traditionally a science related to orthopedics, physical therapy, and physical training, in this context it is used to refer to digital devices sensing the body's movements, and reacting

13. https://smashed.by/gesturedeathmatch
14. https://smashed.by/gesturecontrols
15. https://smashed.by/touchlessinteraction
16. https://smashed.by/kinesthetic

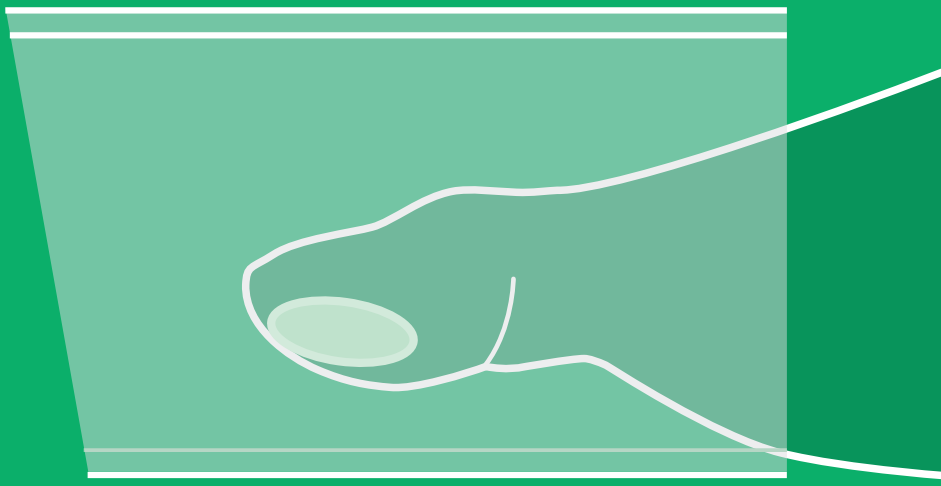
appropriately to proximity, action, and orientation. Kinesthetic sensing is performed in two basic ways: by moving a device, and by the device detecting your hand gestures.

Accelerometers — electronics that detect acceleration along a single axis — are the primary way to detect if a device has moved, and first gained popular attention with the Nintendo Wii. The hand controllers allowed the user to move their arms instead of a joystick or button pad to control the onscreen player actions. Mobile phones today use accelerometers to perform numerous actions to optimize the experience, but are also directly used for fitness apps; step counters, such as the ubiquitous Fitbit, primarily use accelerometers to measure steps, then guess at stride length to calculate distance walked.

Machine vision (mentioned above) can also be used without the tabletop or screen, to detect gestures, motion, and proximity relative to the sensing device. The most popular example was also a game controller, the Microsoft Kinect, which worked so well it is still

used in research projects and custom installations for things like museum exhibits. Very simple versions of this machine vision detection are supported by almost all mobile devices; a very low-resolution IR camera on the front face can recognize if a person is nearby, and if close enough will turn off touch sensing to avoid accidental activation with the side of your head while you are on the phone.

In this chapter we've covered the history of direct-selection and, especially, touchscreen technology. Next we're going to look in some detail at how the most likely technology for your project – capacitive touch – works so we can design better for the nuances of the technology.



CHAPTER THREE

Capacitive Touch

CHAPTER 3

Capacitive Touch

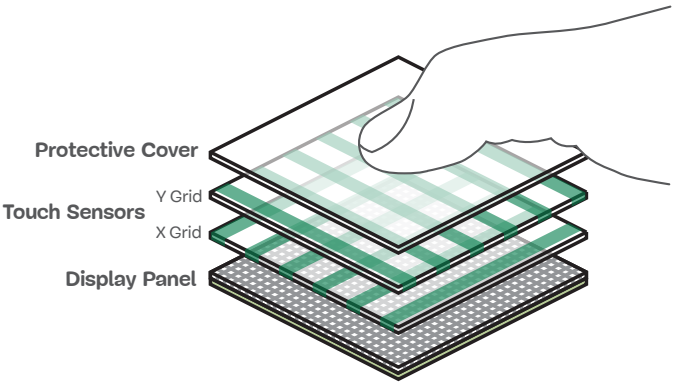
As I write this in 2021, if a consumer product has a touchscreen then it's a capacitive touch device. In fact, a lot of touch devices without screens also have capacitive sensors. Lamps, faucets, crosswalk buttons, light switches, and many other products might now contain capacitive sensors, but the common one we're likely to have come across is the trackpad or touchpad. Yes – a device I regularly refer to as "not touch," and which acts like a mouse instead, uses touchscreen technology.

The Technology of Capacitive Touch

Capacitive refers to the electrical phenomenon of storing charge. The sensing area has a very low voltage electrical charge. When a finger makes contact, the charge is transferred into the finger. The device detects the voltage drop and can tell there was a valid touch.

Single items like buttons and lamps use *surface capacitance*, and the entire surface (or a handful of discrete areas) is a sensor of touch. Capacitive touch, however, has become the standard for trackpads and touchscreens because of

a variant called *projected capacitance*. A grid of sensors is used, each with its own sensing circuit. A protective layer is on top of this, and when the finger approaches the surface it interrupts the capacitive field generated by the touch grid.[1]



Exaggerated vertical scale diagram of how capacitive touch sensors are arranged.

The protective cover is always something fairly rigid. Early devices mostly used plastic, as it was much less prone to damage from dropping. While trackpads and the brand new crop of folding screens still often use plastic, almost all other capacitive touch devices use glass covers.[2]

1. https://smashed.by/projectedcapacitance
2. https://smashed.by/glass

Glass, of course, is prone to cracking, and one thing a world of capacitive touch mobile devices has normalized is the broken screen. While there are regular improvements, this is still a risk and devices must be protected, which leads people to use phone cases and might influence user behavior.[3]

The quest for thinner and thinner devices has also increased the risk of serious damage. In previous generations, the sensing, display, and cover layers were separate, so a broken glass cover was just that. But now device displays are bonded, single units. A sufficiently cracked glass cover will also damage the touch or display layers.

Capacitive devices must have skin in contact with the screen to work. Finger contact is good in that it prevents accidental touches from random objects, but it is also a potential problem. The user can't wear gloves, as they block the electrical signals.

There are gloves with conductive fibers now readily available, and styluses which emulate the electrical composition of a finger, but both of these are still specialized products that may not be available for use at all times.

3. https://smashed.by/newtouch

Capacitive touch devices can mistake water on the screen – rain and other splashes – for touches, and water (including perspiration) can distort or offset the touch point. Conversely, a lack of moisture can prevent touch being detected as well. People with dry skin, living in dry environments, or older people – whose skin tends to be dry – have more difficulty using touchscreens and trackpads. A trick I learned from those who work with older people and computers is to breathe on your finger; this adds enough moisture for several minutes of device use. Musicians with their calluses, and anyone with wounds or hardened skin on fingers similarly do not have the necessary conductivity, so they too cannot always properly use capacitive touchscreens.[4]

Some device makers attempt to solve some of these issues adding kinesthetic control methods (discussed in chapter 2) as an alternative to touch input.

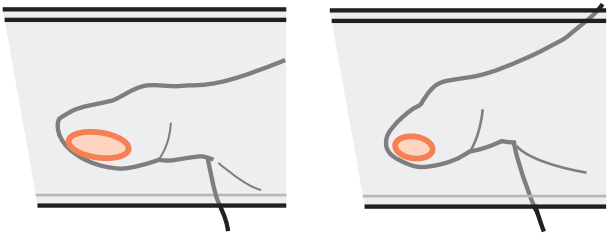
How People Interact with Capacitive Touch

One thing I skipped over in the technology review above was how capacitive touch knows when a user is touching the actual surface, and doesn't react when their fingers are just over the surface. And why do screen protectors work

4. https://smashed.by/touchresponse

and not mess up how touch works? The answer is signal to noise ratios.

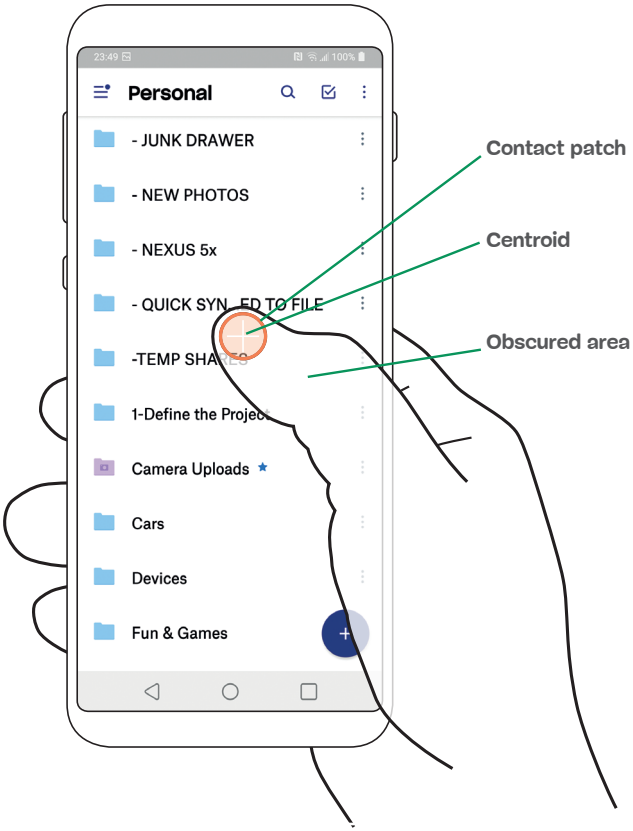
All sensors of all sorts have to filter out noise to find the useful signal. A touchscreen without filters would constantly detect "touches" as a finger approached it, or even as the air flowed over it. Plugging in chargers, changing between frequencies, operating around other electronics, or even the operation of features in the device such as the GPS or camera will add electronic noise.[5] Modern touchscreen devices can use other sensors to self-calibrate based on ambient conditions, to provide the best possible experience.[6] The proper setting for a touchscreen, then, requires a certain amount of the finger to be near enough to the screen. Due to the small vertical sizes, this means the finger has to be flattened against the display over a large enough area. We call this the flat area the *contact patch*.



Contact patches can vary in size based on angle, aspect, and pressure applied.

5. https://smashed.by/capacitivecontroller
6. https://smashed.by/capacitivesensing

Too small a contact patch and the finger can be touching the screen but won't be sensed as a touch. This is how misting rain doesn't get detected as a touch. Press a little harder,



The centroid is the geometric center of the contact patch.

more of your finger is in contact, and it gets detected. This threshold is very small, so even the slightest purposeful contact will be sensed, with very few accidental contacts.

The next logical question is to ask where the touch point is. Since it is impossible to touch only a single pixel, where does the large touch area register on the screen? The answer: at the *centroid*, or geometric center, of the contact patch.

On single-touch devices, this is not even calculated, but is a natural phenomenon. The voltage drain is centered at the geometric center of the contact patch. When two fingers are placed on a single-touch device, the phone just gets confused. The touch contact will be between the two, or it rapidly swaps between the two points.

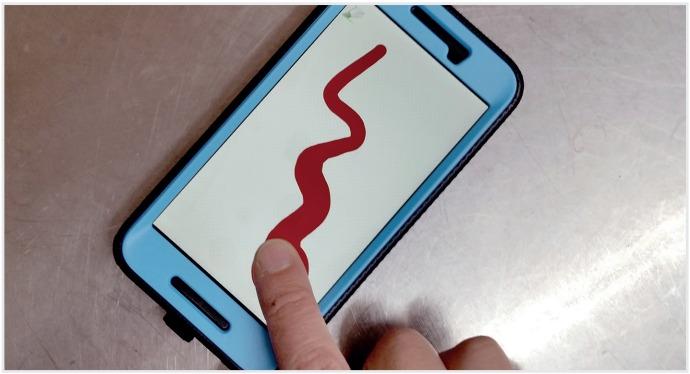
Some device development histories insist they offset the touched point to account for various perceived needs, such as their conviction that users expect to touch at the tip of their finger, or for parallax compensation.[7] However, in my observations I've found no evidence that these are true with any phones or tablets made in the past five years at least.

Most capacitive touchscreen mobile devices available today are *multitouch*, which means what it sounds like. The

7. Ken Kocienda, *Creative Selection: Inside Apple's Design Process During the Golden Age of Steve Jobs*. (2018, Picador)

touchscreen and processors are able to detect many discrete points. Five is typical, but ten or more is becoming common. You might wonder why we need to move from five to ten points, since most of us use no more than two points, for actions like pinch-to-zoom. Aside from OS makers always trying to add complex gestures (sliding from the edge with three fingers means something different from with four), the advantage is back to the signal-to-noise ratio issue.

Since a multitouch device can handle interference better, more touch points provide better accuracy in poor environments. When single touches are applied to a ten-touch device in the rain, the device can sense all the most prominent points, yet still tell that the finger is the most touch-worthy



A touchscreen drawing program that uses contact patch size to allow drawing wider strokes with more "pressure."

without being confused by ignoring the other contact points. Multitouch also allows for pressure sensing of a sort.

Multitouch allows for sensing the size of the contact patch; and the more points, the better it senses this. Since fingers are squishy (or *compliant* if you want to use the proper terminology), contact patch size – and especially changes in it from first contact – can be used as a proxy for pressure.

Most use of pressure is still confined to niches like drawing tools or better handwriting recognition. In theory it is used to differentiate hard and soft taps in some software, but much like kinesthetic gestures, these are not well understood and so not much used. Although heavily hyped, many of them have disappeared even in modern times, such as Apple's 3D Touch.[8] Since they are not universal, I will not be able to address any of them in detail in this book.

There have been many attempts to combine technologies, to sense pressure directly, or separate touch from click. Technologies like clickable areas were never really successful on mobile devices, and are slowly being replaced with strain gauges, and haptics – vibration that simulates physical actions like click – which do the same thing without movement.[9] Apple's Force Touch is an example of a branded

8. https://smashed.by/3dtouch
9. https://smashed.by/taptics

technology like this, but many other devices use it as well without a well-known name.[10]

Now we know how capacitive touch works, and we have addressed all the key variations and types of additional technology and capabilities that are likely to be added to it.

For the rest of this book I'll assume that all touch is capacitive touch, and that it is the only available interaction method. Naturally, this isn't always true, and sometimes other technologies are available. Capacitive touch is neither reliable nor cost-effective for very large displays, so the IR and acoustic technologies especially are still used for those. When we design for any specific device, or find the world has changed so more technology is available, we must always consider what other methods are used as well.

In the next chapter I will briefly touch on how the history of computing has skewed how we design for it. We have to be careful about using existing standards, methods, and techniques when building digital products with new technologies or new models of use.

10. https://smashed.by/forcetouch

CHAPTER FOUR

Standards, Assumptions, and Problems

CHAPTER 4

Standards, Assumptions, and Problems

It's often assumed that successful companies, operating system makers, and standards bodies all know exactly what they are doing, and their models and guidelines are above reproach and will always be trustworthy. But we know that's not necessarily true. People do all this work and they have their own constraints and biases, and they'll make mistakes as well.

Science is in the quest for truth, not its destination. We need to always be willing to question when new evidence arrives; and the move from desktop to mobile, changes in pointing technology, and better overall understanding of human behavior make a lot of old standards only narrowly applicable – or even altogether wrong.

Old Models and New Behaviors

Models dating from the 1950s and 1960s make a lot of assumptions about human behavior that have been proved false, but which persist nevertheless. In the US, if anyone wants to build a new highway, lane of traffic, or bridge, a key part of the proposal is a process called *travel demand*

modeling, which seeks to apply numbers to human decision-making on travel planning. Although established with all good intent, it's been well known for decades that it doesn't actually work and is almost always counterproductive. Adding more lanes to improve traffic flow because the model justifies it actually just adds more traffic and delays.[1]

From the dawn of computing, modeling human behavior has similar flaws of misunderstanding, misapplication, and overassumption. Associate professor Mary Sesto (doctor of physical therapy at the University of Wisconsin-Madison) wrote in her 2012 paper addressing touch:

> *ANSI/HFES 100-2007 recommends a button size of at least 9.5 mm, whereas ISO 9241-9 (ISO, 2000) recommends a button size equal to the breadth of the distal finger joint of a 95th percentile male (approximately 22 mm to 23 mm; Greiner, 1991). Although there is a growing body of research on user performance during touch screen use,* ***research that examines the touch characteristics used during button activation is lacking.*** *Touch characteristics quantify the physical interaction between the user and touch screen. These characteristics include the peak force exerted by the user, dwell time (the contact time on*

1. https://smashed.by/transportation

> *the button), and the impulse (area under the force-time curve).* ***Research on these characteristics is important because the physical interactions may affect user performance, fatigue, and injury.***[2] *(emphasis mine)*

There's a giant body of research into how keys, buttons, dials, and levers are interacted with, but until very recently there has been hardly any on the topic of touch. As shown in the quote, much of the touch research leads to contradictory standards, because it is based on old and poor assumptions.

The ISO standard mentioned in the quote[3] (now called ISO 9241-410) is one of the more commonly referenced standards about touch, and is based entirely on very old technology. The standard was first written when IR-grid touchscreens on PCs and control panels first entered service, and it has never been changed. Technology and our understanding of it has changed, but the standard persists. Here are three ways these old standards no longer apply.

Mobile Is Not Considered

Not too long ago, for a design systems project with a large industrial manufacturer, I ended up downloading and

2. https://smashed.by/buttonsize
3. https://smashed.by/iso

reading through around ten thousand pages of ISO, SAE, and CE-mark documents defining the international, US, and EU standards on the display of control panels, computer systems, the use of icons and labels, and more.[4] Although I started with the assumption that the current standards didn't cover digital displays, I was surprised to find it is even a little worse than that.

The entire set of standards are all very *machine era* – that's the term we use for the pre-digital design methods for buttons, levers, dials, and other mechanical or electromechanical controls that don't rely on display screens. Those that do consider the use of computing or display screens all concern workstations, desktop computers used to perform specific tasks with the assumption of full attention by the machine operator.

As I outlined in the introduction (and which I'll expand on later), mobile devices are now pervasive and are used very differently from desktops. Standards we use every day, including many modern digital-centric ones such as most W3C standards, default to desktop computing in a very and increasingly mobile era.

4. ISO is the International Organization for Standardization, a body that sets all sorts of standards across industries. SAE International (formerly the Society of Automotive Engineers) is concerned with transport industries. CE is sometimes read as *conformité européenne*, though this is not an official definition; to display the CE mark requires many products to conform with standards when sold or used within the European Economic Area.

The standards the W3C does have on mobile are mostly five years old, and much about the technology and culture of use has changed since then.[5]

Technological Assumptions

Standards like ISO 9241-410 assume that touchscreens are mounted to kiosks or computer workstations, and that they are all infrared (IR) grid-based. Of course, that's not how almost any touchscreen works, and not a single one of the billions of mobile phones and tablets we design for.

Touch grids can sometimes be just barely visible under the right lighting conditions.

5. https://smashed.by/a11ymapping

Technological assumptions like this have follow-on effects as well. For example, when the standards were written, IR-beam grids were very coarse and sensed very roughly, so targets had to be correspondingly large to assure they could be tapped at all. Today, sensor grids are about the same size, but can detect much more finely.

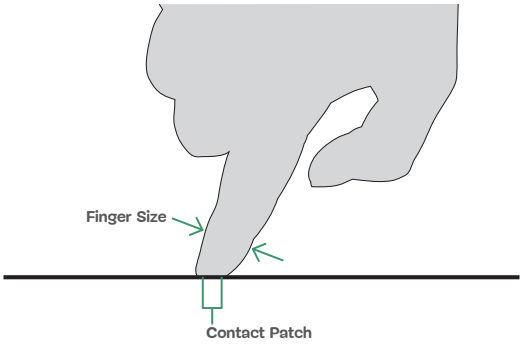
Partial grid sensing is fairly easy to perform on almost all of the available touch technologies I outlined in chapter 2. In the original implementation, a user's finger could land anywhere and might, therefore, either entirely interrupt one beam or only partly interrupt several. But, for technological reasons of processing power – and also because it was new so the issues hadn't come up yet – only one beam could consider itself interrupted and that was the contact point.

Today, the sensing technology for touch, even on IR-beam devices, allows partial and multiposition sensing. If a finger interrupts two beams a little, that data is processed to determine the position between the beams. This can be very accurate, as good as hundredths of a millimeter in accuracy. Since 1988, even very coarse sensing lights or wires have been able to accurately provide positions down to the pixel, and capacitive touch is so accurate there is not even a race to improve it more.[6]

6. https://smashed.by/userprecision

Human Assumptions

ISO 9241-410 also recommends a button size equal to "the breadth of the distal finger joint of a 95th percentile male." The reasoning is never really explained, and there's no clear logic behind it from a human factors perspective. Ergonomics considers how big people are to fit and reach, but doesn't assume that larger people are more clumsy, for example.[7] I will discuss touch accuracy in detail in chapter 6, but this supposition has since been demonstrated to be absolutely not true. Accuracy has nothing to do with finger sizes.



The size of the finger has no correlation to the contact patch or accuracy.

It is likely that the standards' writers were simply confused and conflated grid size with grid accuracy in early systems.

7. https://smashed.by/ergonomics

New standards are often built on old standards, without too much attention given to reconfirming the original results. In this way, many current standards, such as those set up by mobile operating system makers, can't be automatically trusted either. They often just guess, setting their design patterns early in the process with prototype hardware, often running on benches, and only used by a handful of people in the secret engineering team.[8] Many standards, like Apple's still-vaunted 44 px touch size and design cadence, don't even make sense from a human perspective, because the physical size changes from one device to the next.

Let's briefly look at another example of how a technical standard came to be, and what it means for utility and safety today. Automotive lighting standards – the color, position, size, and brightness of marker and signal lights – have been one of the greatest safety improvements of the past century. While seatbelts and airbags get talked about a lot in terms of lives saved post-collision, things like better lighting help avoid crashes entirely.

The SAE Lighting Committee has been the lead organization on this, and many standards set in the 1960s are still with us. But how were the standards set? Too often, by the committee deciding what looked good. For many decades, the official method for approving any standard, or choosing among several competitors, was consensus of 80% of the committee observing it with their own eyes. They'd just go

8. Ken Kocienda, *Creative Selection: Inside Apple's Design Process During the Golden Age of Steve Jobs*. (2018, Picador)

out at night, and see what looked good.[9] Aside from this being pretty much entirely the opinion of a single small cadre of healthy males of European ancestry making observations in good weather, the technology has changed. LEDs work differently from incandescent bulbs and plastic lenses in important ways.

Once set, standards persist and can lead us down bad roads if we use them when they are no longer applicable.[10] Today, we have a decade of touchscreen smartphone use, and better and better sensors; we know users work differently from in the all-desktop past; and we have a body of research that tells us how people really hold and touch their devices, and how to design for them.

The Normal Computer

Mobile methods of interaction are now so common, they are arguably the new normal for computing.

What we generally think of as a normal computer – the desktop or laptop *personal computer* – might simply have been an anomaly or accident of history. The technology now supports the pre-mouse methods of direct-manipulation user interfaces, interpreting the P in the WIMP concept, the pointer, as the user directly interfacing with the monitor instead of a cursor to map mouse position.

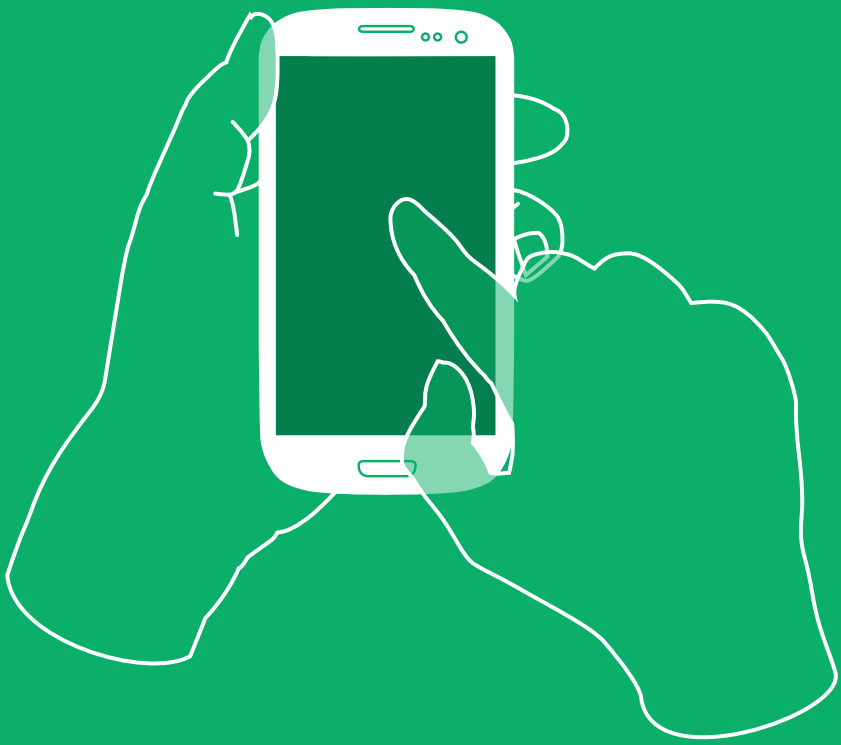
9. https://smashed.by/rearlightingsystems
10. https://smashed.by/federalstandards

Windows (the concept of movable application frames, not Microsoft's operating system) now seem unimportant as well – at least on smaller screens.

Considering the mass of mobile devices in the world, the ubiquity of their use, and the movement to touch and tablet modes of use for the PC, mobile methods of interaction are now much more common. We need to stop assuming that old desktop PC design principles apply to every product. We need to at least include, if not switch to, understanding all product design from this point of view.

In this chapter we discussed how standards can't always be trusted because the norms and principles of earlier times don't always apply to our work today. Technology, workplaces, and everyday life often change in ways that make their prior assumptions no longer relevant.

Now it is time to start considering how humans interact with mobile touchscreens. In the next chapter I will describe my research on how people use mobile devices, including finding out how people really hold and touch. Later, I'll walk through how to use all this information to design interfaces that make sense for the way people really use their mobile devices, and review the ways in which many sites and apps are already doing this successfully.



CHAPTER FIVE

Finding Out How People Hold and Touch

CHAPTER 5

Finding Out How People Hold and Touch

Now that we understand the technology of touchscreens and a little about how bad models of technology have led to poor standards and unusable designs, let's discuss how people interact with touchscreens. I'll also explain the research that led me to a new understanding of how people hold and touch their devices, and I'll debunk some well-known but incorrect models of how people use touch.

Device Diversity Reflects Human Diversity

Back in the early 2010s, the design community started discussing how everyone uses their phone one-handed and how to design for thumbs.[1] At first I was excited by this, because it reminded everyone that mobile was different and not just a tiny desktop computer. On further reflection, I started to realize the core issue of the interaction with a mobile device being different was a little bit off in several key ways.

1. https://smashed.by/thumbzone

I had already been designing for mobile phones for over ten years before this. During that time, I performed usability tests on many of my design projects, performed heuristic reviews of devices for competitors, and had even started undertaking some general observational research to start busting the assumptions I saw flying about as the mobile industry suddenly grew.

There were even many articles at the time about how the 3.5-inch iPhone form factor was the perfect size and would never change.[2] Even as early as 2010 there were many more Android phones in people's hands than iPhones, as well as several other operating systems in regular use.[3] Many of these were already much larger devices than the standard 3.5-inch iPhone, although participants in my observations didn't seem to have any issues using larger phones.

In 2021, Apple has around a quarter of the installed base of smartphones worldwide (Android has all the rest, effectively) and around half in the US, as I discussed in chapter 2. That's just the smartphones. Almost half the world, and 70% in important markets like India, still uses feature phones, which are quite capable devices, but with proprietary and closed operating systems.

Although Android is the clear market leader in smartphones, the vast majority of coverage of design, and almost all

2. https://smashed.by/iphonescreen
3. https://smashed.by/iphonevsandroid

design templates and design tools, are focused first and foremost on the iPhone and the iPad. As you read tips and tricks, data, or use design tools, be aware of the assumptions, preferences, and biases inherent in them.

THE DANGER OF PLATFORM BIAS

With the onset of the global pandemic, I got a new favorite story of bias in platform choices. In April 2020, the Indian government released a COVID-19 tracking tool called Aarogya Setu as an app for iOS and Android.[4] Everyone in India is required to use it, yet only 30% of Indians have a smartphone – and less than 2% of those use iOS. In a country with something like 800 million mobile users, there were about 80 million installs of the app after several weeks of availability. Compliance aside, at least 550 million users have been overlooked and simply cannot install it.

There are often legal, regulatory, or social safety reasons to build for everyone. Accessibility and common access laws increasingly require governments and businesses worldwide to build digital tools that everyone can use. While most commonly discussed as providing proper contrast, captions for video,[5] and other direct accessibility features,[6] simply locking out access due to platform choice is as much of a requirement.

4. https://smashed.by/contacttracing
5. https://smashed.by/videocaptions
6. https://smashed.by/adacompliance

Always try to find out what devices your users carry, what browsers they use, and understand their needs and environments, then keep all that as key constraints during design. Otherwise you are missing out on a vital market, or not meeting your obligations to existing customers or the public.

But let me tell you the story of how I found out how people really hold and touch, and how you can use that information when designing your next product.

Using Informal Research to Find the Facts

Tech reporting very often promotes the latest cool stuff, which can lead designers, developers, and project managers to rely on a far too narrow set of (often inaccurate) assumptions about how their products are actually used day to day.

Throughout my career, I've shared ad hoc observations or lightweight studies on social media and company blogs, and with project teams, trying to spread information about real-world usage rates,[7] and how *installed base* (how many people are using a product) is not the same as *market share* (how many were sold in a given period).

7. https://smashed.by/iphonepercentage

In 2012 I started traveling weekly for work and started to observe, write up, and share snippets of things that I'd observed (usually in airports) about how tourists had entirely different devices than domestic US commuters.[8]

People observed using a variety of mobile devices while waiting in an airport in 2012.

One day in 2013, while I was waiting in the airport and being annoyed by some bad assumptions I had just read – that phones are always held with one hand, and always tapped with thumbs – I observed how people's use of their devices proved those two beliefs clearly untrue. Any three minutes in a coffee shop or street corner or at that very airport showed me that people use their many different devices in many different ways.

8. https://smashed.by/commuters

As I typed into Twitter to complain, I realized that my observations were useful data. Instead of just moaning on social media, I realized that I could gather data and share these observations as real research.

Right there in the airport, I pulled out some paper, quickly created a note-taking sheet, and started recording what I could see. The few dozen observations I made then and another set the following week I used only as pilots, discarding the data from the final set. When I returned to the office, I created a robust document to record observations and modified the checkbox columns for the behaviors I had noted so far.[9] I looked at the data and decided to call that a pilot study. I revised the methodology, made a better recording document, and started recording valid observations. I also shared what I'd found on social media and granted a few others access to the digital recording system to collect even more observations over a wider geographic area.

Off and on for two months, ending on January 8, 2013, these few other researchers and I had made 1,333 valid observations of people using mobile devices: on the street, in airports, at bus stops, in cafes, on trains and buses, and wherever else we encountered them.

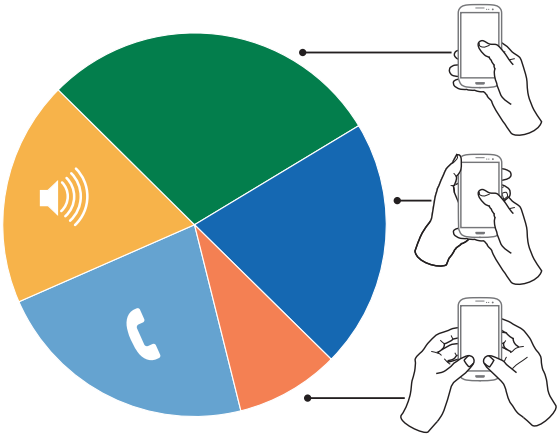
9. https://smashed.by/devicegrasping

This was an entirely observational study, with no attempt to either guess at or intercept the people observed to get demographic, device, or activity data. This allowed the largest possible pool of observed users, and avoided any bias or changes in user behavior from their knowledge of being participants (the Hawthorne effect).[10]

Some of the people I observed using their mobile devices in a BART station in San Francisco.

I noted that 780 of those people observed were touching the screen to scroll or to type, tap, or use other gestures to enter data. The rest were just listening to, looking at, or talking on their mobile devices.

10. https://smashed.by/hawthorneeffect



Summary of how people hold and interact with mobile phones from the 2013 research.

This research report was published at UXmatters magazine as "How Do Users Really Hold Mobile Devices?"[11] and its findings have been borne out by follow-on observations by myself and others.

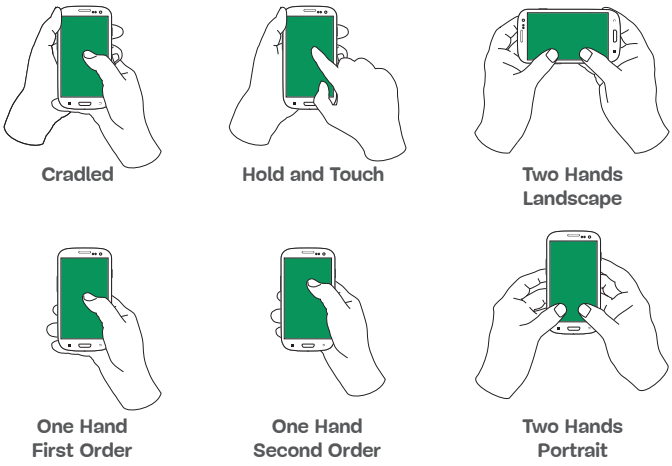
In 2014 I worked with Patti Shank at the eLearning Guild to gather another 651 high quality observations from 22 countries, in less public contexts such as schools, homes, and offices, and of a wider range of users.[12] Many more of these observations were on smartphones and on tablets as well, adding additional dimensions to the data, and proving that the trends already observed were corroborated across the range of device sizes and uses.

11. https://smashed.by/holdingdevices
12. https://smashed.by/mlearning

So, how do people hold and touch their phones? The answer is that old usability joke: "It depends."

People Hold Phones in Many Different Ways

When I began this research, I had a vague hypothesis that people don't all hold their phones with one hand, touching with one thumb. What I found was that there is not one dominant method, but many. There are about six basic ways people hold phones while interacting with the screen.



The six primary ways that people hold and touch their mobile phones.

That's just how phones are held in the hand. Very often, people put their devices down on surfaces, even more so as device size grows, such as when tablets are used.

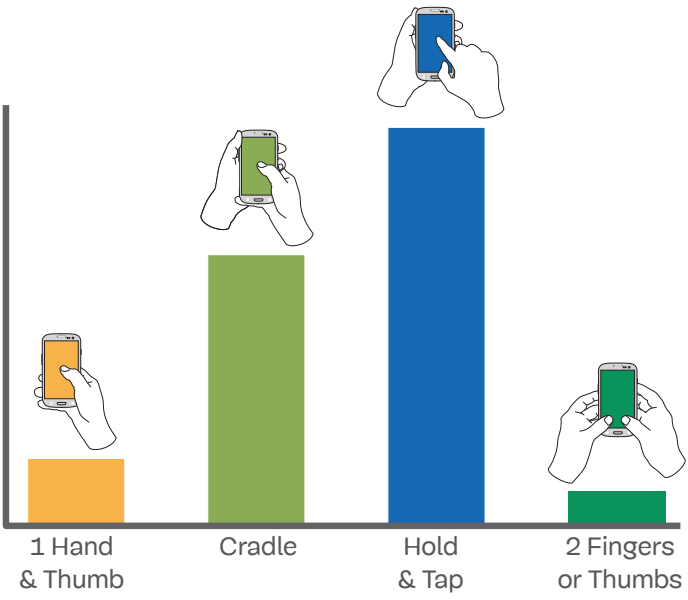
A 10-inch tablet set down on a table, with a keyboard.

While my observations found some overwhelming numbers – for example, about 75% of interactions are with one thumb on the screen – those get quoted out of context too much, so the numbers are misunderstood. Instead of that figure proving that the old "small iPhone, one hand" assumption is true, I found that fewer than half of all users hold the phone with one hand at the same time they touch with the thumb.

For smaller phones, I observed no less than 36% "cradle" the device, using a second hand for reach or stability. Even for

these types of phones, fully 10% hold in one hand and tap with a finger, giving a totally different interaction.

But how many exactly? It depends on the size of the device. When I parsed the data to include only what we then called *phablets* – a portmanteau of *phone* and *tablet*, but are now just considered somewhat larger than normal phones – there was far less one-handed use.

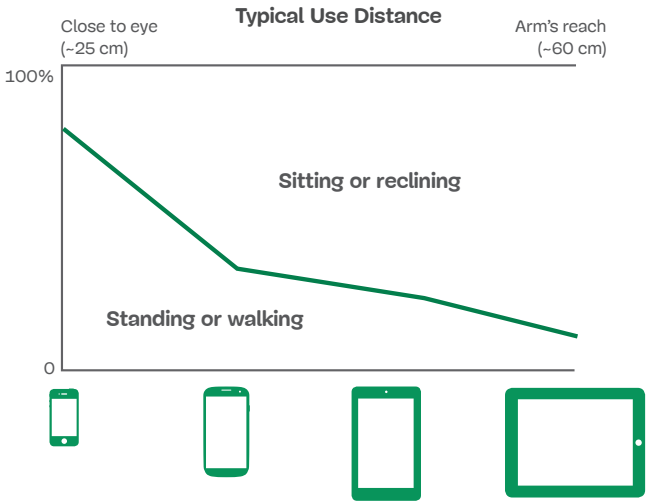


Hold and tap methods for phones with screens over five inches.

Device use varies not just by how they are held, but where they are placed and by what the user is doing, in very predictable ways as size changes. I found that:

- People use smaller devices more often in the hand, and use them more often when standing and walking.
- People use larger devices more on surfaces and in stands, and use them more often when sitting.

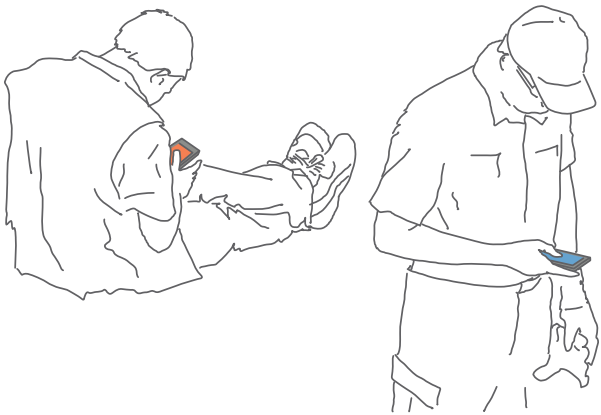
In general, the smaller the device is, the more it is used on the move. (On the move doesn't mean in transit, on buses or trains, but when walking around the house or office.)



Body position and activity as they correlate to device size.

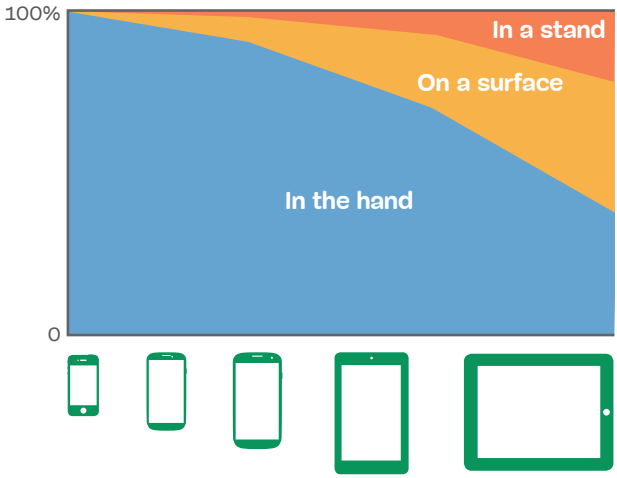
Instead of finding time to stop and use that tablet on the table, or sit and type on a computer at the desk, the device is used persistently. While we can dig into data and find specific percentages, since it's all probabilistic the trends are best understood in these graphical summaries.

We all think of our typical smartphone being held around 12 inches (30 cm) from the eyes and when walking around, but that's not always true. As devices get larger, they are used further away, then less and less in the hand at all. When people set devices on surfaces, they generally lean back into more comfortable working or viewing positions, and the distance increases. In stands, the screen is about as far away as on a desktop computer or a laptop.



Two typical stances for users engaging with mobile devices.

Whether it is larger phones held further away, or tablets in stands being used at desktop distances for typing, distance from the eye can be surmised by device class. You can make the size of elements like text, icons, inputs, and buttons grow or shrink – by design or using responsive principles to adapt a website or app to every device – so everything remains readable.



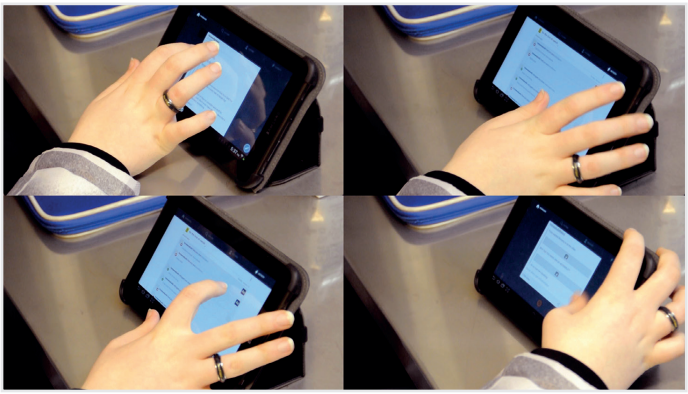
Device position trends as a function of device size.

You can always come back to look at these charts or my raw data for details, but what is most important is to acknowledge the concept that device use varies by device size. The *fragmentation* of screen sizes and features isn't a bad or negative thing: that's a mistake. It turns out that people buy

different devices because their needs are different, and they use their devices in ways that correspond to the size class.

How People Hold Changes All the Time

People change how they use their devices regularly throughout the day and sometimes from one tap to the next. They change their interaction methods depending on the device type and size, their needs, and the current environmental and onscreen context.



Screenshots from a usability test video in which the participant changed interactive methods four times in 14 seconds.

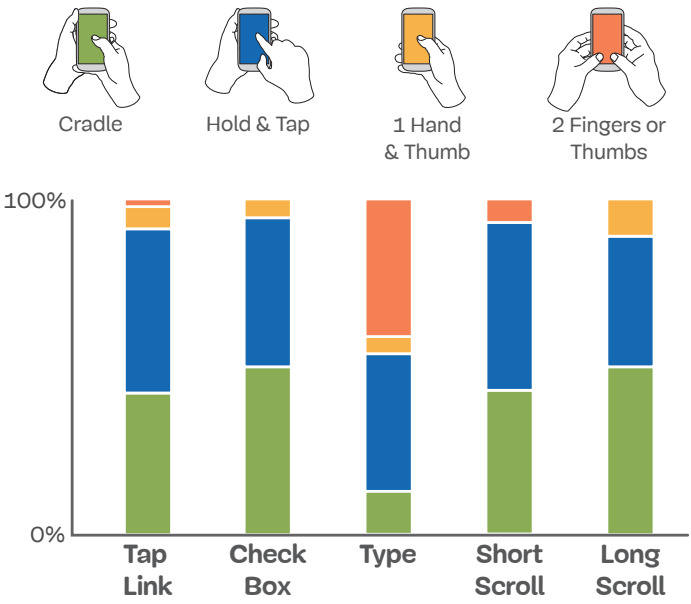
In general, people seem to do this shifting without consciously thinking about it or noticing. That means we can't

ask them to change by forcing them with design. It also means we cannot observe ourselves well enough to predict or perform ad hoc testing.

Once we all understand and accept that users are comfortable moving their hands around, it gives us a lot more insight into the overall rates of holding the phone in various ways. From my observations, people will generally:

- Carry the phone around and view with one hand; might scroll a bit with their thumb while reading an article.
- Switch to cradling or using their other hand to select, for more lengthy, focused scrolling, and for interaction and selection.
- Around 40% of the time, users switch to both hands interacting with the screen when typing – two thumbs for small devices, and more often two fingers as the screen size grows.

Remember these points are based on trends and probabilities. There are outliers and other cases. One of the more common of the less-used styles is positioning the phone so either thumb can be used with little or no shifting of grip. People tap or scroll with whichever one is most convenient at that moment, as shown on the graph.



Observed rates of touching the device in various ways, for several onscreen tasks.

Tablet use is a little more variable because the tablet allows for more engaged input or creation interactions, such as longer typing sessions.

Tablet users are more likely to place their devices on a surface and use the fingers to type, much as with a keyboard, or they may just deploy a keyboard stand or case to type instead.

Regardless of the details, it is clear that people switch back and forth between methods freely, and we can't assume any one method is preferred or used exclusively. Instead, we must plan our designs to work well for all user interactions.

People Are More Alike Than Different

During the study I worked on with the eLearning Guild (see above), we were able to gather some information about the users we observed as they were already co-workers or students. Statistical analyses along many axes were performed and what I found was… nothing.

None of the factors we examined (region, language, gender, and age) made any significant difference to how people interacted with their mobile devices. As described and illustrated above, the factors that made all the difference were device size, onscreen activity, and environmental contexts (such as sitting or standing).

This is great news to me! It reinforces that people are people and vary more as individuals than they do by cohort, just like our unconscious bias may lead us to believe they vary owing to age or disability. We can safely design products for humans and expect all of them to act in the same range of ways. Variations are individual, and contextual, but all people's behaviors are the same.

Handedness Is a Bit Confounding

I keep coming up against one very odd piece of data: handedness is not normal. I mean "normal" in the statistical sense: that there is a disparity between the percentage of people who have a particular hand preference and the percentage who use their phone with one hand.

As well as shifting the grip on their phones, people also switch hands. Right-handed people sometimes tap with their left hands, and left-handed people sometimes tap with their right hands. People shift in lots of ways. The problem is that these should even out and handedness should be the same as total taps using one hand or the other. But it doesn't.

Around 10% of the world is left-handed,[13] and when I have asked for handedness in research studies I have found participants self-report at that rate. However, of all the observations and studies I have performed and reviewed throughout this book, 15.4% of taps were performed with the left hand.

This is not a statistical anomaly or a bug from the test protocol, or an error in the data-gathering. Follow-up studies have shown the same results with the same higher level of left-handed taps than would have been predicted based on the common understanding of the incidence of left-handedness in the population.

13. https://smashed.by/handedness

Unfortunately, this observation isn't about to turn into some great revelation of what is going on. I still have no good idea why this is. But since the same numbers come up repeatedly, it is clearly something intrinsic in how people work, or something I have yet to identify in the way language, devices, digital experiences, and environments are configured. It may simply be a reflection that handedness is not the fixed or easily measured factor we have long assumed it is.[14]

Luckily, since we're already assuming that we design for every type of interaction regardless of orientation, grip method, or touch method, we also don't assume that one hand matters more than the other, do we?

If anything, the increased preponderance of left-handed taps just reinforces the importance of designing for all interactive methods, and we shouldn't disregard those in design, simulation, or test.

Designing for How People Hold, Touch, and Type

These findings are not esoteric or academic. We can immediately put them to use, turning what we've learned into some basic guidelines for mobile design.

14. https://smashed.by/handperformance

DESIGN FOR ALL SCREEN SIZES AND ALL ORIENTATIONS

Respect user choice and user actions. Design not for one or a few pixel sizes but fluidly, for everything reasonable.

Support landscape and portrait. Start with landscape for tablets and portrait for phones, but support both. Yes, even for things that are traditionally only one orientation, like video playback. Let it work the other way as well.

MAKE NO ASSUMPTIONS ABOUT TOUCH INTERACTION

Don't assume that people can only reach one part of the screen, or that their thumb will cover items below the icon, so we need to place labels above. The shifting of grip and different ways of tapping mean we can't predict this.

We'll talk a lot more about where people actually click in chapter 6, and how to design for visibility around opaque fingers and thumbs in chapter 7.

For now, just remember that the variability of interaction simply means: don't overthink it.

SUPPORT ALL INPUT TYPES

Tablets with keyboards and computers with touchscreens are getting more and more common. With laptops increasingly converting to tablet form factors, there's no longer a clear distinction between what is a *computer* and what is a *mobile device*.[15]

Just like I have discussed how people use mobile devices in many ways – switching how they tap and hold based on context and interactive needs – the same seems to apply to all interactive systems.

When observing users, all this makes perfect sense. People use the tablet mode to poke at the screen while walking around and set it on a table to type, and so on.

But none of these are detectable technically. We cannot modify the interface to be touch- or mouse-friendly based on user context. Since we can't tell how people will input or interact, we need to make sure that every digital experience is designed and built to work properly with:

- Touch
- Onscreen keyboard
- Hardware keyboard

15. https://smashed.by/ipadkillers

- Trackpad
- Mouse
- Screen readers and other accessibility methods

In addition, assume people switch. I have observed plenty of interactions with all these devices as well, and along with the assumption that people switch grip and hands, we can also assume that users also switch between keyboard, mouse, and trackpad. People will type, use the trackpad, reach up and tap the screen or drag something, and then scroll with arrow keys.

Designing for every possible interactive method means a typical "desktop" website needs touch-friendly icons and links. We can't rely on hover states (or tooltips), and we need to consider how much content might be hidden behind an onscreen keyboard. It also means our mobile apps and websites need to clearly display what is highlighted when someone tries to scroll through it with a keyboard. And, maybe, determine what the highlight method is so as to not conflict with the design and remain readable.

Designing to these principles also will make our websites or applications more accessible. It will be easier to navigate via screen reader, which is another huge market far too many digital tools do not support.[16] Even since I started writing

16. https://smashed.by/videoads

this book there have been new laws, regulations, and court cases deciding in favor of universal access. Accessibility compliance is driven by legal and regulatory requirements,[17] so if you are not already paying attention to this, you will have to soon.

Screen size, as responsive web design assumes,[18] is no way to determine if something is a portable touch device. Adding device detection is a good alternative. However, I simply say we need to start assuming that everything is touch-capable and can also have a hardware keyboard attached.

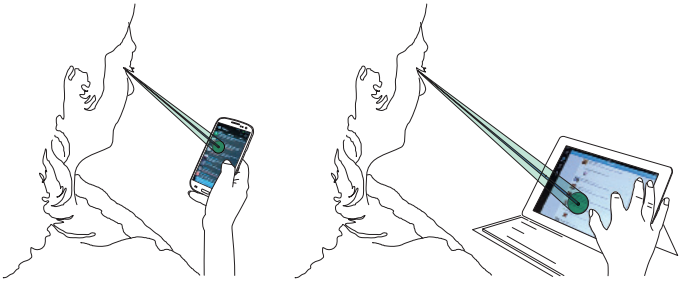
DESIGN FOR DEVICE CLASS DISTANCES

Design larger type, icons, and interactive elements on larger devices, because they are more often used further from the eye. Web and app design always assumes everything is based on pixels, or at least *device-independent pixels* – because screens are so high-resolution now – scaled to a multiple of the display resolution. A key problem with this is that pixels are not one fixed universal size, like millimeters, but change based on individual devices.

Here, we are finding a different issue to add. Since people use devices at different distances, they see things smaller and smaller as they move further away. When searching for design recommendations, you might have noticed the

17. https://smashed.by/banka11y
18. https://smashed.by/adaptivevsresponsive

suggested type or icon sizes are smaller on mobile phones than computers. That's not just cheating because phones are small, but because they are used closer to the eye. It's also why it's sensible to watch TV on a phone, as it is used closer so it takes up a similar percentage of the field of vision.



Angular resolution means items further away from the eye appear smaller.

This change in perception means we need to understand not the size but the relative size, as it changes based on distance. This is the *angular resolution*.[19] (There's a formula to calculate it, and in chapter 10 I'll make it easier to remember and give a simple cheat sheet of suggested sizes for each device class.) For now, remember that on smaller devices, smaller items can still be read, and for tablets we must assume they are set down on tables or actually are convertible tablet computers, so we must make all text and icons big enough for desktop or laptop computers.

19. https://smashed.by/physiology

Angular resolution means at normal viewing distances video on a phone is as "big" as it is on a tablet, computer, or TV.

In this chapter we discovered that the conventional view that mobile phones are held with one hand and tapped with the thumb is much more complex. People hold and tap phones and tablets in many ways, and they shift all the time.

We cannot just design for one phone and one way of using it, but need to allow designs to work for all devices, and in any way people might use them.

And we've learned about angular resolution, and how on-screen sizes don't matter as much as how far away people are when looking at them.

Checklist

- [x] Remember that users are real people with their own lives, needs, and preferences. Design for every user. Accept that users change.
- [x] Plan for every device that will access or install the product, not just the team's favorites or the most popular one.
- [x] Devices are not held in any one way, but a number of different ways and individual people shift how they hold and tap all the time. Design for all methods.
- [x] People use phones almost entirely in the hand, and largely on the move.
- [x] People use tablets much more often while sitting, and with the device in a stand or set on a table.

- ☑ Tablets are used more often landscape (horizontal) and phones more often portrait (vertical). Design for those as the baseline for each platform, but recognize there is enough variation that most of our interactions should be made to work in both orientations.

- ☑ Touch and tablets are everywhere. Most "laptops" are or can be tablets at a moment's notice, with no clear way to detect it, so design all interactions as though for mobile touchscreen devices.

- ☑ Typical use methods per device class based on screen size can be used to predict viewing distance, and items onscreen changed to be readable for typical users.

CHAPTER SIX

Touch Accuracy and the Center-Out Preference

CHAPTER 6

Touch Accuracy and the Center-Out Preference

My foundational research has shown that there's no single way people hold and touch their phones. User choice, variability, and change are real. All device sizes are valid. We have to plan for the many ways people will use their devices.

In this chapter I'll explain how additional research, originally conducted to find how people adjust based on tasks, eventually provided me with an insight into touch target sizes that work for everyone. Most importantly, I discovered that because people seem to prefer to touch and view the center of the screen, it led me to design not from top to bottom, but from the center of the screen out.

The F-Pattern

I didn't start out doing UX design – it didn't exist back then. No one told me about human factors, or human–computer interaction, so I ended up in the art and design school at my university. I learned many of the same lessons from two points of view. Art classes mostly taught about the way people perceive, view, and think about visual media based

on tradition and convention. Design taught mostly the same lessons about how people perceive things, but instead based on mathematical theories and the results of early marketing research.

When we first start learning to read, we begin to learn that language is oriented left to right, then top to bottom – in Western languages at least. In art, design, or user experience education, this is extended to a general lesson, that everyone consumes all content top to bottom and left to right.[1] But it turns out that is an oversimplification and is not always true. Designers of "glanceable" items like TV advertising, billboards, posters, and road signs know it is not.[2]

More recently, as digital establishes its own guidance about design, one lesson is that we scan web pages in an F-pattern, reading lines across then down. The F shape comes from the way people read full titles, the first few lines or paragraphs, but tend to scan and pick up just the first few words of the remainder of the page contents. However, that we scan this way is not entirely true.

The F-pattern is just one way people scan web pages. It was developed and shared in 2006 by the Nielsen Norman Group as a way to understand how people consume web content.[3] It turns out that people scan or read in different ways, depending on the way a particular page is designed.

1. https://smashed.by/designlayouts
2. https://smashed.by/designtips
3. https://smashed.by/textscanning

The F-pattern was never a goal for design, but somehow it entered the consciousness of designers and marketing managers. It has since become a misapplied pattern that has led to many unusable web page designs and misguided design suggestions. An updated article from Nielsen Norman Group even warns of this and suggests ways to not fall into F-pattern scanning as it is often not what we want.[4]

Apple's design guidelines from 2014, which indicate that the most important content should be in the upper left corner on mobile phones, which we now know to be not very useful guidance.

4. https://smashed.by/fpattern

People Touch the Center

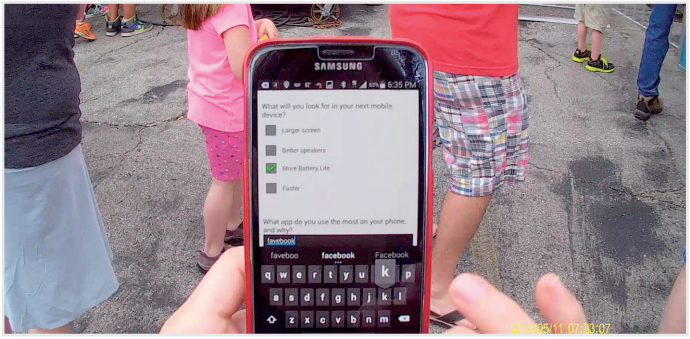
As well as the research discussed in chapter 5, by the mid-2000s I had completed my first research on touch in an attempt to begin to understand what I was seeing during usability tests. I performed a test to measure finger pad sizes and touch accuracy using paper and an inkpad, on a broad range of users of all ages. I had been trying to set standards for how to interact with mobile phones for years, and gained some interesting insights but despite more or less settling on a set of sizes, I was never truly satisfied.

A research participant in 2008 using a flip-style feature phone with attached camera "sled" to observe their interactions.

Because I still had many questions about how people were using their devices and why, I conducted additional research in 2014 to explore more specifically how people use their devices contextually. I built a web page that worked on

all device sizes and recorded 31 users – some remote and unmoderated, and others moderated through person-intercept observations at a town carnival. These tests observed people performing various basic tasks such as scrolling, selecting items, using menus, entering text, and watching video.[5]

Moderating the carnival intercept portion of the touch research study.

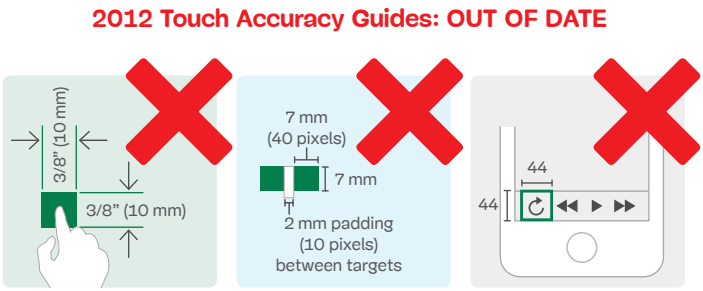


A still frame from the video recorded of a test participant in the carnival intercept portion of the touch research study.

5. https://smashed.by/realworlduse

At first, the results were confusing and inconsistent. A number of my hypotheses didn't pan out so I had to try to determine why. Luckily, all the sessions were recorded; the remote sessions were recorded as the only way to gather data, and the intercept study encounters by way of the participants wearing video-recording glasses. These recordings allowed me to review the tests from the participants' points of view, and gather different data from what I had originally written down when I reviewed the test results.

One of the items that confounded me was touch accuracy. I had been using a 10 mm circle size for years, and every other expert and operating system maker had their own value. My hypothesis was to confirm or update that single number for touch accuracy across the screen.



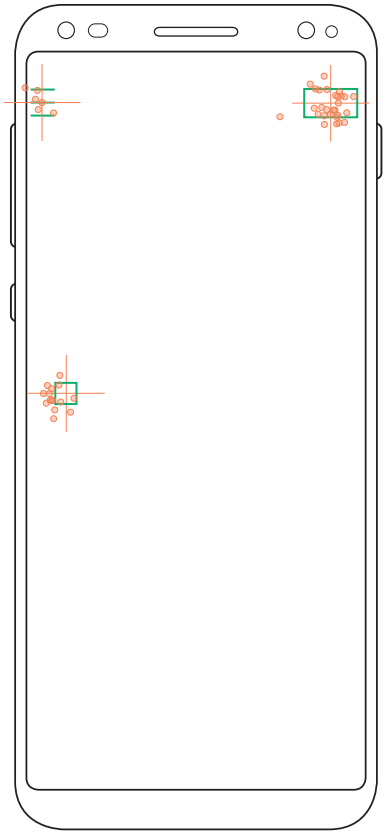
A selection of touch accuracy guides from 2012, including (left to right) those from the author, Microsoft, and Apple.

To try to improve the effectiveness of these guideline sizes, I and others such as Nokia and Microsoft had added an additional margin of safety or a dead space between adjacent tap targets. My previous distance choices were entirely based on what it took to avoid interference from research experience, with no logical reasoning or mathematical model to explain it.

The results of the 2014 research did not immediately show what I expected. Instead, I found a large amount of variation in the distance from any particular target that was needed for a tap to be registered as accurate. Although both icons were the same size, the menu taps had many more misses over a wider area than targets in the middle of the page. I thought that it was based on type of control, but the data didn't line up there either, and form or list selection taps would also be inconsistent.

I cut the data several ways to no effect. But when I plotted them to the position on screen, all became clear.

Hidden in plain sight was that accuracy varies by position on the screen. The diagram overleaf shows the patterns people seem to rely on. People touch more accurately towards the center of the screen: for every tap, for every device, and for almost every person.



Touch accuracy test data recorded for several areas of the screen.

Once I understood this context, I double-checked my work, rechecked the older data I had collected on touch, and then looked for additional research. With this more specific search terminology, I was able to find academic research that also proved the same things. A half dozen

studies were sitting there, buried in academia.[6] One of my favorite studies was a game that gathered touch accuracy by position, from almost 100,000 users, recording 120,626,225 touch events.[7]

Combining these studies with the new information I'd collected, and performing some statistical analysis to normalize the data and be sure I had confidence in it, I came up with accuracy by zone across the screen. One of the easier to understand models of that is shown below:

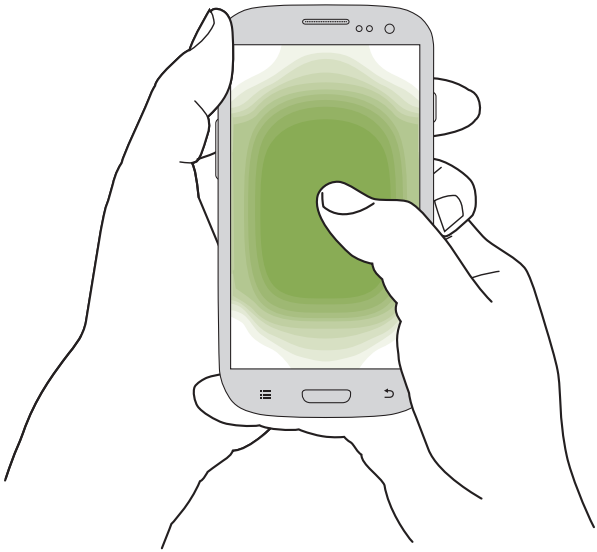


Chart showing that the most touched areas of the screen are in the center.

6. https://smashed.by/thumbtarget; https://smashed.by/softbuttons https://smashed.by/targetselection; https://smashed.by/ballotdesign https://smashed.by/walkinginteraction https://smashed.by/touchdesignissues
7. https://smashed.by/touchperf

So far, my observations and research had led me to these conclusions:

- Touch targets in the center of the screen can be as small as 7 millimeters.
- Corner target sizes must be about 12 millimeters to ensure they are accurately touched at least 95% of the time.
- The old 10 mm target I had promoted was not wrong on average. But interactions don't care for averages.

With this new data, we could understand much better what users were doing and how to design for human interactions. Below is another of the accuracy chart representations, with labels to indicate the typical touch accuracy within each zone.

Designing for Touchscreen Kiosks

Plenty of my research findings might also seem to apply to touchscreen kiosks, or permanently mounted screens, like bank ATMs. Mostly as a result of being immobile, users interact with these a bit differently, so the touch zones do not seem to be the same. Numerous other issues arise with them, chiefly the fatigue of unsupported arms – an ache called *gorilla arm*. (https://smashed.by/largescreenux)

There are plenty of kiosk design experts, so I have

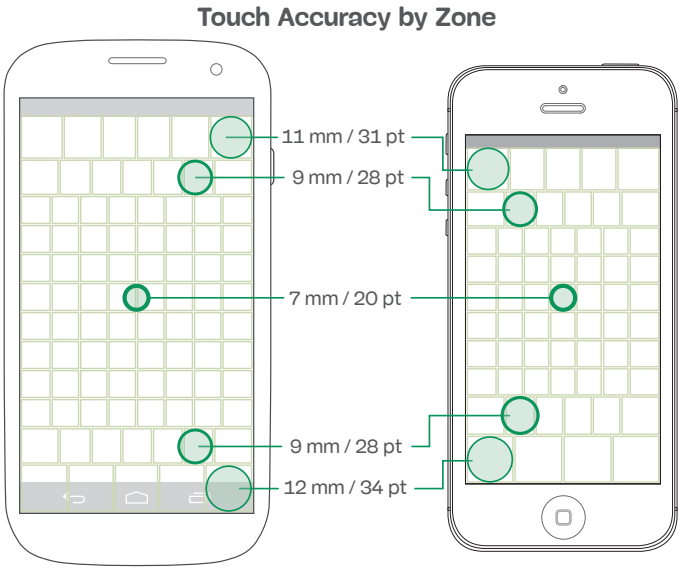


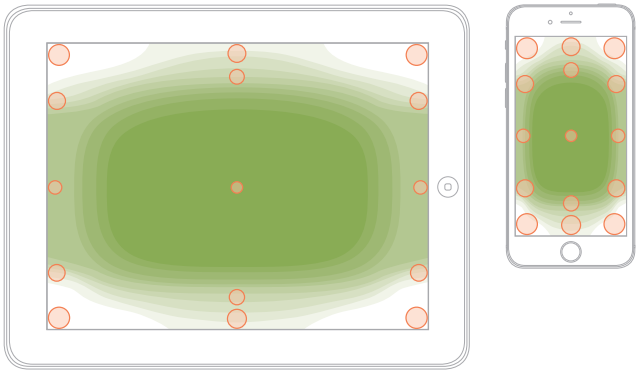
Chart showing touch accuracy for specific parts of the screen.

deliberately not spent much time researching the data. If you are designing a kiosk, even just an iPad on a rigid stand, you should look into the relevant kiosk design heuristics and best practices (https://smashed.by/kiosks) and not assume that because

it's an iPad it will work the same way as portable, individual-use devices will.

Many of the remote unmoderated research participants in the 2014 study used tablets. I had not encountered or much considered tablets as part of my work, so simply set aside their data as not relevant and didn't review them at all. But after I figured out the accuracy zones, I went back to this tablet data to find the touch accuracy for those as well.

The results surprised me. I found that on tablets, all the way up to the largest almost 10-inch-diagonal iPad, users had the same touch accuracy across the same relative areas of the screen. I have found that these accuracy levels (as shown in the charts above) are human-based, so apply to all portable touchscreen devices, from the smallest handset to the largest. The same touch charts are shown below sized to a landscape-orientation tablet.



Touch accuracy is relative across both tablets and phones.

Yet more analysis of the data followed. The academic studies I mentioned earlier also contained timing data. Further review of the recorded sessions in the 2014 tests showed that touches at the center of the screen were faster. Users seem to intrinsically recognize they are worse at interactions close to the screen edge, and they slow down for those.

We know now that many of the existing guidelines are wrong: they are measured in technology-oriented units, not in human terms. Apple and Google still promote touch size standards in device independent pixels as their one and only ideal touch target size; yet these measurements vary by device resolution.[8]

There are at least half a dozen physical sizes a single pixel-based size can be rendered as, and real-world sizes can vary by up to 30%. Fingers, however, aren't measured in pixels. We need to design for physical sizes and account for pixel scaling factors.

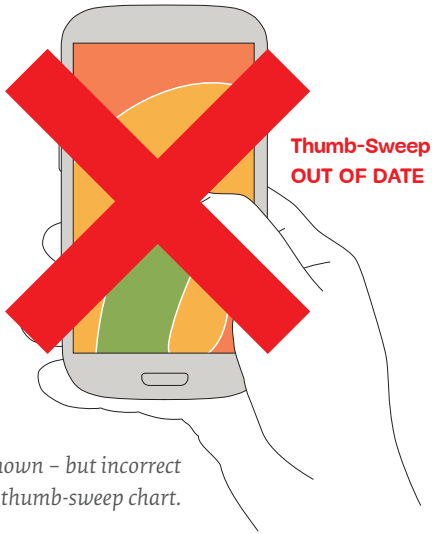
Just like we found with the data on device holding in chapter 5, touch accuracy doesn't vary by finger size or device size, or by age or experience. It doesn't vary by sex or gender, language or region, time of day, which hand is used, or even if it's a thumb or finger being used to touch.

8. https://smashed.by/buttonheight

More accuracy in the center of the screen means we can put more items in the center. At the edges however, we can put just a few functions or tabs, as people cannot touch them as effectively and will suffer more errors because they are too small, too densely placed, or both.

There Is No Thumb Zone

Since I first published these findings on touch zones, numerous other design researchers have conducted their own studies and reproduced the same data themselves,[9] with



The well-known – but incorrect – thumb-sweep chart.

9. https://smashed.by/uiergonomics

their tools, in their regions. The results remain consistent and I've integrated their findings into my work.

But when you ask most designers how people touch their phones, they will say with their thumbs. They will talk about *sweep zones* or reaching and stretching, and the ideal *thumb zone*. This persistent misunderstanding is the principal reason I am writing this book.

Where do most practicing designers decide that people hold and touch their phones like this? I've shown that barely half of all onscreen interactions are with the thumb. And the centering data I just shared has no indication of thumb-sweep zones either.

While writing the report for the original research on holding methods, I asked my co-workers to hold their phones in the various positions so I could take photos. Many of them insisted that they never held it in the ways I described, but instead would hold it in one hand and use their thumb. I'd wait a few minutes and then sneak a photo over the top of the cubicle when they used the hold I was looking for.

I got the photos I needed of the natural grip methods; and they found out they totally do use many methods themselves. This is why usability research is superior to self-reporting surveys: people are terrible at self-observation and self-

analysis. Everyone thinks they hold phones one way, and, in turn, they assume all people work the same way they do.

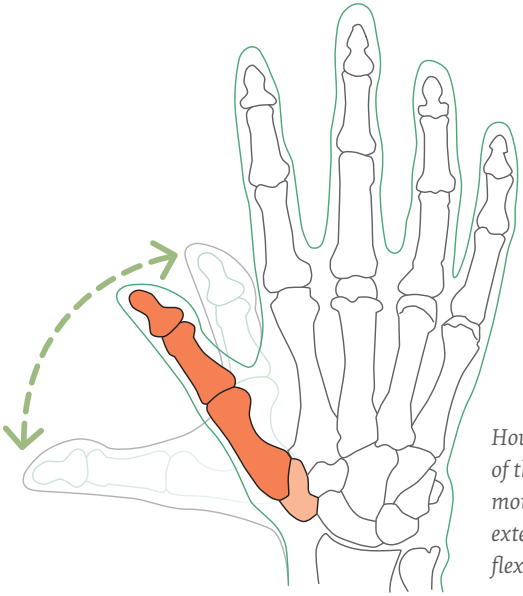
Another observation I have made in every usability study is that people use the back button all the time. In fact, it's usually the most used button on the screen, even when it's in the upper right corner. So I realized something else must be happening about reach. This sort of finding made me continue doing my own research.

HOW THUMBS REALLY WORK

Why do people hold devices in this way? Let's start with the fundamentals, which always begin with human physiology: how our bodies actually work. What do we know about human thumbs? Well, we're not a series of hinges and boards like a machine. Human motion is not that simple. The bones of the thumb extend all the way to the wrist, making assumptions of what constitutes a thumb often incorrect.

Thumbs move in a sweeping range – of *extension* and *flexion* – not from the point at which it connects with the rest of the palm but at the carpometacarpal joint way down by the wrist.[10] The other joints in the thumb let it bend toward the screen but provide no additional sweep motion. The ability to bend the thumb is important because, while the thumb's free range of movement is in three-dimensional space,

10. https://smashed.by/thumbflex



How the bones of the thumb move in extension and flexion.

touchscreens are flat. Therefore, only a limited portion of the thumb's range of movement maps onto the phone's single-axis screen.

To add more confusion, the thumb's joints, tendons, and muscles interact with the other digits – especially the index finger – which changes how much they can move or how strongly they grasp.

When the fingers grasp a handset, the range of motion available to the thumb is more limited. But by moving their

fingers, users can change the area of the screen their thumb can reach. Now we can start to get a hint about why people shift their grip. Because they are humans.

Tapping and Shifting

The key takeaway from chapter 5 was that all people hold their devices in many different ways, and – much more importantly – that this isn't an individual user preference such that one person holds one way all the time. People shift how they hold their devices based on their tasks. For example, people often sit scrolling with one hand, then shift as they change grip methods to interact more intensely with the device.

This change in grip also has an impact on where the screen was touched. One hand and thumb is most often used to touch fairly small areas along the center of the screen. Watching users, this is how most scroll works: very short swipes along the center.

This visualization of touch as odd curly shapes is one I made up before I understood the overall results. Focusing on just the one-hand or first-level cradle yields these patterns that can appear to correspond to the thumb-sweep charts. However, it is important to recognize first of all that

Tap areas as they correlate to preferred holding methods, for right-handed interactions.

these shapes are only right side touches; left side touches mirror them. Second, these are very narrow observations.

Remember that people shift their grip all the time. While I could share charts showing how individual hold methods have these inverted-comma sweep areas, everyone's hand and grip is different, phones are different sizes, and ev-eryone shifts all the time. The exact position of any touch

or gesture area changes from one device, user, and one moment to the next.

Larger interactions or mode changes, from choosing to start using a menu or pressing back, to starting to type, often are accompanied by a shift in grip. A second hand comes to the phone, and the user might switch to tapping with a finger instead.

Such action is still preference and not a hard-and-fast rule. Most people use the indicated methods but not all of them. People absolutely also shift the phone in their hand to allow their thumb to reach the back or menu button while still holding with that hand. They didn't stretch their thumb, though – that's impossible. And they suffered no discomfort because it's all voluntary, and people generally don't cause pain to themselves.

Some of you may now jump up and say, "Ah ha! The thumb-sweep zones are right and we all need to avoid faraway corners!" But for one bit of data. No one in any research I have performed or read avoided any tasks that required shifting their grip, had notable delays in performing the tasks, or had any measurable complaint or dissatisfaction about it. I have done a number of usability studies for products since I conducted the touch research, and I have not seen anything

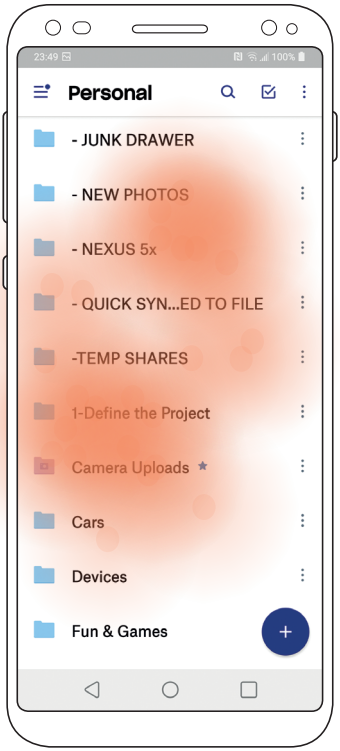
in any of those that indicates people won't shift or reach to press faraway corners.

People adapt and change. And part of life on small touch-screen devices is constant shifting and adapting to interact as they need to and want to. On average, across all people and their entire experience tapping and swiping on their phone, the symmetrical chart of touch accuracy is true. Don't over think it and try to identify specific zones with deep-dive ergonomics or because you think you hold the phone one way all the time.

People View the Center

The research I conducted on touch included a lot of specific tasks to see how people changed their interaction for different functions and interfaces.

One of the tasks was scrolling through a list. Participants had to pick their preferred genre of music, then tap it. It was a surprise to see that everyone scrolled their selection to more or less the middle of the screen before tapping. Some even saw their selection at the very bottom or top of the screen, then scrolled to bring it closer to the center before tapping.



Positions where people prefer to tap.

Once I noticed this behavior, I found it wasn't just in picker lists or any other narrow contexts, but that users scroll content to the center all the time. In other parts of that research, and since then in usability testing, I have found that it explains a lot of other user behaviors and is very useful to keep in mind during design.

For example, if we're creating a web form, we will often find that people do not fill in the very last field or two, or miss

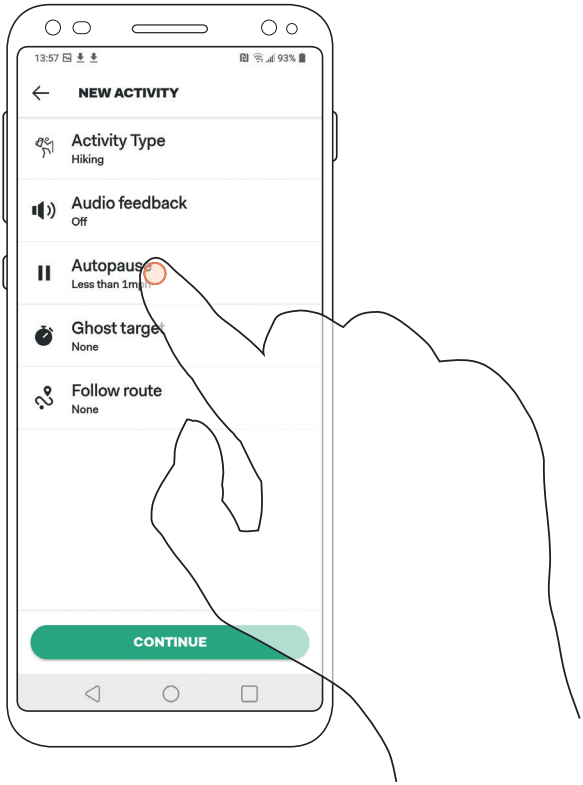
the submit button below the form. It's not a labeling issue, or that users don't want it, but a focus issue. People look at the center of the screen and are less likely to notice or absorb content at the edges.

If we simply *pad* the bottom, adding space below the submit button so the end of the form can be scrolled further towards the center of the screen, then people will start clicking into those fields, even if there are no other changes to the design.

For lists and readable content, users will still try to move the content to the middle of the screen. Since many OSes and browsers allow overscroll (when content scrolls further than its visual limits while a finger forces it, then snaps back), this will simply waste their time, increase their frustration, and make them feel that little bit more dissatisfied. I have often seen users try this repeatedly, then simply give up, not reading the last paragraph of an article, not finishing the form, not selecting the call to action.

Designing for the Center

While the details can be as complex as you wish, the basic design implications here are quite simple. We design so the primary functions and content are in the center of the page. Which is easy, and we're already doing that with the way most apps are designed.



Tapping a control in the center of an application.

A lot of effort is spent bringing attention to the tab bar, edge controls, search, or menus. But these are all secondary functions. What most people do is read and interact with the content. The list of tweets or articles or videos or products is most important.

The advice, then, is to just keep putting content in the middle and controls along the edges. But do this consciously

so we put the proper items in the proper place. Let's make sure we use the touch accuracy guidelines so our interactive items are big enough, based on where they show up onscreen, for users to interact with them. When designing interactions, remember to:

- Respect user choice, and don't assume they will hold the phone one way, or only use our favorite devices.
- Allow users to scroll, and don't design all elements to fit the viewport precisely, snapping back when touch is released. It just reduces readability.
- Plan for input methods. Another very good reason to allow content like forms to scroll to the center of the page is onscreen keyboards and picker lists that block the bottom of the viewport. Ideally, the operating system solves this for us, and allows items to scroll up, but that isn't reliable, and self-scrolling content is disorienting, so users can become lost.

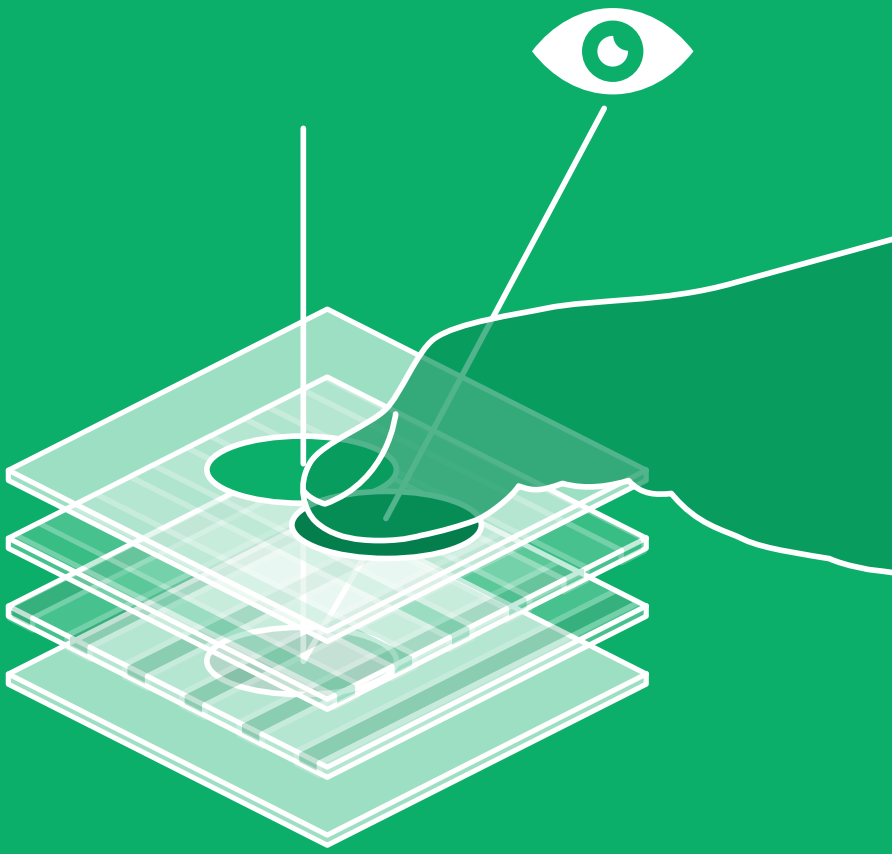
In this chapter we found that people don't read or interact with mobiles top to bottom and left to right, but from the center out. We now know users touch the center more accurately and more quickly than the edges. They prefer to read items in the center of the page and work with

interactive items away from the edges, so they will scroll to make that happen when they can.

With this information, we're starting to understand why successful mobile interactions occur, and how to design our pages and widgets to take advantage of this information.

Checklist

- [x] Place key actions in the middle half to two-thirds of the screen.
- [x] Recognize that everything else is secondary, so it's OK that it's along the edges.
- [x] Allow users to scroll the last content on the page up towards the center. Simply pad the bottom of the screen to allow for this.
- [x] Make sure tap and click areas are sized to meet where they will be on the screen. At least 7 mm for items in the center such as list views, but at least 12 mm along the edges.
- [x] Don't crowd things too much. Most phones can only handle about four buttons or tabs along the edges with that level of touch accuracy.



CHAPTER SEVEN

How Fingers Get In the Way

CHAPTER 7

How Fingers Get In the Way

We've seen that people view and touch the center of the screen, not the edges and not in any so-called thumb-sweep areas. People are always more accurate and faster at touching the center of the screen, and are more likely to consume content and interact with items in the center.

Now it's time to dig down another layer and address a variety of confounding issues to see how people further adjust their behaviors based on how their fingers touch the screen – and how their fingers get in the way.

The Contact Patch

Finger size itself has absolutely nothing to do with touch accuracy in any capacity. Many articles and standards imply or expressly say so, but are simply wrong.[1] This misconception arose from standards built from older technologies and a misunderstanding of the relationship between sensor pitch and detection accuracy, as discussed in chapter 4. We discussed in chapter 3 how capacitive touchscreens work and mentioned the contact patch; we were reminded how people's fingers are squishy, so a large area is flattened

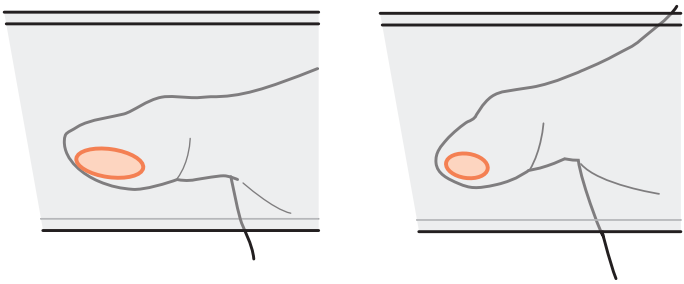
1. https://smashed.by/isoergonomics

against the screen. This entire area is detected by the touchscreen and is called the *contact patch*.

Human pointing accuracy is about people, not technology. Touchscreen accuracy never varies based on the detector technology. But fingers are opaque. They get in the way, and that changes things based on the size and shape of individual people.

Many assume that larger people have bigger contact patches and that children have much smaller ones because of their smaller fingers. But that is just the first of many common misunderstandings about fingers and touch. When capacitive touch emerged, my first purely personal research was an attempt to understand contact patch sizes. I found that almost all adults have broadly the same size contact patch, and children's were not much smaller as they tend to press too hard much of the time.

Adults use more care for critical actions or small targets, and tap with finger (or thumb) tips, leaving a very small contact patch. But this has no real effect on most uses of touchscreen devices. Only a few phones or applications use their multitouch technology to sense contact patch size as a proxy for pressure. In every other case, the centroid is detected and there's no direct effect on the tap or its accuracy.



The contact patch for a typical thumb pad and tip on the screen, as it would be seen from below.

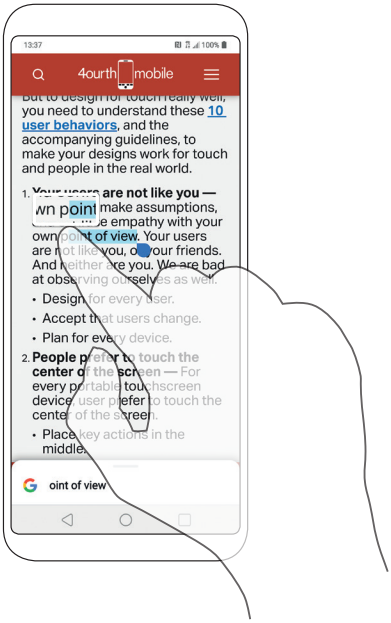
Slowing down can increase touch accuracy a little, just like slowing down is generally a good way to increase precision. No one rapidly threads needles.

However, tapping with the fingertips may reduce the obscured area. It appears to be easier to see around the finger, especially when we recall that people have two eyes. I know that seems obvious, but models such as the drawings above often simplify things as some of these factors are hard to depict.

Try it yourself now. Point at a word on the page here with your finger. Note how much is obscured with one eye open versus two. Especially for devices held closer to the eye, binocular vision helps us see around objects a little more easily since we can see on both sides of them.

THE SURPRISINGLY REAL ACCURACY OF TOUCH

While we're discussing odd aspects of touch accuracy, I always found the limits of touch precision fascinating. There are a few studies that really explore the limits and have found that people can position their fingers with 0.1 mm precision. That's around 0.004 inches, or 1/256th of an inch.[2] Though that may seem impossible, many of us do something close to it pretty regularly on our mobile phones. This research was performed with a digital zoom,



Magnification for precise text selection.

2. https://smashed.by/userprecision

so the user could see their actions magnified. That feedback loop was what allowed participants to demonstrate that level of precision.

When this zoom accuracy research happened, it required a very specialized system, but the various press-and-hold methods to move the cursor on smartphones use the same trick to afford users much greater accuracy than without the magnification. We can all move cursors to the individual pixel level, easily, when zoom is provided. In the rare case a product needs more accuracy than touch can offer, remember that we can add similar features to our websites or app for high-precision selections.

Parallax, Liftoff, and What a Click Really Is

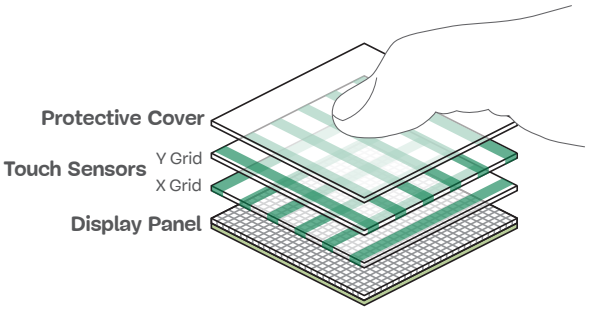
I've used pens for input since the 1990s, when I first encountered one. Wacom tablets are still my primary pointing device on all my computers.

Even back then there were also pressure-sensitive pen devices that integrated with displays; the most common is sold as the Cintiq, also by Wacom. By the early 2000s, Microsoft was making tablet PCs widely available. I used

many of these devices, but I was always troubled by two key issues when using them, and in my attempts to use pens on iPads and other touch devices. It turns out there's good research to understand these two issues: *parallax* and *liftoff*.

PARALLAX

When you see an object change position because you're changing where you see it from, that's parallax. That can be hard to understand, but is the simplest definition I can find. If you bear with me, I'll explain and even have a diagram that might help.

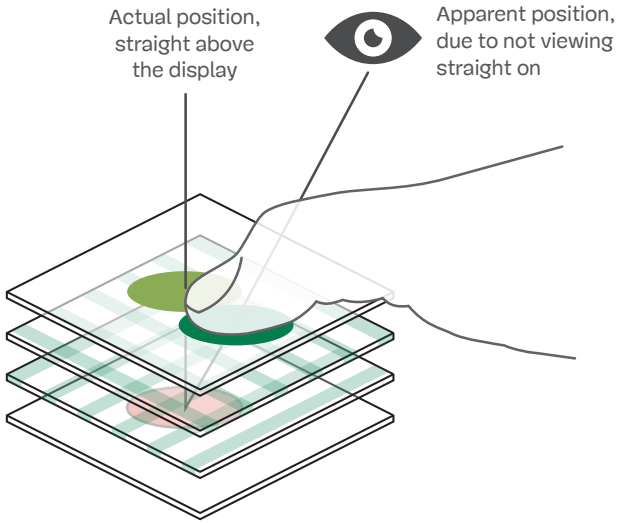


Exaggerated scale diagram of the typical layers of a capacitive touchscreen.

Parallax is a serious issue for optical systems like lenses and scopes, but a version of it arises in all touchscreen devices. The reason is pretty simple. A touchscreen is not a single thing, but a stack of different materials. The touchscreen is a

few transparent layers on top of the display itself. These layers, while very transparent, still have a physical thickness so the touch surface is a measurable distance above the screen.

How parallax causes a problem should start to become obvious. For any particular part of the screen, its corresponding touch area is directly above it. But when the user makes a tap they will instead tap the area that appears in their line of sight to the screen – usually just below the item to be clicked.



Viewing a touchscreen at an angle will cause the touch and view layers to not line up.

In the early days of touchscreen computers there were calibration tools, which were required because the system would get out of sync. When calibrated from the user's normal seating position, it could then compensate for the effect of parallax.

Some devices or OSes contain predictive algorithms to provide parallax offsets, though this is highly secret sauce and it's hard to prove on any specific phone when it occurs. Former Apple engineers have discussed it, so we know it happens there, but not exactly how.[3] Parallax tends to become less of a problem as devices get smaller. Screens and their various layers are much, much thinner now, so the surface is much closer to the display than in years past.

Now let's move from optics to biomechanics.

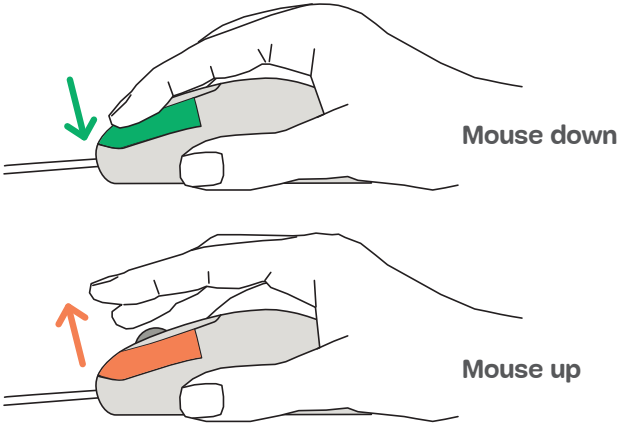
LIFTOFF, CLICKS, AND DRAGS

The action of lifting the finger or pen off a surface – *liftoff* – can cause issues that at first glance seem the same as parallax. Users go to tap or select a target, then miss by a tiny and consistent amount. This is in addition to the target accuracy by position on the screen, and has no direct relation to that position.

3. Ken Kocienda, *Creative Selection: Inside Apple's Design Process During the Golden Age of Steve Jobs*. (2018, Picador)

We need to start with a brief understanding of how computers register a click. Touch interactions are not generally perceived as different from mouse or trackpad input. The default action is a click, which is only activated on mouse up.

When a pointing device other than the finger is used, it moves across the screen and has coordinates assigned all the time; the mouse pointer or cursor indicates this position onscreen. When a button is clicked, that click action is sent to the computer, and onward to the application or web browser as *mouse down*. Usually, nothing happens at that



The physical input for up and down on a traditional desktop mouse.

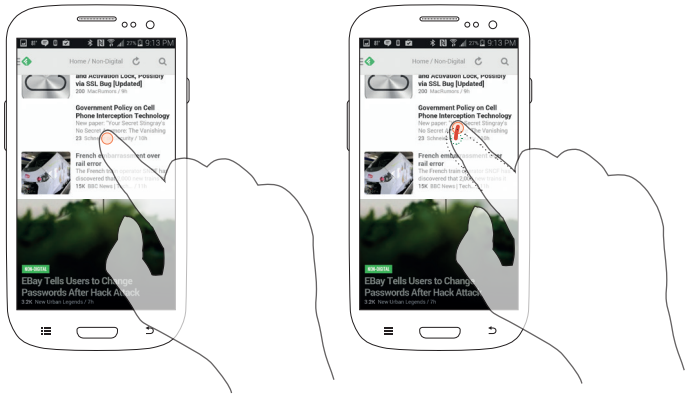
moment. It's detected, but mouse down on a button or link will cause no action. If you've never noticed, go over to your computer and try it. Now, when the button is released, that's *mouse up*. If you let go of the mouse button on a link, then the computer considers that a click.[4]

Moving the pointing device or finger between mouse down and mouse up is a gesture. If the item is available to drag, select, or scroll, the relevant action will happen. Of course, as we know, people are inaccurate, so well-designed systems ignore small displacements between mouse down and mouse up. The device, though, still considers the mouse up position to be the action state. That's important, though not much noticed.

If a user clicks the mouse down on a button and decides they don't want to click it, usually they can just drag to somewhere else and then let go. Nothing happens unless the mouse up is on a different action, or some drag and drop accompanies the selectable item.

This all deliberate; the assumption is that people can fine-tune their selection between mouse down and mouse up. The down-click is coarse and might move the pointer position, but releasing it can be done very smoothly.

4. https://smashed.by/clickevent



Typical liftoff inaccuracy in a touchscreen selection.

Touch uses the same exact controls of down, up, and the two together as click. But the way people touch is quite different from the way they use mice or trackpads. While mice and trackpads have inaccurate down-clicks and accurate up-clicks, for pens and fingers the initial touch is as accurate as we can expect, and the liftoff is less accurate.

This inaccuracy is due to natural but unintentional movement in liftoff, caused by the way our hands' bones and muscles work.[5] The result is that a user lines up properly with the item they want to touch on the down action, but moves slightly (shown in the figure above) as they take the finger off

5. https://smashed.by/displacement

the screen. If two targets are very close together, or the initial touch was near the edge, they could select the wrong target or make no selection at all and be confused as to why nothing happened even though they indeed tapped the screen.[6]

PRACTICAL CONSIDERATIONS

The good part is that for most design work parallax and liftoff inaccuracy are already accounted for. Operating systems provide offsets and for typical interactive items, such as buttons and links, the touch accuracy levels explained in chapter 6 already account for this inaccuracy.

Understanding the issues is important to make sure we all know why there are real-world limits to accuracy, and do not push too far the presumption that people should just try to be more accurate. As long as we all accept that inaccuracy happens and we follow the recommended touch target sizes and spacing, most designs should be fine.

However, if we have to make a high-precision touch-based tool then it is time to start planning to account for the effects of parallax and liftoff. For example, if we include the ability to draw on the screen to select an area, add notes, point out items, and highlight text, it is a good idea to consider other methods, such as snapping to likely selections,

6. https://smashed.by/accuracy

snapping to a grid, a zoom control for increased accuracy, a method to easily correct selections, and some other ways to mitigate the errors that could arise from these issues.

If your system allows drag-and-drop of selectable items, lift-off inaccuracy can cross the drag activation threshold, and taps can result in accidental drags instead of clicks. Consider adding a switch to drag mode instead.

Reading Around Fingers

One of the challenges we encounter with pens or fingers for screen interaction is that users can't actually see what they are clicking. The centroid of the contact patch is obscured by our opaque fingers, so precise work is impossible. I always figured there was no way touchscreens would ever take over the world just because of this problem.

It turns out I was wrong about the impact of touchscreens. But I did have a point about the issues with them. Fingers are opaque, they get in the way, and this is nothing new. Controls of the *machine era* – mechanical or electromechanical pushbuttons, dials, switches, and so on – were also used by people and evolved over time to meet human needs safely and effectively.

To understand how interactive systems evolved – for writing articles and research reports (and this book), and to design actual hardware controls for projects – I reviewed the literature on machine controls and their standards. Many issues of perception, affordance, and usability have been solved by machine-era controls, in one or several ways:

- Controls appear to be interactive and are never hidden or obscured.
- Buttons are large enough to press with fingers or thumbs.
- Labels are adjacent to the button instead of on the button face, so the user can confirm they are pressing the right thing before activation.
- High-density buttons whose only labels have to be on their faces are arranged in regular and standardized grids, such that looking at adjacent buttons can imply the label and function. Keyboards and keypads are the most common manifestation of this.
- Buttons move and make perceptible clicks to indicate when they have activated.
- Buttons change state by toggling illumination or changing color when activated.

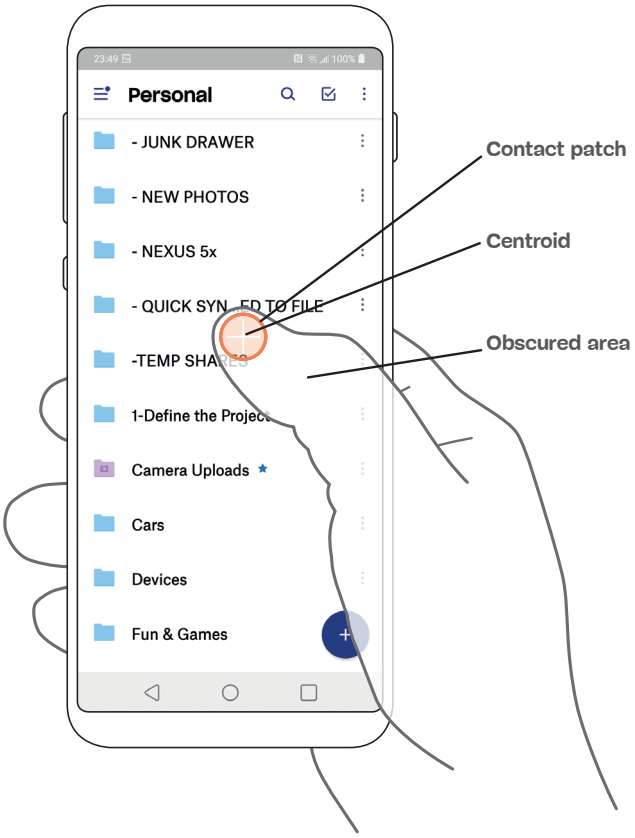
Industrial buttons are clearly labeled, easy to activate, and indicate when they are pressed.

For mobile touchscreens, similar issues exist because they are the interfaces for people interacting with systems. And we know enough about how people work to apply simple solutions for them as well:

- Selectable items like buttons, links, and inputs need to be large enough to press with fingers or thumbs. Refer to "Touch Accuracy by Zone" on page 125, based on their positions onscreen, to make sure items are large enough.

- All items need to be labeled for the user to confirm they are activating the right control. Text labels must accompany all icons, and form inputs must be labeled above or next to the input, not with placeholder hint text.

- Actions must respond using state changes and haptics (vibratory response) whenever technically possible.



As the contact patch is much larger than the centroid, the digit obscures a great deal of the screen under both of these.

- Users can see the visual change while performing the action. Buttons and links should be large enough to be seen around the finger or thumb.

I'll talk more about making controls appear interactive and not hiding them or making them look decorative or display-only in chapter 10.

A few of these issues are well known in interactive and in some cases we readily take the necessary steps, but far too many are ignored. For example, we don't always use labels on icons, and far too many forms label the field inside the input.

We know the best practices and should follow them every single time, not just when convenient to the design style. But there's one issue that is really poorly dealt with. That is the relationship between touch and the obscuring nature of the finger or thumb.

Accuracy is not the only issue in selecting items, so the smallest target that works should not always be our goal. The user must also be able to see the target as they are interacting with it.

While many examples exist, I'll pick on Twitter. Below each item in the feed are quick actions to respond, retweet, like,

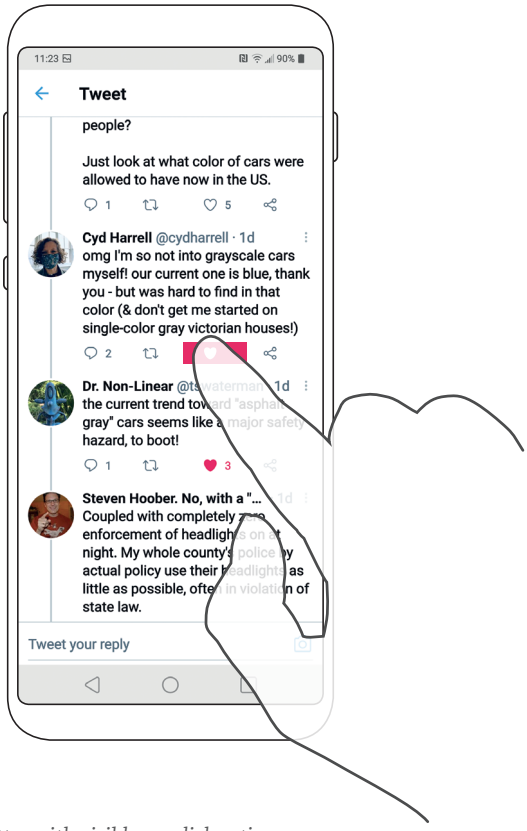
and share. Each is an icon. They also lack labels, but mostly their problem is they are tiny, tiny icons.

Twitter response icons are far smaller than a finger or thumb so are hidden on click.

When tapped, a single icon is entirely obscured by the finger. Though the icons change state, the change is not visible. The user cannot tell if they activated the item – or, indeed, activated the right item – until they remove their finger and perform a new physical and cognitive action to find and identify the action button again, then recall what state it was in, and compare with their memory. All these steps really are what happens. We do it all very quickly, but it adds up and can be disruptive. Any time someone's brain has many tasks to perform, even simple ones, they can become distracted, lost, confused, and make a mistake.

The Twitter icons are (mostly) good in one particular way: most of them change state very clearly. The Like icon, for example, goes not just from gray to red, but also from hollow to filled. Instead of only changing color, which can be hard to see (especially for color-blind people), or changing the shape, which can be hard to identify, this stage change is clear and obvious. Furthermore, the icons are more clickable than they appear, dividing the bar into four sections. Click anywhere nearby – not just the icon – and it is selected. That helps, and we'll discuss this very good principle more in chapter 10.

Imagine, instead, if Twitter made the on-click activation indicator as big as the whole selectable area?



Mock-up of Twitter with visible-on-click action areas.

While invisible, a lot of items we click on every day – most buttons, links, and rows – already have larger active areas than they appear, based on the text label or icon. The Twitter app already does this on Android, and allows the selection of not just the icon but also the whole area. I've mocked up

what the Twitter app could look like if it was coded this way. The entire selectable area could flash red.

It is important to understand that I don't mean the entire design has to change, or imagine that there are large, colorful boxes across the Twitter feed in my ideal unsolicited redesign. Instead, let's take a cue from the default behavior of other interactive systems.

When a link or button is clicked in a web page (applications perform basically the same functions as well), the HTML `:active` pseudo selector causes it to flash for a moment to confirm the action has been performed.

Of course, many designers remove this via CSS in an attempt for "cleaner" design. As the Nielsen Norman Group says, "The idea behind flat design is to simplify the interface. However, stripping away too much undermines this objective by making the interaction more complex."[7]

Now we have discussed the value of this function, I hope we can all remember how to improve the interactivity of design without it impacting our UI choices too much.

We can follow these guidelines for any control on our designs:

- [x] Make sure the tappable area is big enough to be seen around a typical finger size.

7. https://smashed.by/clickableelements

- [x] Make the tappable area indicate it's been activated, with a background color change or in some other visible way.

Even if tapping a function will perform an obvious action, such as loading a dialog or a new page, it's always best to have the link or button indicate that instantly. Always assume there are delays in the action itself being performed.

When building an area to be viewable around fingers, how big should that area be? Fingers are inconveniently big compared to phone screens. The average width of an adult index finger is between 17 and 20 mm. Even an average five-year-old has fingers around the size of the largest recommended touch target size, 12 mm.[8] Those corner sizes seem enormous to people who first hear about them and try to apply them in their designs, so how much bigger can we go?

The answer is actually pretty simple, although it took me a few years to get there. We only need to go bigger in one dimension, as shown in the Twitter mock-up. Use wider elements, just like we already do in many cases. Buttons with text labels, links in text, and form inputs are mostly wider than they are tall. If the selectable area is tall enough to meet the touch target size, they will often be wide enough to stick out past the selecting finger or thumb simply by the nature of their content's length. It actually works very well to design many mobile interfaces in list views, with selections comprising entire rows and the full width of the page.

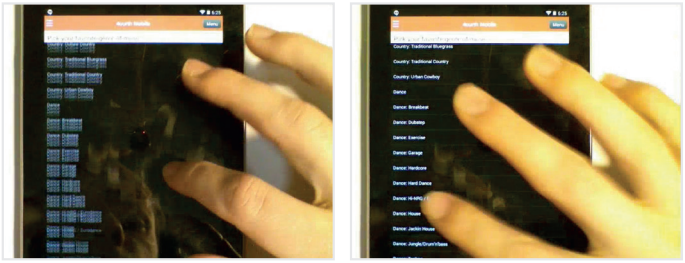
8. https://smashed.by/genderkids

Whether it is a full page list or a picker, whole-row indicators can help alleviate these selection issues with small items such as radio buttons and checkboxes.

Gesture and Scroll

Part of the research I described in chapter 6 was about scrolling. I asked users to perform a lot of different actions to see if they changed the way they held and touched the screen. Much like the initially confounding results of the research on touch, I was also confused by what I observed with scrolling gestures.

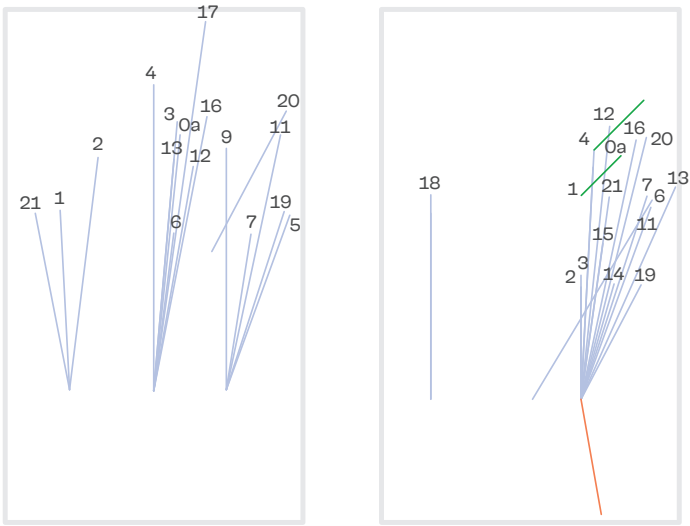
I set aside analyzing gestures until I had figured out the findings on touch. Once that was done, I thought I'd be OK. But it was still odd. The results were not as consistent, and most were clustered well to the right to one degree or another.



Two frames from the test video of a participant scrolling far to the side of a tablet, then moving across the screen to tap the label for the content itself.

As expected, almost everyone made selections on the lists by tapping the first few words of the label text, as they do everywhere when a selection is made. But they had to move over to the left to do this, as scrolling happened in an entirely different part of the screen, often very far to the right side.

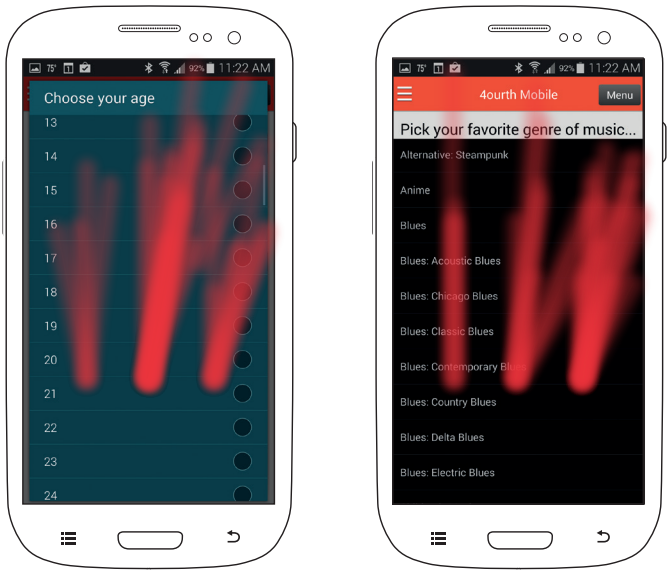
During analysis of the results, this non-centered cluster confused me. I expected to find the gesture in the center, just like clicks were preferred in the center. It turned out that this was not evidence of a mistake in touch results, but that I just needed to look at the context. The cluster-



Some of the vertical scrolling gesture data being analyzed. Start positions are slightly normalized in this view.

ing location correlated to what type of content was on the screen. When people selected, they almost always tried to tap the content itself, clicking on a word or icon.

For scrolling, the opposite result emerged from the research. Users are trying to avoid the content, so they perform their scrolling gestures away from the content-filled area. Overlaying the gesture areas onto the actual screens used revealed this pattern of use. The images below show the ways users scroll, depending on the onscreen content.



Representation of where people were observed gesturing when scrolling.

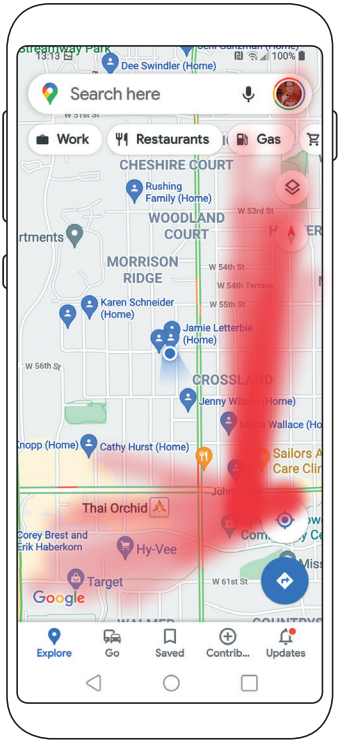
The image to the left is a scrolling selection list with very short content (a list of numbers with radio selectors to the right), showing the majority of scroll actions in the middle, since it is empty and people prefer the middle. The image on the right shows longer content intruding to the center of the screen but without any right side selectors, so the gestures are more often to the far right of the screen.

Observing these images, it is easy to see there are a few gestures clustered in other areas. Analyzing those in context, they can also be explained in the same way.

Take the farthest left gesture on the right image, for example. While it overlaps the content, it doesn't overlap *all* the content. I checked the videos again and those users began their gesture – placed the finger on screen – on top of very short lines of content such as the "Anime" and "Blues" selections shown, so were still avoiding the content at the moment they started their specific action.

Even users scrolling with their left hand were equally inclined to avoid touching the content and could be observed reaching all the way across the screen to assure they only gestured in blank areas. Users of tablets performed the same actions, even though they often reached very far across the screen to avoid the content.

The scroll images above are normalized to the screen size example shown, but the same behavior was observed in all phone sizes and even on tablets. Observations of horizontal scroll were the same. If content is not in the way, people will try to scroll the same place they tap, generally in the middle of the screen.



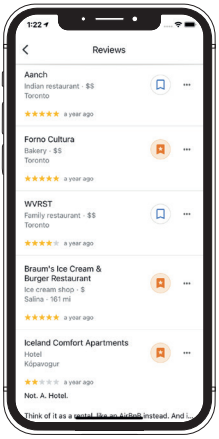
A combination of several charts showing the most common areas in which people scroll vertically and horizontally.

If there is content where people intend to scroll, they tend to avoid it and move towards the edges. If the whole screen is content, such as a map as in the example above, people will scroll to the side to stay as far away as they reasonably can from the content they are trying to consume.

Most pages have scrolling content, so remember to design to allow users to scroll.

- ☑ Use left-aligned text to leave natural irregular empty space on the right.

- ☑ Try to leave room on the right, or space between columns in tables or lists to the right of center. Users can scan the content but feel comfortable scrolling in a non-content area.



A list with multiple columns of content that leaves space between columns to the right of center.

One last note about designing to support gestures: avoid overly gridded designs where every part of the screen is used. Specifically avoid using *justified* text, where both the left and right margins are straight. It creates problems of readability and scannability anyway,[9] but also leaves no ragged right space for users to comfortably scroll.

TWO-AXIS SCROLLING

It is also interesting to look at the angles people scroll. If there's anything like a thumb-sweep range, this is it, but it is true even for people who use a finger off the screen.

People don't scroll in straight lines, and we must map their intention (straight scrolling through lists) to this gesture.

For most designs, this is unimportant. People scroll vertically very well, so we mostly restrict scrolling to the vertical; all sideways actions like gestures are ignored once a vertical scroll begins. Horizontal scrolling is used in some places but generally has poor usability and poor acceptance – it should be avoided whenever possible.[10]

Two-axis scrolling, where users can scroll both horizontally and vertically at the same time, is generally a very poor way to allow users to navigate data such as multicolumn text, charts, and tables.[11]

9. https://smashed.by/justification
10. https://smashed.by/scrolling
11. https://smashed.by/scrollingbehavior

When designing large, two-axis scrolling tools, such as a map or image detail viewer, consider the foundational information about how people scroll. Since gestures are not in straight lines, users will become lost, so try to make sure users can stay oriented. Provide an inset view or a grid to help them. Fallback methods allowing them to return to home with a single tap are also valuable and familiar, as maps always make it available.

Scrolling makes some assumptions about how people read and scan. Most of my recommendations and research are in English or with other Western languages and users. But there are other ways to read.

RIGHT-TO-LEFT LANGUAGES

Arabic and Hebrew are both right-to-left languages. That means a lot of graphic design standards and conventions are entirely the opposite of what most Western designers learn. Default alignment is right, with ragged left.

Once I began sharing my research findings on gesture, designers would respond saying they had seen the same result in their work, and couldn't previously explain it, because they had been seeing exactly the opposite.

Right-to-left languages have the extra empty space to the left, so scroll avoidance is observed to the left of the con-

tent instead of the right. Aside from being very good advice to keep in mind for our regionalized, multilingual designs, this finding helps prove that the research on touch and scroll is universally true.

In this chapter we found out more about how fingers work with screens, and the ways people change how they interact because of that.

People tap with fingertips when they are trying to be accurate, but it doesn't make a lot of difference in accuracy; there's no need to design for this, or instruct users to be careful as they know to do that already.

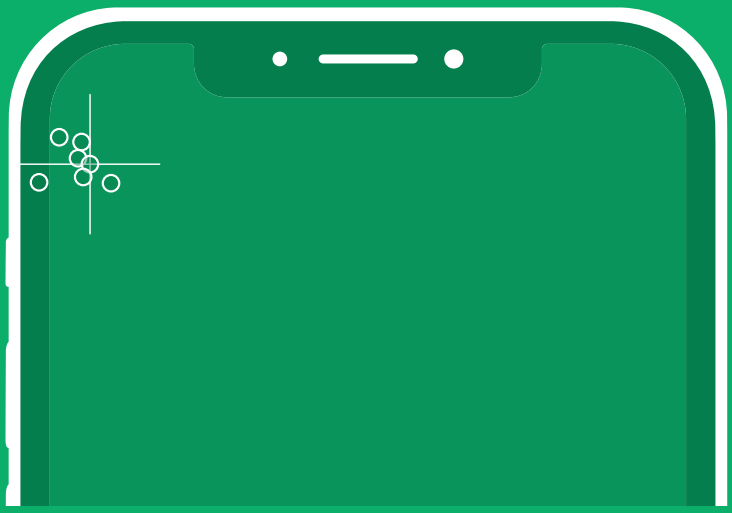
Items big enough to be accurately clicked are often too small to be seen under the finger, and we need to design to counteract the problems this can cause.

And we know people avoid dragging across the screen to scroll on top of content, so will use empty areas, or move far to the right (or left for some languages) to avoid content.

Using the lessons of these behaviors, we know of a few additional tactics we can use to optimize how people interact with our websites or apps.

Checklist

- [x] All actions should respond visually the moment they are tapped or clicked so users are assured they actually hit the target.
- [x] For bonus points add haptic response to the click reactions.
- [x] Make sure visual responses are big enough to be seen around fingers.
- [x] Take advantage of content length to make interactive response items wide, or use entire selectable rows.
- [x] Always use left-aligned text, never justified, to allow for ragged space to the right users can feel safe scrolling across.
- [x] Design multicolumn lists with internal gutters and gaps to encourage scrolling in empty areas, viewing of the whole content.
- [x] Assume users will select near the beginning of label text, so will move from the scroll position, across the screen to select.



CHAPTER EIGHT

Imprecision and Probability

CHAPTER 8

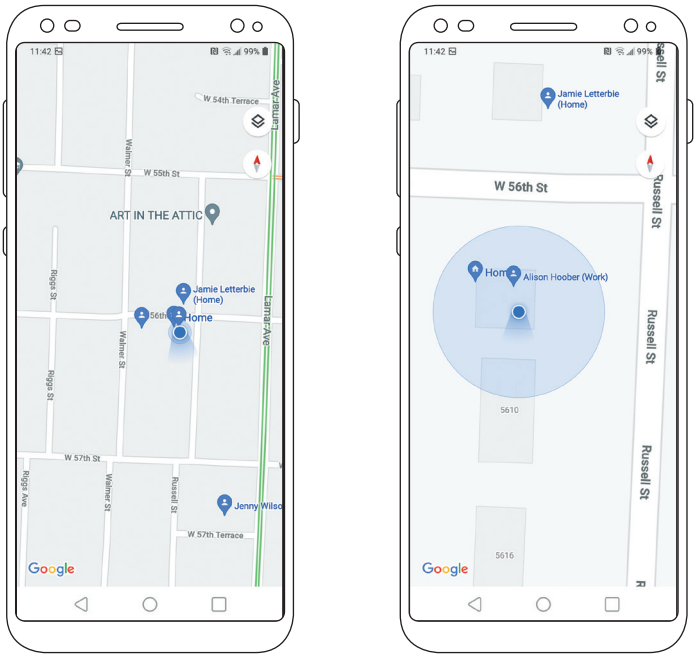
Imprecision and Probability

Now that we are starting to know how to design for different touch accuracy by area on screen, and how to avoid issues with human sensing including parallax, liftoff, and the screen being obscured by fingers, it's time to dive further into why touches happen the way they do. We'll see what really makes up accuracy, and how to design around the natural inaccuracy that occurs with every selection.

Precision and Inaccuracy

Join me in a little experiment. Open up the maps app on your phone. You will see where you are, that little dot or whatever icon the app uses. Now zoom in. You might have to zoom a lot to see it, but sooner or later you'll see that the icon representing your position is not a dot alone, but the center of a bigger ring or translucent circle.

You might even be able to see the little icon for your position wandering or jumping around. That's because the phone doesn't know precisely where you are. It only knows to some degree of probability that there's a good chance you are somewhere inside the circle.



Google Maps, zoomed in far enough to see the accuracy ring indicating that I might be anywhere from the street, to the neighbor's garage.

As it turns out, a lot of the things we consider proven facts are probabilistic approximations. For instance, things like location accuracy use something called the *circular error of probability*.[1] We can think of this as the mathematical representation of inaccuracy. This probability isn't the phone just guessing – it is actually built into systems like GPS, which include an accuracy value as part of their positional calculations.

1. https://smashed.by/circularerror

Location systems by default use a 68% probability circle, referred to as *R68*. That means while there's a 68% chance that your location is inside the big circle, there's also a 32% chance you are somewhere outside it. But how does this relate to touch accuracy?

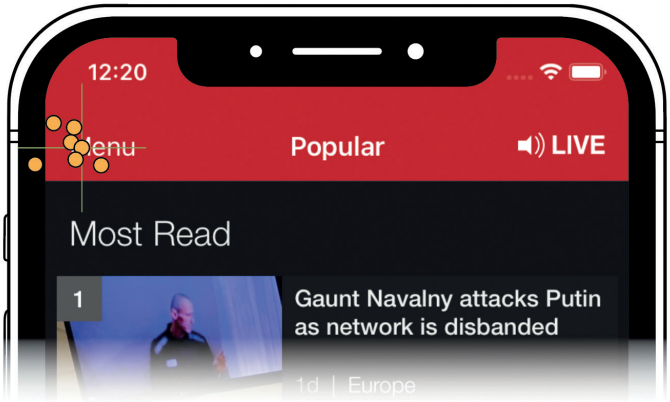
The app GPS Test, showing the more raw Global Navigation Satellite System (GNSS) data, including the accuracy, shown here as 18 meters.

Touch Accuracy and the Circular Error of Probability

For all the touch research results, I selected R95 as my probability radius. The circles shown in diagrams throughout this book all contain 95% of taps. I wanted to have much

higher accuracy than the 68% of location, for example – but why not 100%, or at least 99%?

The figure below shows the actual touch positions for users attempting to select a menu button on a 7-inch tablet. Each dot is a tap recorded, and the circle denotes the probability of 95% of the taps falling within it.



Not all taps fall within the touch zone, or even the tappable area of the screen.

This diagram shows that a 99% probability ring would be a lot, lot bigger. Two or three times bigger, which would be so inconveniently large that we couldn't design usable products at all. I worked this out with some statistical analysis during the research analysis phase, and rounded it down to 95% to make sure the size was reasonable.

Much above 95%, the size started to increase in radius much more rapidly. We couldn't design an app for 99% probability of hitting the target unless almost the whole screen was the button area. Without getting into all the math explaining why, there's simply no such thing as 100% in probability.

What's much more important than worrying about edge cases, and the specifics of why it's 95 and not 96%, is that we understand the concept that inaccuracy is inevitable and embrace the imprecision. What happens when the user makes one of those taps and misses the button?

Embracing Imprecision and Imperfection

The traditional way to design is to address every possible aspect of the system's behavior. A typical way to do that for digital systems is the use case.

A systems analyst looks at the requirements, maybe informed by some early design concepts, then decides if the user clicks *this* button, and selects *those* items, the result is *that*.

The problem is that we allow users to input data, use sensors, and have a giant database on the back end. Most systems are arbitrarily complex, where *arbitrary* has the mathematical meaning of "more or less infinite."

I remember a meeting about adding a feature to a banking application and website. There were a dozen use cases, but as we went through the requirements the team kept finding new needs or gaps in the use case analysis. The analyst kept writing down new ones to create. I realized another was about to come up, which would have had any of several hundred selectable options, each of which interacted with the other matrix of options.

I shared my observations with the project team and quickly did the math on the whiteboard. I showed that we couldn't perform use case analysis on this, because there were millions of permutations and it would have added approximately 100,000 years to the project schedule. Unfortunately, most product design teams have "solved" this by ignoring most use cases, only designing and specifying "the happy path" – one error-free flow through the system – and assuming the user follows it.

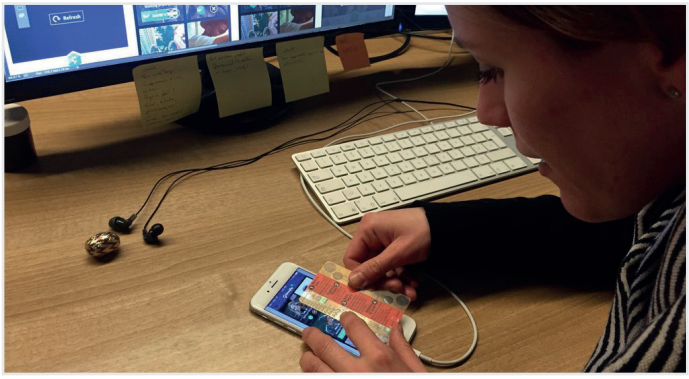
Designing only for "the happy path" and considering all others as edge cases is a dangerous method of design for our products.[2] Although architecture and strategy are beyond the scope of this book, the same mindset pervades the UI design layer as well. There's an implicit assumption that everything happens deliberately, and that users make no mistakes. Throughout this book we have seen that people make mistakes all the time. For information design and user

2. https://smashed.by/happypath

interface design, I like to use simple analytical techniques to avoid the problems discussed here.

Designing for Imprecision

The basic method to analyze for the impact of touch inaccuracy is not dissimilar to kinematic analysis methods, as used in the design of machinery.[3] For touchscreen design, the first step is simply figuring out what is likely to be touched and what it may interfere with. We can then work across the design, one target at a time, and see what problems might emerge.

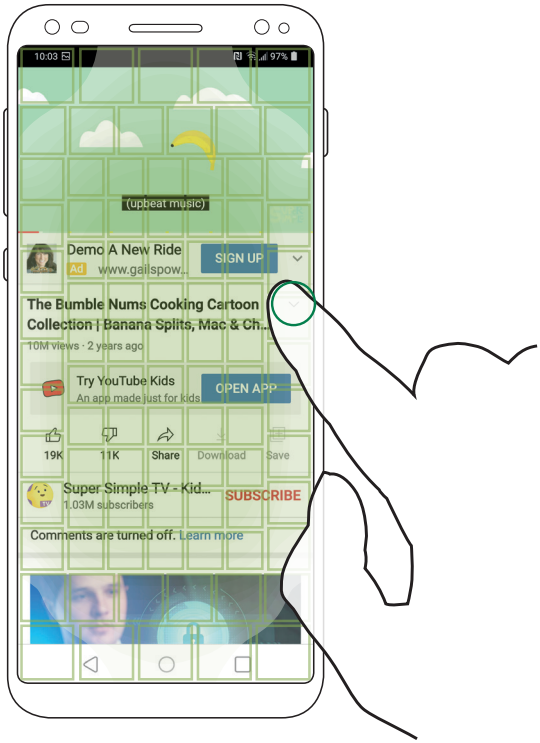


Inspecting a design for touch inaccuracy with a Touch Template.

To inspect a design, place a touch accuracy circle,[4] sized for the screen position, on top of the element you are inspect-

3. https://smashed.by/mechanisms
4. See chapter 13, "Practical Mobile Touchscreen Design," for an example Touch Template.

ing. Though I turn the layers off and on when doing this digitally, I often leave the grids and center-out accuracy areas (see chapter 6) visible to remind myself when to change sizes, or where to better move items when redesigning.



Inspecting a touchscreen application using kinematic methods.

I sometimes even show it with a finger or a cradled thumb scaled to the right size to remind myself of how the hand

will obscure items onscreen, meaning labels or changes elsewhere might not be seen immediately.

The example at left shows a finger on the YouTube app. We can see that the down arrow icon only appears to be spaced a good distance from the other icons. Measuring the distance reveals it is actually a bit tight. If the touch areas are as large as they should be to ensure they are activated – not just the icons alone – then there's a good chance users will hit the new video or account icons instead. Users are never going to be able to precisely click targets and there's limited screen space, so correcting items like this can be difficult. But we can still account for it in design.

AVOIDING TOUCH PROBLEMS THROUGH DESIGN

There are three basic tactics that I use in my product design work to alleviate touch inaccuracy issues.

Larger Tap Areas

The first thing that has to happen to make a functional touch-screen app is to allow people to touch things on the screen. It is surprising how often this is an issue. If a user taps and nothing happens, it's not their fault but the designers' or – more often – the developers'. The default seems to be to make words and icons clickable and only the words and icons. Even inside buttons, the big box is all too often not clickable, just the word or icon inside it.

Instead, we need to make sure that whole areas are clickable. Not just for buttons, but for everything – make whole areas clickable. Very often this is natural, as designs are in blocks and boxes. Find and define these box areas and make them the tappable zone.

For its iOS-native features like the back button, at least, Apple calls buttons like this "charged" and uses a somewhat

Resilience Design

In 1956, computing pioneer John von Neumann called computational errors "an essential part of the process" of computing (smashed.by/problogistics). This was almost forgotten for decades with the advent of the highly reliable computers we work on today (smashed.by/inexactdesign).

At the scale data centers operate, errors are simply impossible to avoid. A typical data center has hundreds of hard drives fail every day, not to mention server crashes, network collisions, and more. Systems administration at the scale of services like Google, Facebook, Etsy, Flickr, Yahoo!, and Amazon employs teams of *resilience engineers*, who make sure their data centers stay running by planning for graceful failure. When failures occur, users (usually) don't even notice (smashed.by/resilienceeng).

cleverer way of detecting misses to make sure clicks work, much as you may have noticed in keyboard prediction algorithms for touchscreens.[5]

Just expanding the touch areas to the outer bounding row or block is much easier and doesn't have to impact the look of the app at all. The area remains blank, leaving us all the white space needed without changing the size of the icons,

Resilience in this sense is the ability of a system to absorb disruptions without tipping over into a new kind of order. A building, when exposed to too much lateral ground movement, settles into a state of rubble on the ground unless it is designed to resist this disruption adequately.

I believe there is a concordant concept of *resilience design*, which takes into consideration the same assumptions about the arbitrary complexity of humanity, life, and the many systems people work with to perform their tasks (smashed.by/imperfection).

Let's make sure our designs are resilient because users will never, ever do what we expect them to do. Even aside from system failures and errors, people follow their own processes in ways we simply cannot predict.

5. Ken Kocienda, Creative Selection: Inside Apple's Design Process During the Golden Age of Steve Jobs. (2018, Picador)

buttons, or labels. Always remember those touch accuracy sizes. If we give someone a 2 mm icon to tap, they are going to struggle to hit it, or might give up and may think the website or app is broken and not buy from or use it. Default web and app behavior actually follows, or can easily be made to use, these large selectable area methods. Real buttons are, of course, all clickable. As long as we do not use odd UI workarounds, everything works pretty well.

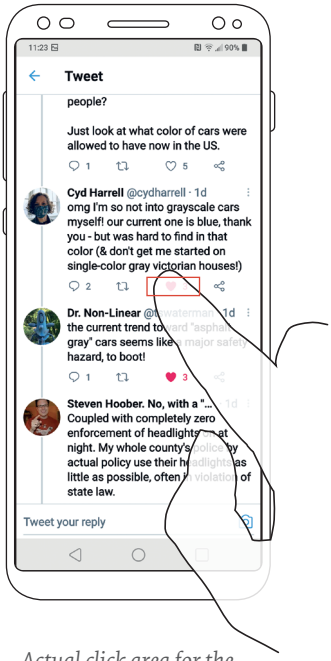
If the human performance and accessibility factors[6] were not enough of an argument, maybe ease of implementation will be enough to get our product design and development teams to stop making everything from scratch and just style default elements so their behaviors are good and consistent.[7]

Let's revisit Twitter again – the app and website are very similar – and look at the same clickable item we noted in chapter 7. It's a very good way to discuss these topics, with both bad and good examples of touch interaction and interference at the same time.

In the example opposite, Twitter has coded the three icons below each tweet, including the retweet button, to be not just the size of the icon but to occupy the whole space between each button. Well, on the app they did; on the web, there's a smaller square around each icon, and they're rather too small to be a safe touch target. But at least the app works right!

6. https://smashed.by/htmla11y
7. https://smashed.by/semanticmarkup

Tapping the Like button on the Twitter mobile app.

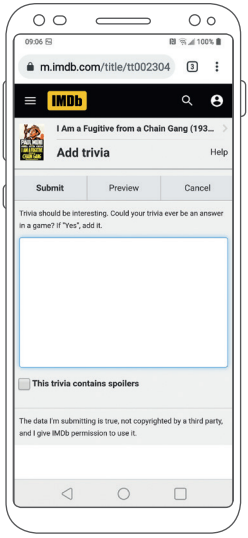


Actual click area for the Twitter Like function.

Space between Tap Areas

Long ago, when I was first coming up with sizes for touch targets, I encountered issues around the edges. I learned later it was literally the edges of the screen, but I didn't recognize that at first. My early research and borrowing from other guidelines led me to simply add spaces between items. A gap of a few millimeters in addition to the actual target size.

The extra space was a workaround, and as I learned more about how touch worked I simply adjusted the sizes per area on the screen. For non-critical items, use all the space we can and simply let the touch areas bump against each other; properly sized items present no undue risk. But for critical items? When the consequences of a wrong click are permanent or destructive, do we still need space between items? Yes, we should space things out. For some reason, a lot of forms still cluster submit and cancel buttons right against each other.



This form on the IMDb mobile website has three buttons touching each other.

Not all are as bad as the IMDb example, with buttons literally touching, but accidentally tapping cancel is still pretty likely – and quite destructive.

While there's no need for every click item to be surrounded by dead space, dangerous items should be placed far away from commonly used items. How far away? As far as we can. Ideally, functions like send, undo, delete, and cancel should be far from everything, but usually they are fine if clustered into a low-use area or menu.

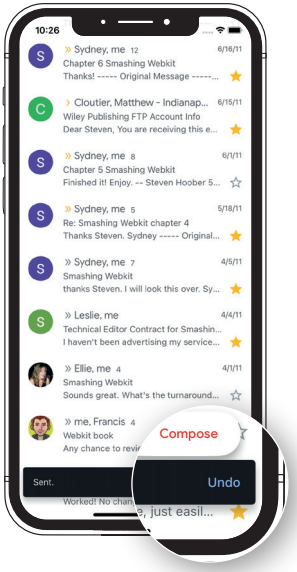
Users will become familiar with common actions and perform them faster and with less attention. Faster actions are less accurate, and unattended actions may not be monitored enough to avoid danger, so placement is important.

Avoid Catastrophes

Users will sometimes miss and hit the wrong target. What happens then? What if they need to add an attachment to the email, but press send by accident because the buttons are far too close together?

We let the user stop the catastrophe. Gmail has an optional undo function in its desktop-oriented web service, but it is always available on the mobile side. Google acknowledges that mistakes occur and lets users fix them.

Mistakes shouldn't be unrecoverable. *Guard conditions* – "Are you sure?" dialogs – are not super effective and are quite intrusive, but things like undo are very useful. Aside from not adding clicks when users perform the action specified,



The undo send function for the Gmail app.

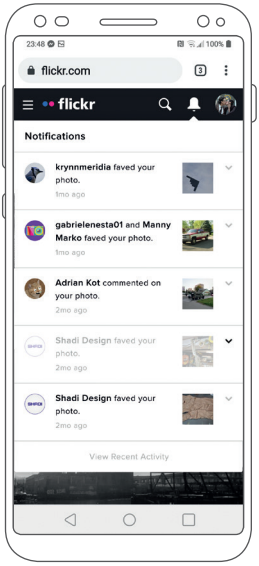
they also counter the panic of the user: "Oh no! Make it stop!" And there's the Make it stop button right there.[8]

Another option is to make accidental clicks less inconvenient. On Twitter, for example, the buttons below the tweet are far, far too close to the tweet text for safety. People will accidentally hit the text and that loads the tweet as a full page. But the same functions are on the tweet page, so the user can just tap again to retweet from here, and then press back to return to the feed. No real problem.

The best way to avoid catastrophe is to not allow it at all. Most forms do not need a cancel button (people can navi-

8. https://smashed.by/surevsundo

gate away by pressing back instead), and no form I have ever worked on in my entire career needs a clear button.



Notifications in Flickr that are turned off become grayed out so they can be tapped again and re-enabled easily.

Simply removing dangerous actions is often the easiest way to solve this.

We've discovered by now that touch accuracy is governed by probability, and there's nothing we can do to make sure all touches hit the target we want or are as the user intended.

We know that errors occur and need to create systems to accommodate them by designing interfaces using the best data we have, and designing interactions and processes to avoid catastrophe if mistakes are made.

Sometimes the errors people make on touchscreen devices are caused or exacerbated by factors outside their – and our – control. In the next chapter we'll find out that people and the conditions under which they use our websites and apps on touchscreen devices are often not what we hope for or expect.

Checklist

- [x] Make touch targets as large as possible; design using whole rows of lists, and entire containers.
- [x] Never make the interactive target just an icon, or a word.
- [x] Space out dangerous items so they are not near positive or commonly used items; don't put Delete right next to Save.
- [x] Design processes to avoid destructive actions entirely, or provide undo to allow the user to back out of any mistakes they may make.

CHAPTER NINE

Phones Are Not Flat

CHAPTER 9

Phones Are Not Flat

We've already moved from technology to human factors research: how people adjust their behaviors based on how they touch the screen. In this chapter, we're going to talk a little bit more about how people vary, and how their particular needs and capabilities, and their environments, influence how they interact with the websites and apps we design for them.

Empathy, Temporary Disability, and Universal Design

In article comments, or questions after speaking, one of the things I get challenged on the most is accessibility. I often present touchscreen and other design tips without a special section on universal design or accessibility. This is deliberate.

I don't specifically address accessibility because of the concept of temporary disability.[1] While around 15% of the world's population – about one billion people – suffer from some sort of identifiable long-term disability, almost anyone can suffer a short-term injury, be blinded by glare, not be able to hear over loud traffic or machinery, have a hand occupied controlling a small child, or any number of things.

1. https://smashed.by/christopherson

Desktop computing often ignores these factors and assumes all our systems are set up well, in good environments that allow us to focus on the work. Others who did important research into human factors and interaction with machines, like the US Air Force, explicitly ignored those with color-deficient vision (color blindness), so many standards have developed that are skewed to exclusion.

We all know, however, that mobile devices are for everyone, and in every environment we can imagine. By establishing these accessible use cases as the default, we can ensure our products work for every user, all the time.

- We can't rely on sound cues or audio, as people often use their mobile devices in loud environments or, indeed, quiet ones where the phone should not make noise.[2] Therefore, always make sure there are onscreen notifications and captions for video. Now we're designing for people with hearing impairments.

- We can't rely on color to signal information because glare, odd viewing angles, or sunglasses can wash out colors or change how colors are perceived. Design with enough contrast and for words and shapes to convey meaning. Now we're designing for those with color-deficient vision.[3]

2. https://smashed.by/noise
3. https://smashed.by/dark

- We can't assume that a mobile device has the full attention of the user at all times, so we should never set timers for critical functions, and avoid session timeouts and messages that disappear.[4] Now we're designing to accommodate any number of users whose interaction is slower, from low-vision or blind users with screen readers to people with cognitive disorders.

- We can't assume mobile devices are used in stationary settings. There may be vibration or other movement, or the user may glance away, so we need to design type sizes large enough and at high enough contrast to work when walking or riding in a vehicle. Now we're designing for people with vision impairments such as cataracts, or with tremors who cannot hold the phone steady.[5]

We should design products that address all users' needs and behaviors. We have to embrace the complexity and design to accommodate the messy lives our many different users live.

As mentioned in chapter 5, people shift the way they hold and touch their devices depending on the device, the input necessary, the position on the screen they are trying to tap, and their context.

4. https://smashed.by/expiry
5. https://smashed.by/tremor

Users interact with their devices in a wide range of situations that are often filled with distractions.

I have observed people changing the way they hold their phone when in these contexts, among others:

- opening a door
- carrying a shopping bag
- carrying a small child
- walking down the street
- walking in difficult terrain, up or down stairs, or stepping off a curb
- riding an escalator, train, or bus
- when there is danger to themselves or the phone, such as when walking in strong wind, near water, or near a drop-off

In chapter 7 I described new data with you to confuse the simple touch accuracy levels. I'm going to go over the research again; but this time introduce a few things people encounter in the real world for which we have data about how it impacts touch accuracy and our assumptions of how to design for touch or gesture.

Inadvertent Movement

The issue of accidental taps mentioned in the previous chapter is even more important than you might think as mobiles move around. Users make accidental clicks and miss their intended targets all the time as they grab the phone to pick it up, to grasp it better, or simply when jostled by other people or moving around themselves.

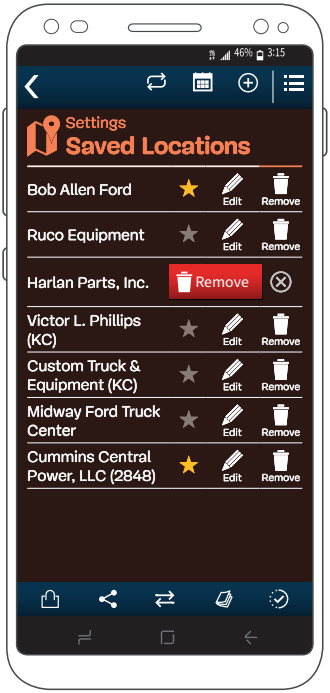
Among the most common of these mistakes is accidental multiple taps, especially while users are walking, riding in vehicles, or just in busy areas where they are nudged and bumped against. And, of course, as we age, many people develop tremors or inadvertent repeated muscular shaking.

Unintentional double-clicks even happen on desktop computers with mice and trackpads. So much so, software (like ClickFix and others) is available to prevent it, and macOS comes with a setting to avoid it as well.

Other accidental tapping motions include drags when the user meant to tap, and drops during drags, due to the same reasons of external movement or vibration, or user tremor.

DESIGNING FOR INADVERTENT MOVEMENT

The principles of designing to avoid catastrophe as described in chapter 8 are the most critical here. Make sure that catastrophes can be avoided by designing systems to not allow them, or to undo actions.



To prevent accidental deletion due to double-clicks, the delete confirmation is not superimposed on the initial action.

One obvious way to avoid drag errors is to have fewer drag functions. Drag-and-drop to reposition or customize interfaces is arguably overused, and accidental activation is one reason. Consider requiring users to enter a special mode for customization.

Double-clicks can often be avoided by designing systems that ignore them. Buttons, links, and controls should only accept a single press and not cause a double submission. Combined play/pause controls are a good way to save space and make media playback simple-looking, but are a common case of frustration due to double-clicks.

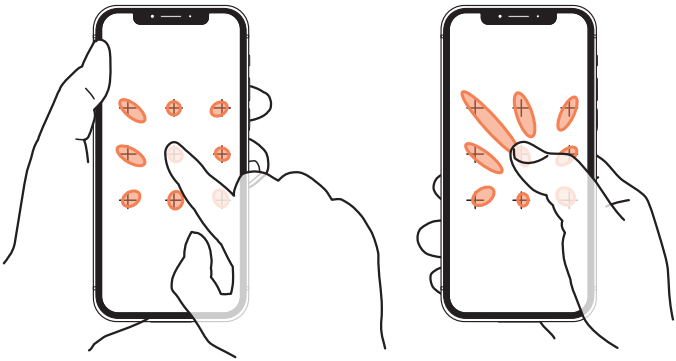
Inline guards are a different risk. If the user pressed a delete button and then the "Are you sure?" button appears in the same place, accidental double-clicks can cause deletion. Consider offsetting all controls so users must deliberately activate each one.

Hand Availability and Other Contexts

One of the key takeaways from this book is that people don't use mobile devices in any one way, but change and adapt to their preferences, needs, and contexts. But in some cases they can suffer consequences from this adaptation.

The standard – though flawed – assumption is that mobile phones are used one-handed with a thumb. There are lots of other suppositions that follow this, chiefly that one-handed use is the ideal case of mobility being on the go. People can perform other tasks as a result.

The reality is harsh and unforgiving. Even when using the phone one-handed, doing stuff with the other hand affects touch accuracy.



Accuracy of nine targets when holding the phone in two different ways and carrying a bag.

Researchers have explored the effect of trying to carry other objects while walking and using the phone. Just carrying a shopping bag in the other hand can reduce touch accuracy in the most distant corner of the screen by over 30 mm.[6]

6. https://smashed.by/encumbrance

Remember, the research I outlined in chapter 6 says that this area has an accuracy of around 10–14 mm. People simply walking down the street will sometimes hit not the row next to their selection but two or three items up. People missing targets by a factor of three has not even occurred to us before this.

DESIGNING FOR UNUSUAL CONTEXTS

Again, the principles of designing to avoid catastrophe as described in chapter 8 are our best guide here. And now it's easy to see why that is so important. Taps can accidentally be far from where the user intended because of huge inaccuracies due to unusual contexts of use. We should try to make no assumptions about what users intend to do and when they might notice a mistake; we should prevent accidents and allow users to back out of or undo changes.

It is hard to suggest anything further for controls intended to be used while in difficult contexts. While large mechanical controls such as push buttons with locator elements like raised edges have solved this problem well, onscreen there are additional problems with little research and few guidelines.

I have observed several times how oversized onscreen controls can easily stop looking like controls at all. Very large

buttons or other large selectable areas may be less usable. But I have no detailed research on what counts as "oversized."

Edges could provide some physical guard, locator, and guide, but they present their own issues.

The OS Steals the Edges

Mobile operating systems used to be predictable and granted the application a nice, reliable area to work within. The viewport – the space available to display the application or website – was more or less the whole screen. When it varied, such as when browser chrome disappeared on scroll, or a full screen video hid all controls, it was predictable and easy for users to bring back. Over time, controls like the Android navigation bar have moved from the hardware to the screen; iOS did much the same. As a result, gestures are not ours for the taking anymore, as all edge gestures are stolen by the OS to do one thing or another.

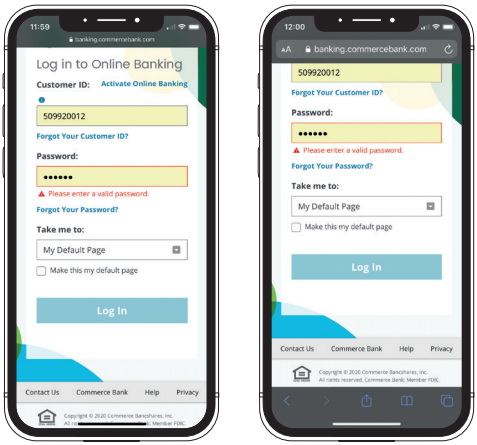
Taking all these factors together, our viewport is much more variable; edges can be commandeered by the OS at any time, and sometimes the controls we place near the edges can move, be overlaid, or become inactive when we'd otherwise have every reason to expect them to work. Since most designers seem to gravitate to iPhones, one I hear about a lot now is the mobile Safari toolbar.[7] The browser Chrome mini-

7. https://smashed.by/safarimenu

mizes like it does for all browsers as users start to scroll. But instead of returning only with a scroll back up, the bottom area is never really part of the web page.

If the user taps in the area where the toolbar displays, no clicks are registered on the web page at all. The browser bar will appear, and the page will scroll to put the tapped content above the toolbar.

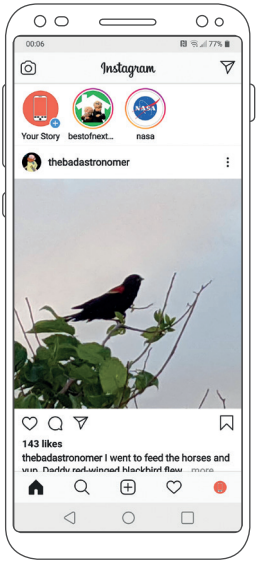
In the example on the left in the picture below, I tried to tap "Help" and got… nothing that I asked for – the right-hand example appeared. To actually get help, I'd have to click it again in the new place it has been scrolled up to.



Two views of a web page in mobile Safari, one before and one after tapping along the bottom.

Behavior when an expected action is not performed can be confusing for users, especially if the website were to place controls there rather than links. Users notice when the page doesn't change, but might tap to make something remote happen and not notice for some time that the requested action did not occur.

As discussed in chapter 6, people view the center best. For most web pages, the suggestion would be to pad the bottom of the content, with even footers raised towards the middle of the page to avoid this problem.



The Android navigation bar, at the very bottom of the screen, below the app navigation in Instagram.

The basic concept of control bars taking over the page edges isn't unique to iPhones. Android browsers have various sorts of hidden bars, some with iOS-like behaviors where lower gestures cause unexpected results as well. In fact, the whole Android OS, on every screen, has a bar of controls at the bottom, the navigation bar, with at least home, back, and the app switcher, though others can be added to it.[8]

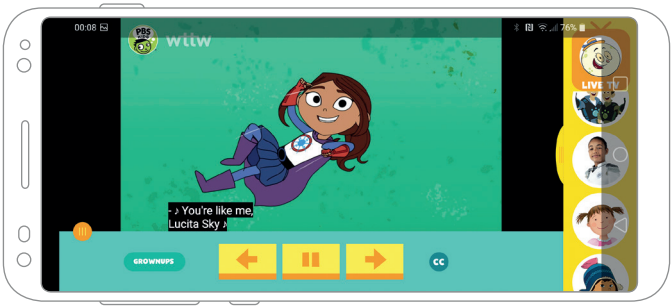
Except of course, when it doesn't. Any app can choose to dim or entirely hide the navigation bar, and the user must make a gesture to bring it back onscreen.[9] This is most often seen in legitimate full-screen tools, like video playback or games. Usually a tap or gesture up from the bottom in this area makes the bar reappear. But maybe not, if it was oddly implemented or the device maker has added some extra control.

Far too many full-screen apps forget to design for the full range of cases, so build their controls, closed captions, or other items in the navigation bar area. Depending on how it was coded, the navigation bar and controls could then either be hidden as they overlap each other, or jump around and shrink when the application frame reappears.

The PBS Kids app serves as a good example of how not to design for Android navigation bars. The app is always set to full-screen mode, so it hides the navigation bar and the

8. https://smashed.by/androidnav
9. https://smashed.by/hidingnav

status bar. If the proper gestures are precisely used, they can appear, but they are translucent, overlaying other UI elements, which makes them hard to read, and for so little time that it is hard to then orient and tap the proper control to do things like exit the app.



The Android navigation bar visible on top of the fullscreen PBS Kids app.

Since the app has two modes (full-screen viewing and a controller version as shown in the screenshot), a better solution would have been to hide the navigation bar (and the status bar) when in full-screen. When in the controller mode, both of those bars should be permanently visible and not overlaid on the app itself.

How else should we design to avoid conflicts with edges and fixed OS elements?

ALLOWING FOR OS-CONTROLLED EDGES

There is no way to mitigate most of the issues caused by edge-based OS controls or function bars. The best choice is to entirely avoid conflict with these areas. Some ideas or suggestions include:

Leave Space

Never cheat and try to squeeze in more content, either vertically or horizontally. Use the minimum tap sizes as shown in "Touch Accuracy by Zone" on page 125 as the starting point. Make sure no adjacent controls or gestures interfere with the controls. Add space to the margins or move controls somewhere else entirely.

Avoid Chyrons for the Web

Don't build items that will be disrupted every time the user tries to use them. I like chyrons for some controls, notifications, or simple buttons that are always visible. (I call those bars docked to the bottom of the viewport "chyrons." The name is derived from the graphics bar at the bottom of the screen used for scrolling news or other information in TV production. The technology was originally developed by the Chyron Corporation in the 1970s.) But when making anything for the mobile web, a lot of

people will use iPhones with that Safari toolbar behavior. This is likely to get copied into other devices, so expect it to become more widely troublesome soon.

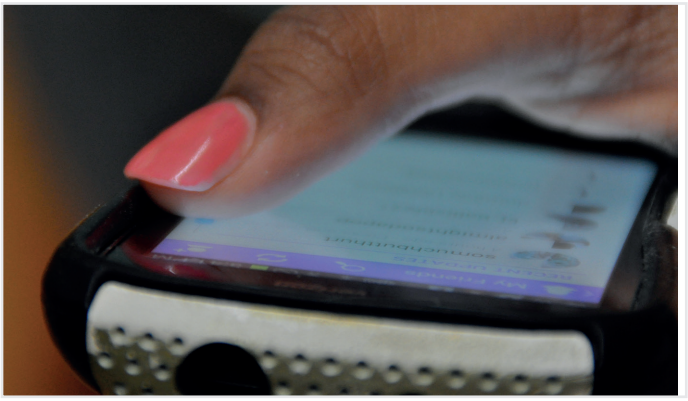
Use Caution with Chyrons in Apps

If your app has a row of frequently used controls right on top of the Android navigation bar, it will cause problems. Users might not just tap the wrong control inside the app, but they could unintentionally use the OS-level controls on the navigation bar. They might even tap the back button, which will cancel processes that the app might be running. The removal of the home button from most iOS hardware may cause similar issues there with accidental gestures at the edge, but I don't have enough data on this yet.

Issues with the edges are not just about what is actually on the screen, however.

Cases and Bezels

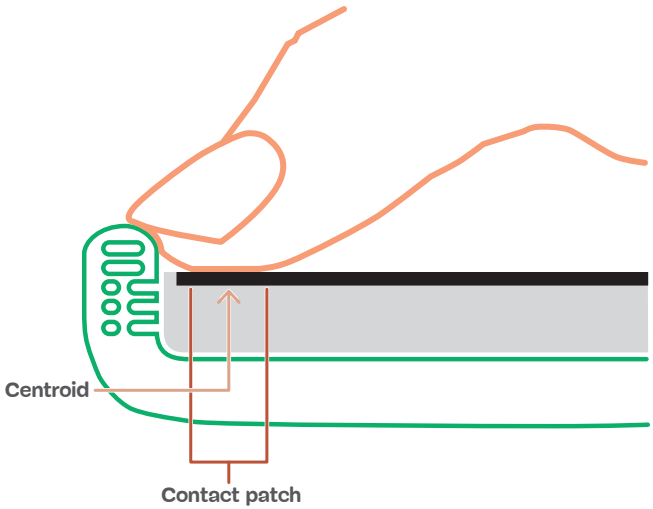
While few phones these days have a raised *bezel* – the edge around the display area of the screen – the vast majority of phones are put inside cases. Without any good data on these, or a means to detect them programmatically, there's no way to tell what sort or how much they interfere with the way the phone works.



Cases on smartphones can interfere with touching the edge of the screen.

Cases change the way users interact with the screen; sometimes, in fairly important ways. The most protective cases – and an increasingly few somewhat ruggedized phones – are notably raised right next to the screen. These protect the screen a bit from scratches and other damage when set or dropped on facedown, or when objects fly at the face of the device.

For touch inputs, raised bezels mean many users cannot reach the very edge of the screen. If they really press their finger they can get skin onto the edge, but remember the screen senses the center of the contact area. The effect is that the interaction between finger and bezel distorts the touched area and shifts the centroid of the contact towards the center of the screen in ways the user might not expect.



The distorted contact patch and centroid detected as a touch with an exaggerated raised bezel.

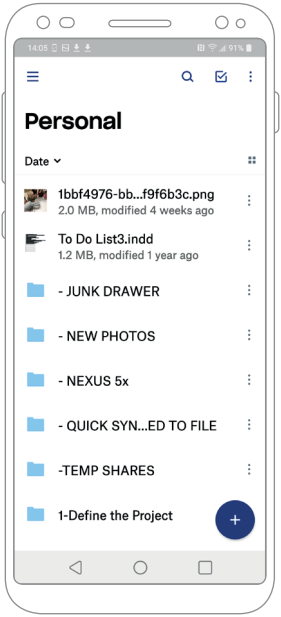
It could be severe enough that edge gestures (which must originate within a pixel or two of the edge of the screen) are simply impossible to activate. But since we probably can't design our own unique edge gesture within a modern OS anyway, taps are the important part here. Fortunately, design solutions to this are not that hard or unique.

DESIGNING AROUND CASES AND BEZELS

The top-level solution is to provide extra room for edge taps, and more tolerance for gestures that originate offscreen. A

straightforward general value is 20% added to the touch sizes defined in chapter 6. However, an easier way to deal with taps is to simply avoid placing anything on the edge. This is not as restrictive as it might seem: people can't read anything, including on paper, when there is no space between items or edges. We just need to make sure there are good margins, and extend them to the top and bottom edges.

Assume we are designing a list view with actions like delete or a dot menu (⋮) to one side of each row (as shown



An example of a list view with icons to the far side, but margins between the edge of the viewport.

in the illustration). Adding margins makes everything more readable, and it also bypasses any issues users may encounter when trying to tap the edges. The interactions farthest left and right are safely away from the edge. This advice overlaps neatly with the need for some empty space for gesturing as described in chapter 7; it doesn't add to the requirements, but instead adds more reasons to use the same design solution.

Mobile device use in the real world renders many of our assumptions about interaction incorrect. Often, these are simply confounding issues without clear solutions, but what we've learned in the last two chapters is that understanding is more important.

People and their environments are frequently not what we hope for or expect. We have to design our systems and interfaces to work for the real-world needs, expectations, and capabilities of our users.

Yet no matter how interactive a digital product is, if people cannot read and understand its text, they cannot use it. In the next chapter, we'll see how readability and affording action are essential qualities for successful touchscreen design.

Checklist

- [x] Audio notices or audio tracks may not be able to be used due to environment, social norms, or individual user capabilities. Design to provide visual or on-screen cues, notices, labels, and captions to reinforce or support audio.

- [x] Color vision deficits (color blindness) are common in the population, and the same effect of changing or removing colors can be seen by any user due to glare, odd viewing angles, and other factors that come up regularly with mobile phones. Design so icons and other graphics communicate their intent with only shape and contrast, instead of relying on color.

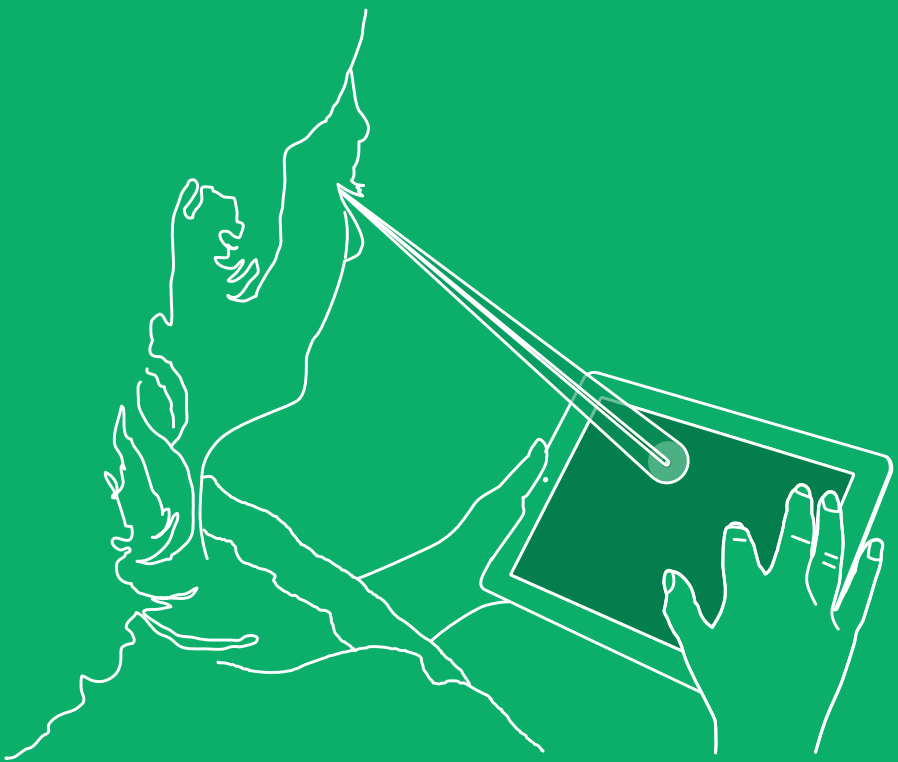
- [x] Time-limited or transient notices, warnings, or just changing data not be seen by users, as mobiles may be used in distracting environments, or other applications may draw the user away from our tasks. Design notices, warnings, and any other important data to be on-screen until the user dismisses them.

- [x] Users will sometimes be subjected to external vibration, movement, or jostling when they are trying to interact with the screen. Design to avoid catastrophes when accidental touches do happen.

- ☑ Accidental double clicks, drags or drops are common and unavoidable. Design to avoid accidents, by using a single action button, displacing confirmation actions, and assuring all actions are reversible.

- ☑ OS-provided gestures and edge elements can interfere with the design of our app or, especially, website. Design with additional spacing near the edges of the screen. Avoid use of the chyron (bottom bars) entirely on the web, and use caution with them on Android or newer iOS devices.

- ☑ Cases may interfere with touching the edge of the screen, especially to the sides. Design all interfaces with margins wide enough to allow easy reading, and interference-prone items will no longer be close enough to the edges to be an issue.



CHAPTER TEN

People Only Touch What They See

CHAPTER 10

People Only Touch What They See

Through the course of this book we have come to learn that we can't rely on best guesses, and neither can we let everyone just muddle through. We have to design to work for the broadest possible range of users and environments in the real world.

So far we've covered technology, human factors, physiology and cognitive psychology, and we have learned a lot about how people really work with digital devices, and especially with portable touchscreen phones and tablets. Now, we're going to start diving more fully into tactics.

First we will discuss UI design and how that intersects with interactive design to ensure users can see and know how to interact with our touchscreen designs. And in chapter 11 we'll cover some simple methods to design templates and pages to take advantage of all that we've learned to date.

Look and Feel

When I start a design project with a new team, one of the many misconceptions I have to fight is that UX design is just

visual design. The features are set, engineering will build the product, and I will just come make it pretty afterwards.

It is worth diving into one phrase I hear a lot: *look and feel*. This is commonly used to refer to visual design or graphic design. But by this definition it distinctly doesn't include the rest of what UX does, so it bars us from working on information architecture and most interactions.

I've long objected to the use of the phrase for this reason. Recently, however, I started looking again and noticed its two words:

- look: the visually perceptible part of the experience, the UI or *user interface design*
- feel: the response to user input, the IxD or *interaction design*

It turns out the phrase *look and feel* is actually an appropriate one that succinctly summarizes what our job is. Though it is simply misunderstood, it is a great starting point to discuss our role with client groups, or simply to remember that our job focuses on two related things.[1] As a phrase, with an *and* in the middle, it emphasizes that we can't design interactions in isolation from the visuals or vice versa.[2]

1. https://smashed.by/lookandfeel
2. https://smashed.by/visuals

People Don't Perceive All That They See

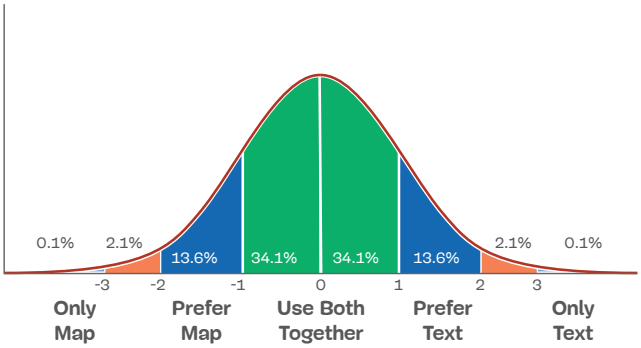
In chapter 6 I talked about how the F-pattern is part of a system to codify how people scan content in web pages. But we've found that people appear to scan, consume, and interact with content differently on small and handheld devices as compared with desktop computers.

There are a number of interesting and useful theories about how visual perception works. Several make sense, but none adequately explain how our brains process visual information. It is pretty likely that this is because several or all of the theories are true, and it is best to not think of the vision system as a camera that processes all images, but as a system of systems.

Each subsystem processes the information it specializes in, which is one reason I love the concept of *multi-encoding*. You have certainly seen this as a design principle, even if not named as such. At the simplest level, it is the reason why it is recommended to label icons with text. Graphics and text are understood by different subsystems, so we get a chance to double up on which systems process a particular input. It raises the question, however, of what if people are bad at one type of processing?

Long ago I was designing a fairly early map system and found that lots of people simply did not understand it. It performed well, judging by the top line numbers: approximately 85% of people could use it, and 60% of them used it without any issue. But unlike the failures of a normal usability test, where the "failures" were just slowness or a workaround, that last 15% simply couldn't use the map at all. Eventually, I figured out they didn't seem to understand any of the graphical data. Maps, graphs, icons, even the photos in the banner ads didn't resonate, or the users couldn't derive specific and accurate meaning from any of them.

This experience and working on many other products and tests since then has taught me that people are not all the same, but are *normally variable* in every way. Normal variation or normal distribution is the technical term for what is commonly known as the *bell curve*.[3]



The normal distribution, or bell curve, of how people use maps vs. written instructions

3. https://smashed.by/normaldistributions

What this tells us is actually super useful and is part of my entire philosophy of design: we can't design for a single idealized person. Instead we have to understand those variations. For the map project, I found that the people on the far right of the curve could read words just fine, so when I changed the design to include written descriptions and graphical maps side by side, it worked perfectly for everyone. I have extended this lesson to many other contexts, such as displaying graphs above tables.

MULTI-ENCODING

Multi-encoding for UI elements is often much more subtle – and more critical – to making the interface effective. Icons without text are a common problem, and error icons are my favorite example of the issue of perception. Here is a good example of icons I see all the time as I try to interpret what the errors might be from digital products I use.

They all follow a single consistent style, so they work nicely together and match the style of the app or website. But at a glance, what are they? They are all circles. Users will have to take a moment longer to properly read them and identify the inner shapes. We know people glance more than

they read and become accustomed to things. If we were to show them the success symbol (the tick) a lot, they might miss the warning symbol (the exclamation point) because it's just a shape in the same circle. To bypass this, many designers add color.

Easy solution! The bright yellow pops now, especially if we pay attention to the contrast and make sure to invert the inner colors, and so on. But even avoiding the use of red for the warning, what about those with color vision deficiencies? What about glare and badly adjusted screens, and other issues that make colors hard to read? We're back to the same problem of a bunch of undifferentiated circles.

What if we make the shapes themselves convey meaning? Round is friendly, square is sharper and more urgent, triangles are downright dangerous: even without the cultural familiarity that triangles denote warnings, they still look sharp and dangerous.

Adding text labels tells users what the product designer knows about the differences between the icons. Displayed like this, the icons serve the user in two different ways, allowing for a glanceable icon and a slightly deeper reading. The different modes might even address different parts of the brain. Some people will actually recognize the words better than the icons.[4]

As you've already probably surmised from the color portion of this, addressing accessibility is a perfectly natural extension of the principle of multi-encoding: individual capabilities vary from person to person and from one situation to the next. That's normal and to be planned for, so we need to make sure our designs work for everyone without making them think too hard or risking them missing out on key information.

Making Interactions Glanceable

Designing digital products is not, first and foremost, about making things pretty or delightful, and certainly not about

4. https://smashed.by/iconlabels

surprise. Digital products should most of all be easy to use, accurately and immediately. We need to make sure we're not forcing users to explore to find features or meanings. Start with what works: simple controls that work in expected ways. And the most expected controls are those that are visible and communicate what they will do.

The title of this chapter is "People Only Touch What They See," but it is really about how people will only touch things that they believe will work and result in that action. Interactive elements must:

Color Theory and Contrast

For some years now, UI designers have been surreptitiously reducing contrast, especially in digital. Black text is rarely fully black anymore, and there is a lot of gray text or backgrounds or buttons as well, besides that for graying out disabled functions. There are good reasons to manage contrast to avoid retinal overstimulation and dazzle, but I always try to maximize contrast to ensure all users can read content in all environments (see "Contrast" later in this chapter for a deeper discussion).

Note that I said contrast. The term *color contrast* is misleading. Despite it being the W3C's accepted term, I pointedly never use it because I went to school for art and graphic design, so I have

- **Attract the eye** so people notice the item at all
- **Afford action** so people know it is an action and not just data
- **Be readable** so people can tell what the action will perform
- **Inspire confidence**, through being properly positioned and large enough to select without accidentally tapping other items

a background in color theory. That's a whole field and there are books on just color, so what follows is a summary. (smashed.by/colortheory1)

Color is made up of three components:

- Hue: the spectrum on which a color appears.
- Saturation: how intense a hue appears.
- Value or brightness: the amount of black or white that is added to the color. Adding black makes *shades*; adding white makes *tints*.

To most people (including designers, developers, and product managers), the term *color* usually means hue and nothing else; color is red versus green, for example.

(continued overleaf)

Now let's go over each of these tactics, so I can give you all the tips and tricks I have learned from designing touch-screen products over the years. Yes, there's still a lot to say about how and why this all happens, including some pretty obscure stuff I've worked out on my own over time.

At the end of each section you'll find a simple checklist you can use to improve your design, but I encourage you to really read and absorb what you can about the principles, so you can still design well for edge cases or when conflicts among design principles arise.

Color Theory (continued)

The term *color contrast*, then, implies that contrast relates to hue, but it does not.

Contrast instead means the difference between the value or brightness level of two different design elements. Small differences create low contrast, so elements are hard to differentiate or see. Large differences create high contrast, so elements are easy to tell apart or read.

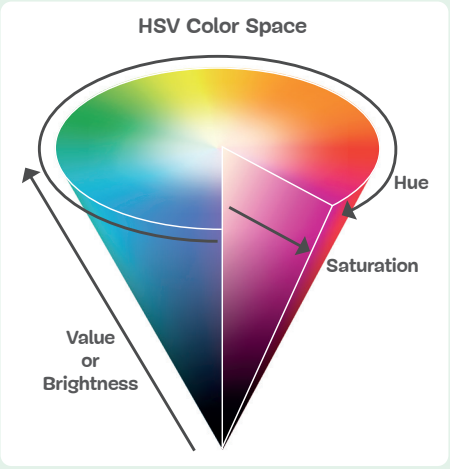
Critically, this difference exists regardless of hue. Contrast itself is important – as is understanding its impact. (smashed.by/colortheory2) Contrast lets designers address the need for readability and accessibility in the unpredictable environments

Attracting the Eye

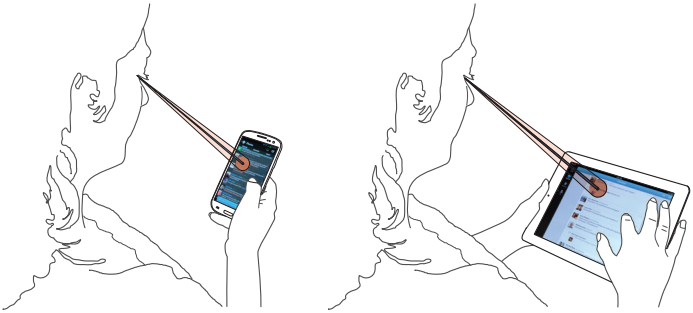
First, we must attract the users' eyes to the app, which means it literally needs to be visible. Contrast is the key way of making UI elements visible, but don't forget the other design principles that impact the contrast of icons, text, and other on-screen elements: differences in size and weight.

The AAA standard from the Web Content Accessibility Guidelines (WCAG) 2.1 recommendations that I always

in which mobile users find themselves and – most relevant to this chapter – when changing color palettes. (smashed.by/applyingcolortheory) Contrast is contrast, regardless of the color or hue of the elements.



follow defines two minimum contrast ratios: 7:1 for normal text and 4.5:1 for large text.[5] Why does contrast change for sizes? Because of the *smallest perceptible difference*.



A phone is held closer to the eye, so smaller items appear to be the same size as those on a tablet.

Smaller things are harder to see, and this varies not with size but with size at distance. We call this *angular resolution*.

As far as the smallest perceptible difference is concerned, what can be differentiated varies based on the perceived size.[6] Up close, the little ticks and numbers on a ruler are easily visible, so it can be used to measure things. But held up across a room, the observer can perceive the ruler only as a line in space.

5. https://smashed.by/contrastenhanced
6. https://smashed.by/jnd

And yet farther away, across a large parking lot, say, someone might only be able to perceive the ruler in someone's hand as something, but it has lost all shape so could be a phone or a candybar instead.[7]

This rule extends not just to font size but to component size as well. Generally speaking, thin type is harder to read than thicker type of the same size. The trend toward outline shapes in icon design, and away from filled in shapes, results in clearly less readable icons, and users take more time to find and understand them.[8]

Improve designs to attract the user's eye by:

- Increasing contrast. I like to use black and white, to get a 23:1 contrast ratio.
- Increasing the weight of type and icons. Don't use thin type unless in very large sizes.
- Using type weight to indicate prominence, and background color changes (inverting shapes) to indicate position or selection.

7. https://smashed.by/iphoneres
8. https://smashed.by/solidvsoutline

Affording Action

Next, we must afford action. *Affordance* is the characteristic of objects that tells the user what they are and what they do.[9] Although we learn many of these affordances, a lot of it appears to be more or less intrinsic. That's why we pull levers and push buttons. For touchscreen interfaces we learn that we can tap or click things, so people will try to do that. But not everything is clickable, so how do we tell the users which items to click?[10]

When scanning pages, people's eyes naturally move across gaps and spaces. The movements are stopped by lines and boxes, and especially when two boxes are next to each other. The empty area between them can be called *trapped space*.[11] Users might not scan from one box to the next as we intend. Over time I learned these lessons, and have used fewer boxes around grids of items, and thinner and fewer dividing lines, letting elements define their own space.

BOUNDING

Still, sometimes I would put a box around something to set it off in a complex page or emphasize its importance. A few years back, I placed important information on status in a box, and displayed actions to take as a list of icons with no lines or boxes after it. In a usability study out in the field, al-

9. https://smashed.by/affordancesdef
10. https://smashed.by/affordancecontext
11. https://smashed.by/tufte

most every participant clicked the box – and no one clicked the actions. After some further investigation of how people undertook this test and previous ones, along with some experimentation with the next round of design, I figured out that people generally perceive bound items as selectable, and unbound items as unselectable information.

Think how physical buttons look and work as items distinct from the background. A light switch or the volume button on a remote sticks up from the surface. This principle of interacting only with distinct or perceived-to-be-closed shapes seems to have been generally recognized in the hardware side of the design world, even if it's not a codified design principle. Most membrane panels, like those on a microwave oven, also have printed or raised edges or areas to bound the button even though they're mechanically superfluous.

Several styles of bound buttons on the flat membrane control panel of a microwave oven.

This principle also works well to help make sure we have sufficiently large touch targets without imposing other design restrictions and making the design full of large icons and words.

Once I figured this out, it explained a lot of other issues I'd found but couldn't quite identify why they failed. For example, I have long made sure all form inputs are completely closed boxes, because many users would fail to recognize anything else, even if it was the OS standard. This appears to support the principle of bound meaning selectable.

On the other hand, it is critically important not to bound unclickable elements, because people will read them as interactive elements and try tapping them to get more information. That's how I figured this out, and I keep seeing it whenever a project team makes me box something for emphasis. Of course, the alternative answer is that if the information seems like it should have a detailed version when tapped, we can just provide more information instead.

LINKING

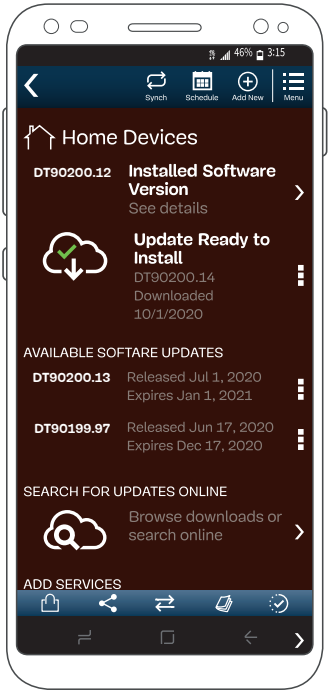
For a while I was sure that lists were bound enough that people understood they were inherently clickable, but after some experimentation on products, I have found it's not

quite true, and it helps if we include some other indicators of interaction. Simple links in lists of items in an app, or sometimes on the web, are best served (from usability tests and analytics of real products) with reveal arrows. These are usually the OS-default arrow shape placed to the far right to indicate that clicking anywhere in the row will perform the selected action.

When rows in a list or other similar items will perform an action, they certainly should not get arrows, because that means a link. They should still have icons, however, if it's possible to integrate them into the design. Sometimes there are too many actions to label them all distinctly, but ideally a recognizable icon helps a lot.

I am still not sure which is the best method: to place the icon to the far right to replace the arrow; or, if all rows can have icons to the left, whether the lack of arrows indicates functions rather than links. Both seem to work adequately, but I have yet to perform a test comparing the effectiveness of the two side-by-side on the same product.

Multiple actions per row is also a bit tricky. So far, I've just been muddling through. For example, for a row containing a link and also functional options, there are a number of ways to approach it. I usually hedge my bets



A variety of options to select, all indicating the type of function with either arrow or dot menu icons to the right, and some with function icons to the left.

and make sure the primary function is inside the options or menu, much as discussed in the part of chapter 8 about accidental clicks and the fallback functions when users miss the target on Twitter.

Inline links in text, especially on the web, should be underlined. Not just change color, nor anything else – but underlined. However, apps should use underlining carefully and

sparingly. Apps are not websites and users rapidly become familiar with the conventions and differences.

GRAYING OUT

As important as making sure people know what they can click is telling them clearly and unambiguously what they cannot click.

Never have clickable actions that result in an error because the function isn't available. Instead, indicate when items are disabled, and then disable them so no action occurs.

Disabled text, buttons, and icons have been dimmed or grayed out for a long time because it works. People understand that in a site or app of bright or high-contrast items, the gray and low-contrast stuff doesn't work.

Graying out is also overused, and too often conveys the wrong meaning. Actions the user can't perform break down into two basic categories. First, hide things the user *can't do at all*:

- Hide all functions, features, and data not available for the session because of who the user is (their account level, region, role, and so on).

- If the user cannot enable a function by pressing buttons inside the app or website, the function should not be shown.

Second, disable things the user *can't do right now*:

- Disable or gray out all features that are temporarily unavailable but that the user can resolve in-session. A simple example would be graying out a form's submit button until all the fields are complete: the user can fix the situation themselves.

Remember that disabling functions is more than a color change. We also need to disable the function so clicking doesn't result in an error. As to the visual display of graying out, the same shape in gray is often still read as the same shape or words, so it doesn't have the effect of removing the function. When a design uses gray for general purposes, this can particularly confuse users. By nature, gray has lower contrast and all too often is hard to read. I believe gray is used too much these days to be helpful for functional items. It's a design trend, so I'll keep pretending it isn't happening and keep hoping it soon fades away.

Do not forget that, at best, graying out alone is a single change, and we need to multi-encode whenever possible.

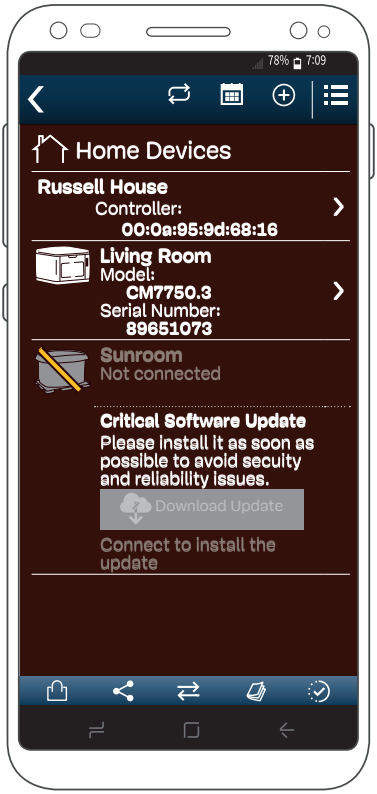
A turn lane arrow grayed out during road construction by overlaying the white arrow with black.

Instead of simply graying out the same shape, like the road construction crew did in the photo, we also need to change another aspect of the function. *Do not enter* symbols can replace activity icons on buttons, or we can cross out the icon to show by shape that the function is not available.

Improve designs for better affordance by:

- Using common and recognizable interactive items, like tab-shaped tabs.
- Placing interactive elements within existing bound spaces, such as defined rows, bars, and boxes.

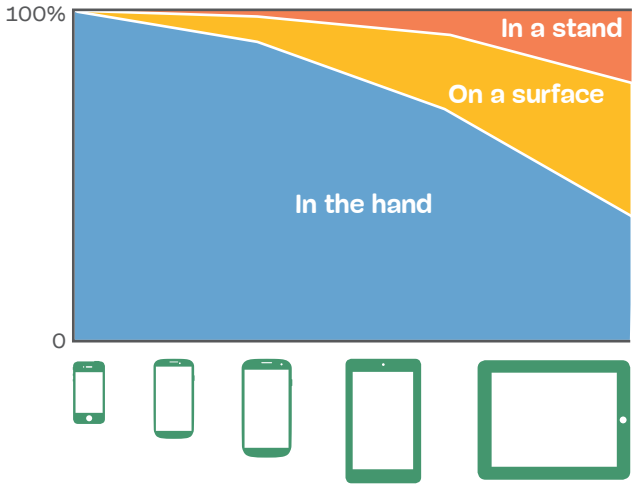
- Adding bounding shapes to floating interactive items to make them appear to be clickable.
- Using underlines for inline links.



Both the disconnected device icon and the update action are grayed out, struck through, and labeled unavailable to indicate they cannot be selected.

Making Content Readable

Aside from merely being visible, content must also be readable. That means users don't just notice or discern it but can understand what the content communicates. We can make text readable by picking more legible font faces and avoiding less legible styles. Use italics minimally or not at all, for example.



How and whether people hold different device types.

Back in chapter 5 I discussed how people use different devices in different ways. Recall that one of those differ-

ences is in the distance from the eye: smaller phones tend to be held closer to the face; tablets are mostly used at arm's length, the same distance as a laptop or desktop computer.

Why does this matter? Just a few paragraphs above I mentioned the concept of angular resolution. This tells us that type sizes don't matter because their perceived size changes with viewing distance. To pick the size to design our content, we first have to know the device it is for. There's no one good size for digital products, but one for small phones, another for large phones, yet another for tablets, and none of those are the same size as a desktop computer.

We can calculate angular resolution using this formula:

visual angle (minutes of arc)
=
(3,438)
×
length of the object
perpendicular to the line of sight
÷
distance from the front of the eye
to the object

This is actually the simple version of the formula. Getting the number 3,438 requires knowing the average wavelength of visible light and the size of the sensors in our eyes. Thus, to get the same readable type size, it has to be larger when a display is further away.

This is why, despite what many film directors say, people can watch a TV show on their phone or iPad as comfortably as they can on their TV. Since the portable devices are closer to the head than a TV or movie screen, users will have about the same field of view.

In fact, this correlation is so strong that it seems deliberate, and people prefer having their screens show a particular angle of view so move themselves near or far (or pick the right seat in the auditorium) based on this preference.

We can calculate type size for different device classes based on our understanding of what people can see and understand. I have done the math for you for the main device classes based on the typical distances at which people hold them (table overleaf).

Note that these are minimum sizes. It is critical to understand that we will need to use larger type sizes for almost all circumstances. (In the next chapter, I will give some guidelines on those sizes.)

DEVICE CLASS	MINIMUM SIZE
Small phone, feature phone, smartwatch	4
Small smartphone (4-inch screen)	6
Large smartphone, "phablet" (5–6-inch)	7
Small tablet (~7-inch)	8
Large tablet (10-inch), desktop, laptop	10

minimum type size in points

Small sizes are also difficult for ageing populations and in difficult environments, like in moving vehicles (the vibration causes pixels to blur over a larger area, effectively making the weight too large to differentiate shapes unless it is big enough).

I find the minimum sizes to be a useful guide as "never exceed" values and baselines to work from. I often use them for things like labels under icons, which as just one or two words are easier to peer at temporarily with no undue hardship.

Remember that all readable elements should be readable. Icons, status graphs, and other readable elements follow

Units and Conversions

Whenever we discuss reading text or understanding icons, we're talking about human behavior on many different devices and platforms. Many usability standards and accessibility guidelines assume that all digital content is on the web and that pixels are the standard unit of measure. Many others assume one device at one size because everyone on the team has that. But people live in the real world where things aren't measured in pixels, and devices vary. We have to use physical units and then translate them to apply to each platform.

Because of variations in device scaling, there is no such thing as a real-world translation to pixels. If you see a physical ruler that measures in pixels – and yes, they exist – it is for one device at one moment in time. I suggest literally throwing it away: it's dangerous.

For many centuries, the standard unit of measurement for type and graphic design has been the *point*. This, of course, is not the iOS device-independent pixel for which Apple inexplicably stole the name, but a *typographer's point*, which Adobe rounded slightly to 1/72 of an inch for the PostScript standard some decades ago. (This is sometimes also called a *PostScript point*.)

While there is no perfect translation between units of measure for different

(continued overleaf)

Units and Conversions (continued)

devices and platforms, we can get close enough to write specifications, instead of just throwing our hands up in the air. I use the conversion values shown in this table for all my design documentation.

I create all of my designs in drawing programs at full scale, defining type sizes – as well as icon sizes and margins – in real-world typographer's points, then translate the sizes for the developers so there is no ambiguity. This has served me well for many years, on many projects, on many platforms. (smashed.by/typesizes)

TO CONVERT FROM POINTS TO...	MULTIPLY BY
Web pixels	1.34
Web ems	0.8
Android scale- or density-independent pixels (SP or DP)	2.0
iOS points	2.25
Windows device-independent pixels (DIP), or px	1.34

the same scale rules and, roughly, the same sizes. The same concerns regarding readability apply for text.

READABLE TYPE SIZES

So far I've discussed only minimum visible type sizes. But most content needs to be larger than the minimum size to allow users to more easily consume it, whether they're glancing at it on the go on portable devices, they have vision or cognitive issues, or they are in an adverse environment (vibration, glare, and so on).

There has been enough very good research performed using eye-tracking with participants who have trouble reading small type sizes that we have a good handle on what sizes are necessary to be readable by typical users.[12] My own product research bears these findings out quite well. In the table below, in addition to the minimums, are more reasonable optimal type sizes for reading different types of content on various devices classes.[13]

These sizes are around 40% larger than minimum for basic content, and 80% larger for enhanced content. The larger size can be used both to differentiate based on importance and if we expect the environment or user needs to make larger text necessary. Note that if we go too large, readability

12. https://smashed.by/onlinereading
13. https://smashed.by/mobiletypography

DEVICE CLASS	MINIMUM	BASIC CONTENT	ENHANCED CONTENT
Small phone, feature phone, smartwatch	4	5.5	7.2
Small smartphone (4-inch screen)	6	8.5	10.8
Large smartphone, "phablet" (5–6-inch)	7	9.8	12.6
Small tablet (~7-inch)	8	11.2	14.4
Large tablet (10-inch), desktop, laptop	10	14	18

Type size in points for each device size

begins to drop again, because long words are difficult to scan easily. Don't go much over the enhanced size except for short items like titles or individual key values.

Also remember that these measurements make a lot of assumptions about standard device usage. We always need to find out how our users work with the actual product in their actual environment.

Older users with poor eyes (like me) and prevalent usage of devices on buses and when walking down the street necessitate larger type sizes.[14] More use outside means more glare, so we need larger type sizes, better contrast ratios, or both.

A favorite example I have seen across industries is service technicians who buy magnetic holders for their tablets, then stick them to a nearby surface such as the vehicle body or a duct. They can then work but also read instructions. These often end up a little further away than usual, so defaulting to larger sizes is recommended.

And finally: always give users choice. Many people do not change their settings or access accessibility modes, but allow those who do know about changing default type sizes or who zoom the website to make that choice. Respect the control inputs and settings. Never lock out users from basic functions just because we think we know better.

14. https://smashed.by/fonteffects

COLOR VISION DEFICIENCIES

Color vision deficiencies are commonly called *color blindness*, though complete loss of color vision is extremely rare. Instead, different deficiencies mean certain colors cannot be distinguished, can only be distinguished with difficulty, or are not as prominent.

The multi-encoding warning icon exercise I outlined earlier is something that grew out of what I encountered over a period of years and many usability tests. One participant has stuck with me, and I've made sure I never forget the lesson of it. We were testing a process that involved the user reacting to an unexpected error. One user simply did not get it and I couldn't figure out why.

Through the eye-tracking, I could see his gaze go right over the big, bright yellow warning icon. But even when prompted, he never noticed it. At the end of the test, I asked him to look at an alternative design where the warning was a triangle instead of a circle, but there was no other difference. He immediately noticed it.

Almost certainly he had a slight color vision deficit, and we overcame that by multi-encoding with a shape distinct enough to catch the eye without color.

Red–green color blindness is by far the most common form, but blue–yellow and other forms also exist. Color deficiencies of one sort or another occur in about 8% to 12% of men and about 0.5% to 1% of women.

So says the official information, which we ought to qualify with some significant caveats. Such figures are usually based on populations of Western European origin, and they only represent those people who have sought medical help.[15]

More minor color vision deficits are often not obvious to the individual: they can still tell colors apart and learn to cope with how they see. My impression is that over 20% of the population has some kind of color vision deficiency. Disability is not necessarily a health condition, but rather when interactions mismatch expectations;[16] at a school for deaf children, the individual who doesn't sign is the disabled one.

As I discussed in the previous chapter, color blindness is not an illness but a condition that, like distraction or transient physical circumstances, changes how people work. We can't assume that disabled people are a class, but must assume the concept of *temporary disability* can impact anyone and design to account for it. When someone is in the kitchen or the workshop, might they have dirty hands? When it's cold,

15. https://smashed.by/prevalence
16. https://smashed.by/microsoftinclusive

they wear gloves. How do they use their phone or computer? Differently. Maybe poorly.[17]

Environmental conditions are many and varied, as I touched on in chapter 9, and some create problems similar in effect to color blindness. Glare is the most common and can reduce device contrast, but there are also dust and dirt on the screen, rain, wearing sunglasses, or simply viewing at restricted angles. All of these factors can destroy the visibility of colors and change their appearance.

We address issues of color blindness, glare, and other color-altering effects by designing with contrast as the first choice for all color selections.

CONTRAST

The W3C's WCAG 2.1 provides the most robust and widespread recommendations, but they are very limited in their scope. Their recommendations are tied to web pixel sizes, and while they are slowly becoming aware that more use is on mobile devices than on desktop, WCAG 2.1 still makes a lot of assumptions about viewing distance, screen size, and – most of all – screen position.

17. https://smashed.by/accessibility101

As mentioned above, I only ever use the enhanced WCAG 2.1 AAA-level standards. Their contrast ratios are: 7:1 for normal text; and 4.5:1 for large text.

However, I have found over the years of designing digital products that having the best possible contrast ratio is important. Whenever possible, I use black and white for almost all content. This gives a contrast ratio of 23:1, so you can see that 7:1 isn't a terribly stringent requirement. If we put some thought into ensuring proper contrast early in the design process we will end up with a good color palette to choose from for our designs.

On mobile devices, I usually prefer to use white text on a black background for two key reasons. Primarily, I do this because white on black reduces the total amount of light reaching the user's eyes. This means that high screen brightness works well in the dark as well as in full daylight.

Remember, these are mobile devices: people move around constantly, and we can't tell under what lighting conditions they might use them. We should not rely on users changing their brightness settings but design for the broadest possible use.

Dark mode is also more readable in normal lighting conditions than light mode,[18] and is a way to mitigate issues with excessive contrast for people with myopia or other vision issues.[19]

The other reason I like dark mode or *inverted* or *negative polarity design* is the way many mobile device screens work. Dark mode reduces power consumption because only the pixels that are lit up consume power, as opposed to the old method of backlight and masking. I'll often tweak the recommendation of a black background and use a brand-adjacent color such as very, very dark blue or brown instead, so the branded colors for the rest of the design look good and the same principle applies.

BLINKING

Readable content, first and foremost, is visible content. Aside from the issues of size, weight, and contrast, there's another tricky one I encounter occasionally: content disappearing periodically.

Blinking is a very common signaling technique to indicate a warning or notification. It's even codified into international standards to mean something more important or danger-

18. https://smashed.by/metrosigns
19. https://smashed.by/myopia

ous than a steadily illuminated light. Your phone may still have a blinking LED at the top to indicate a new message or notice. However, the logic that drives the fundamental nature of blinking lights as a warning is flawed because of a change in technology.

When all electric lights were incandescent, they had considerable start and stop times. The filament took a noticeable amount of time to power on and then to go dark after the power was removed. For a simple blink circuit, applying and cutting power would not make the light turn on and off, but instead would slowly build to full power before gradually dropping off. The light would pulse between off and on, instead of blinking. This is why many warning lights, like those on top of fire trucks or police cars, rotated instead.

LEDs, on the other hand, turn on and off almost instantly. When the blink cycle is off, the light is completely off. A problem I have come across many times is that people happen to glance at a panel or the top of their phone *between* blink cycles, so they can miss the bright, blinking light entirely or just see a blink out of the corner of their eye (outside of their foveal vision range, which will be discussed in the next chapter) then quickly look at it only to see that it is off.

To avoid this problem, we can ditch blinking entirely. It's rarely needed, nor the best solution. If it is essential, perhaps to comply with old standards or legacy behavior, then have the software or circuitry simulate an old-fashioned light bulb.

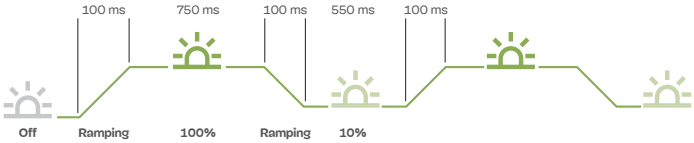


Diagram specifying the blink-rate behavior for an LED.

Bring the light slowly up and down, and never ever turn it all the way off. Users should see it no matter when in the cycle they observe the light.[20]

Improve designs to make them more readable by:

- Increasing type weight and contrast.
- Making sure type is the right size for viewing on the target devices.
- Treating icons like type and making sure they are of readable size, weight, and contrast.

20. https://smashed.by/touchlanguage

- Adding text labels to all icons, and icons to many labels.
- Using color as a secondary effect, and designing for contrast to work without any visible colors.
- Avoiding blinking or other transient, temporarily or periodically visible items.

Inspiring Confidence

In chapter 9 I talked about the many confounding influences on touch accuracy. While people can miss the listed touch accuracy any time, it massively increases when walking, when jostled by others or on a moving vehicle, or even just carrying something in the other hand. People are aware of this, and will often avoid trying to interact with their mobile devices until they reach somewhere they will have more control.

From my observations in usability tests and the research outlined in chapter 6, it seems that users are often leery of clicking anything that they don't understand, is potentially dangerous, or is too close to another item. We can design our digital products to help alleviate these concerns, and make sure users don't simply fail to interact with the systems. Refer back to the topics covered in chapter 8 under "Avoiding Touch Problems through Design" to ensure there are no conflicts with touch selection.

Improve designs to inspire confidence by:

- Making interactive elements large enough. Use the center-out touch size guidelines in chapter 6, and make the selectable targets whole boxes, not just words and icons.

- Isolating interactions. To avoid mistaken targets, don't position them too close to other elements. Provide room for comfortable scrolling and other gestures (as discussed in chapter 7).

- Indicating selection. Make sure selection is immediately shown and is large enough to be seen around fingers and thumbs, so users are confident their selection was made properly.

No matter how interactive a digital product is, if people cannot read and understand its text, they cannot use it. Effective use of typography starts with basic legibility, which depends on choosing the right type sizes and using

good contrast. Whether we're designing apps for mobile devices or websites, make sure your basic design principles include appropriate type sizes, a good sense of hierarchy, and some way to ensure readable contrast.

Readability and affordance go hand in hand. Maybe your users know the icon is clickable, but do they know what clicking it does? We may have to label them to make that clear.

Checklist

- [x] Remember that people vary, so multi-encode (color, shape, text…) all content to assure everyone can read it. Use text labels on all icons, and add icons to as many text interactive elements as we can.
- [x] As much as 20% of the population may have some level of color blindness, and everyone can suffer the same effects from glare, dirt, moisture, eyewear, or viewing angles.
- [x] Make sure all interactions are designed to attract the eye, afford action, are readable, and inspire confidence they can be safely tapped.
- [x] Increase the weight of type and use solid icons. Avoid thin type and outline icons except in large sizes.

- ☑ Use the highest contrast we can, so content is readable in as many environments and for as many users as possible.
- ☑ Use type weight to indicate prominence, and background color changes (inverting shapes) to indicate position or selection.
- ☑ Angular resolution determines how large type needs to be based on how far away the user is from the device. Different devices are used at different distances, so for mobile devices minimum sizes vary from 4 to 10 points.
- ☑ Make interactive items appear as common and recognizable items, buttons, tabs, and inputs.
- ☑ Bound interactive elements, or place them within existing bound spaces, such as defined rows, bars, and boxes.
- ☑ Avoid blinking, or other transient, temporarily or periodically visible items.
- ☑ Make sure interactions are large enough, and isolated from other items so users are confident they can tap without inadvertent consequences even in difficult environments.

CHAPTER ELEVEN

1, 2, 3: Designing by Zones

CHAPTER 11

Designing by Zones

We're almost done, and from this point on the focus will be on information design, and how we can use all that we have learned so far about how people look, see, hold, and touch to provide a single, simple method to design templates and pages for our mobile touchscreen apps and websites.

Information Design

In chapter 10 I briefly mentioned that we need to move past how the role of user experience (UX) design is commonly perceived by most clients and project team members. I'd like to explore that a little more deeply here.

As someone who worked for a few years in print graphic design before going more or less full-time to digital, I believe that graphic design has some lessons we can follow in the digital era. One of those is that we should never design pages, but instead templates.

I spent a lot of my graphic design career like most designers, working on a series of projects for a client. We kept design consistent across products by using and developing

style guides and creating templates for each type of item that would be reused. Each brochure, catalog, and advertisement looked like the others and part of a single whole. All together, they are perceived by the reader to communicate a single sense of what the organization meant or was trying to tell them.

The same process works well for us in digital products. The app or website is not one page, and it is unlikely to be a single app or website at all but both, as well as emails and SMS messages, maybe packaging and printed manuals, and perhaps even installed hardware and control panels. To make this all sensible and perform like a single experience, we shouldn't design each view or page individually. How do we approach this? We create templates that can be used consistently on an app or website.

The term *information design* predates digital design philosophies by several decades. Originally this term was mostly applied to multidisciplinary library science, such as codifying the information placards that accompany museum exhibits. My favorite definition of information design is that it's about *sense-making*; that is, making sense of information by ordering, presenting, and designing it so people can easily and accurately understand it.[1] In the digital era, I and some others use this term to refer to defining the structure of information presentation within a single view. We can

1. https://smashed.by/informationdesign

think of this as the information architecture of a single page – the defining structure beneath all the other elements.[2]

If we don't have a consistent, sensible, and shared information design, it doesn't matter how good the UI of any one page is. We have to create reusable and explainable templates that apply to the entire product set, even across platforms.

How do we do that? First, by being conscious designers and understanding what and how we are designing.

Hierarchy of Design

Throughout this book, but especially in the previous chapter, I've talked a lot about how people perceive and interact with onscreen elements. Through a mixture of theory and practice over the years, I've developed a way to leverage how people perceive the visual attributes of items onscreen to communicate the relative importance of and relationships between informational elements on the page.

This concept, usually described as a visual hierarchy, is well understood but not as widely as I'd expect.[3] There are several examples, such as the four basic principles in Robin Williams' *The Non-Designers Design Book*, but I have tweaked some of them based on my experience to reorder, remove, add, and

2. https://smashed.by/adaptiveinformation
3. https://smashed.by/visualhierarchy

emphasize slightly different attributes. From most to least important, they are:

1. Position
2. Size
3. Shape
4. Contrast
5. Color
6. Form

POSITION

What is placed higher on the page is more important – but only within reason. Look at the center-out viewing area charts in chapter 6 and don't put the first element too high. Position also controls relationships. Adjacent items are generally assumed to be of similar importance and related to each other.

SIZE

Larger elements attract more attention as well as providing more room for content, thereby giving more value. However, don't let them become too big, either obscuring other items

or exceeding the area people can view. Buttons can be so large that they are not recognized as buttons, for example. See the explanation of foveal vision later in this chapter.

SHAPE

Attributes like cuteness and danger are not restricted to pets or cartoon characters but extend to simple shapes. Pointed shapes attract attention. Warnings should be triangles and helpful icons circles, for example. Rounded corners on some boxes and squared corners on others will imply meaning, so make sure it is true, and don't just pick and assign shapes or treatments arbitrarily.

CONTRAST

Contrast refers not to color but the comparative value (darkness) between two different elements, discounting color. Elements with more contrast are more easily read, less affected by lighting conditions, and not as troublesome for users with color vision deficiencies. Relative contrast can also help connect or separate items. Those with similar value are considered either related or of similar importance. Before becoming so dim they are hard to see or so desaturated they are perceived as grayed out, lower-contrast items are seen as less important in the same way that smaller type is less important, so reducing contrast can be useful for notes and hints.

COLOR

It's true that high-visibility colors attract more attention, though there are significant caveats. The most important is to always acknowledge color-blind users. Glare can make certain colors less prominent. Pervasive use of brand color can also be problematic; if the brand uses red and there's a red masthead, we cannot rely on users understanding that yet another red bar or red text is a warning. Desaturated colors can be interpreted as disabled functionality, so be careful with tints or shades, or entirely designing with gray.

FORM

The least important attribute is the specific form of an element. Form comprises type treatments like bold and italics, underlined links, and direction, such as arrows changing which way they point. Whenever someone asks you to apply bold – to menu items to indicate where they are, or to highlight important information – remember that form is last in this list, the least consequential method. You might get better results by changing an attribute further up.

Note what is not in the list: the content. How we design is *about* the content but not *of* it. Templates can use any

content, which will change from page to page or over time. Several times above I have mentioned the relative size of text, but very often this is approached non-scientifically, and type is bigger or heavier or somehow different in a way that just looks good. You know, of course, that won't work for me, and we can come up with an ordered, repeatable method instead.

Content Hierarchies

After I figured out the minimum and readable type sizes presented in chapter 10, I started trying to use them for projects and immediately bumped into more issues I hadn't foreseen. Fortunately, the same principles of smallest perceptible difference work, and a simple set of steps can be created to foster an easily understood hierarchy of content.

We can, of course, style suitably on top of this. Want to use bold? Go ahead. But think about the consequences of using lower-contrast, heavier, or lighter weight type. These size levels are for typography so may need to be nudged up and down to compensate for other changes we add.

As explained in chapter 10, these sizes are in typographer's points. We can easily convert them to other sizes depending on the platform needs. Don't confuse them with iOS points, or with pixel sizes.

While I have used HTML element names (H1, H2, H3) to indicate the hierarchy of headings, this is just because they are very short and convenient labels. The same principle holds true for any platform, with H1 as the title (and only one per page), H2 the first subtitle, and so on.

We can, of course, use these relative sizes for any other key information that requires hierarchical differentiation, such as labels on a dashboard.

DEVICE CLASS	MINIMUM	BASIC CONTENT	ENHANCED CONTENT
Small phone, feature phone, smartwatch	4	5.5	7.2
Small smartphone (4-inch screen)	6	8.5	10.8
Large smartphone, "phablet" (5–6-inch)	7	9.8	12.6
Small tablet (~7-inch)	8	11.2	14.4
Large tablet (10-inch), desktop, laptop	10	14	18

Type size in points

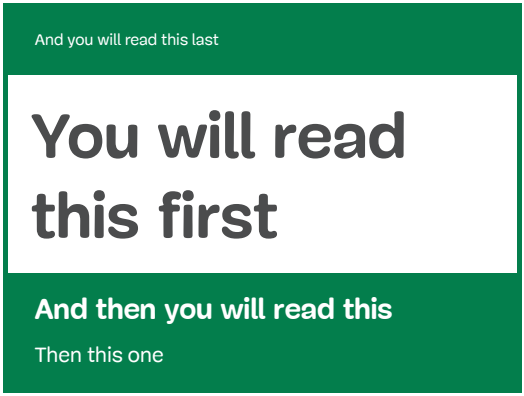
Note that I've listed only three levels of hierarchy: H1, H2, and H3. My experience in putting complex data in front of people is that more than three levels of headings becomes difficult to understand, regardless of how much we try to differentiate them. If you only need fewer levels then you could get away with cutting off sizes at either end of the range. Only have page titles? Then make the titles the H2 or H3 size. This works pretty well when they need to take up less space.

H3	H2	H1
8.5	10.8	14.4
12.6	16.2	21.6
12.6	18.9	25.2
14.4	21.6	28.8
18	27	36

HTML heading size in points

I don't talk about it as much as many people do, but mobile devices have small screens, so larger type sizes can cause problems. Multiline titles, at any level, are usually not a good thing. They are harder to read and at the same time attract undue attention owing to their size (#2 in the design hierarchy above). Work with the actual product content and the smallest likely device, then choose the size that works best or ask to edit the content. If short headings abound, you might be able to try an even larger type size.

Remember that text-oriented people are inclined to read from the center first on mobile devices, so be sure to use the type hierarchy as presented here. Expect users to read the title first, then other content more or less in the order



In combination, our center-out inclination and type size hierarchy will – probably – make users read the top line of this box last.

offered by the hierarchy – not top to bottom. Expect that users may scan up to read smaller application or site titles, and controls in the header and footer.

I have seen some evidence, from others' research and my own, that excessively large sizes of type, icons, and form controls can become unreadable or unrecognizable.[4]

My favorite example from a usability test was a button whose size was so inflated that users thought it was a banner ad. Staring right at the big, red button, they could not find it and submit the form because it didn't look like what they expected. This is related to field of view issues, which, aside from the angular resolution discussions that led us to these sizes, can also impact design in ways that might surprise you.

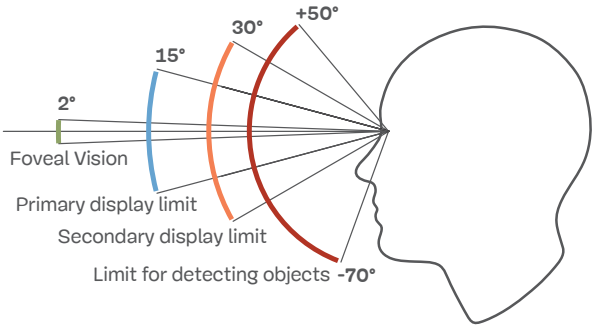
Foveal Vision and Design Speed

The common model of how the eye works is probably deeply influenced by our understanding of how cameras work. Our eye has a lens, the rods and cones are like the digital sensors, and finally the brain processes the image. Aside from the image processing being both more complex and a lot more unknown to us than just "look at all pixels," the area viewed is actually a lot smaller than is commonly understood.

4. https://smashed.by/sizematters

Yes, the sensors in the eye are rods and cones, with the rods detecting light intensity, and the cones detecting the color of the light falling on them. But they are not scattered evenly through the eye. Almost all the cones are right behind the pupil.

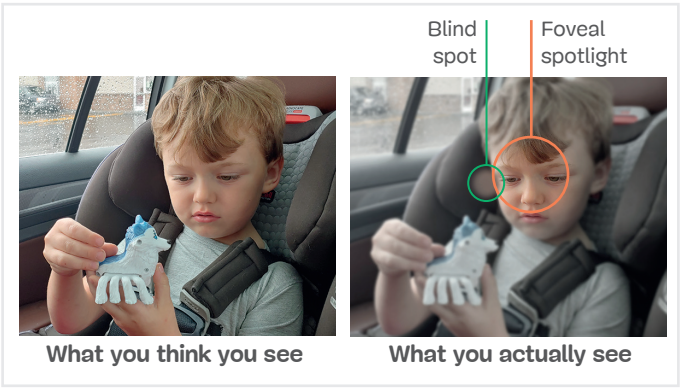
Areas the eye can see with different amounts of acuity



Rods live everywhere but especially where the cones are not. The eye flits about to scan scenes, and our brains interpret the image so we think we see a complete color scene of the world – but we don't.[5]

Instead, we only sharply and clearly see a fairly small area centered right where we're looking. A good rule of thumb is that this area is the size of a closed fist held at arm's length. Try that now, and see how little of your field of vision is the foveal vision area.[6]

5. https://smashed.by/spotlight
6. https://smashed.by/fovealload



An image simulating a moment of actual human vision, showing the small foveal vision area of sharpness and full color.

It should now be very clear why things that are too large aren't perceived properly as single objects. Or how very long lines of text are hard to read, as the eye has to move so much more to get to both ends of it.[7]

Also, realize that focus isn't just a physiological area but is also defined cognitively. Our brains can, at best, only handle input a little smaller than the foveal vision area, and this can be smaller yet based on other factors.

It turns out the old trope of being so focused that people miss other actions is totally true – and measurable. The blinking discussed in chapter 10 is an example I have seen of this effect in digital experiences.[8]

7. https://smashed.by/automotivedisplays
8. https://smashed.by/deaf

The human brain can only take in so much information, regardless of the input channel. Roadway designers know that the faster someone drives, the more they have to zero in on what's in front of them; their brains can't process everything that's going on as it flashes by. They can miss road signs, cross-traffic, or pedestrians off to the side.

These designers have a concept called the *design speed* of a roadway that, in theory, means the road speed is matched to the amount the driver has to pay attention to peripheral activities, such as those pedestrians, cross-traffic, or signs.[9]

A higher design speed limits the expected field of view of the driver.

Highway signs are very large because they must be seen from far away. Drivers need time to see them; at speed, attention is very narrowly focused on a spot far down the

9. https://smashed.by/crosswalks

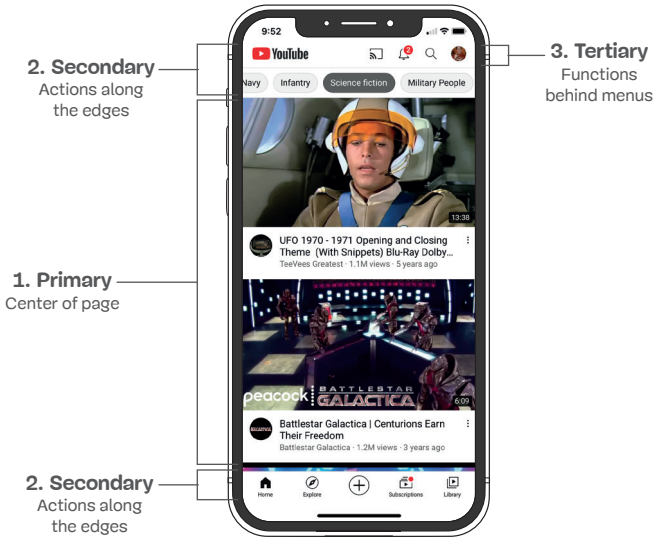
road. I like the concept of design speed for all design purposes. We should design for an expected amount of information being absorbed at once.

What is the design speed of a particular website or app? How do we place items on the page to be visible to people at speed, without getting in the way? These are questions worth considering when designing an app or a website. And right now we'll address that with a simple method of placing items by zones.

The Three Zones

Here's where we apply everything we've learned, and I'll give you a simple, easy-to-remember formula for designing the high-level structure of mobile apps and websites. Place items so:

1. The primary content or functionality is in the middle.
2. The secondary information or controls are visible along the edges as buttons or tabs.
3. The tertiary or rarely used items are hidden behind menus in the corners.



The three zones as shown on the YouTube mobile app.

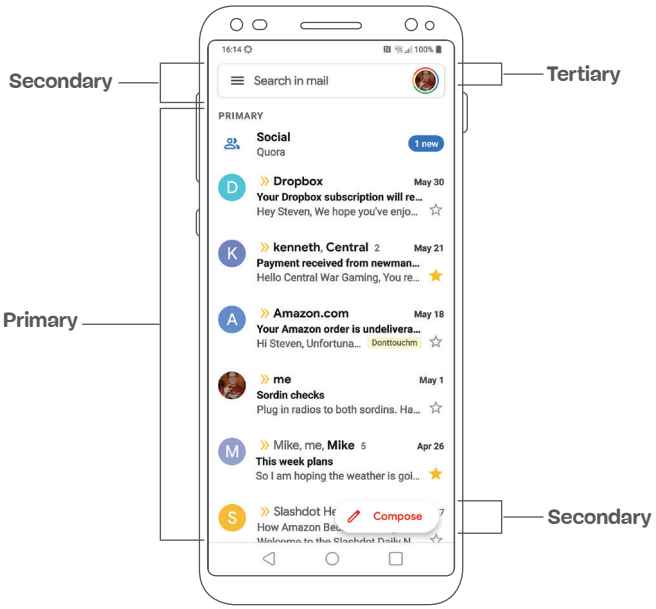
When product teams write stories and create PowerPoint design briefs, they very often say the landing page will be a grid of icons for all the functions, but that's not how anything we use day-to-day works. Users don't open their email client – or SMS, or Slack, or Twitter, or Facebook – and decide to view, compose, or search. Instead, they are all already arranged along this three-tier hierarchy:

1. Messages in a list in the middle.

2. Compose a new message, search, or other key functions along the sides.

3. Options such as settings, formatting, or add more people behind menus.

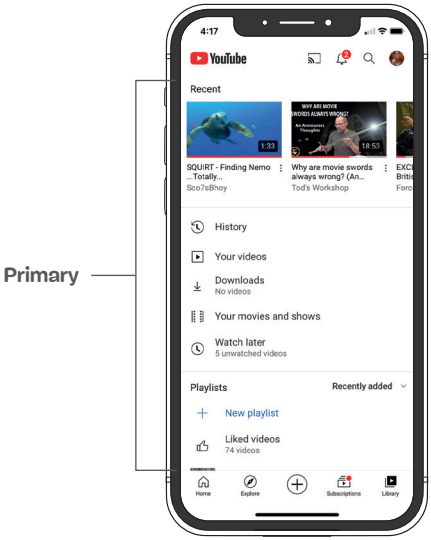
Gmail, for example, has a list of messages, a compose button, a search box, and two menus.



How the Gmail app is arranged to use the three zones.

When structuring the content and functions for our apps or websites, no matter what they do, take all the content and functions and rank them: 1, 2, 3.

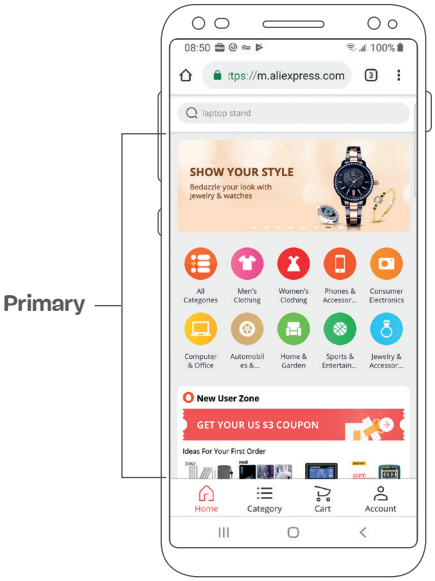
Let's go back to the YouTube app, and see how a more complex page for details about a video works. There isn't just a single list of content but several lists with categories and functionality all grouped up but also still in the center of the page.



The YouTube app places the most important content in the center.

The YouTube team trusts that the center is the best place to look, and that people scroll and understand subsections. They don't add subtabs or pulldowns or menus or anything else that makes it more complex and they don't have an animated icon to say "scroll down" to encourage people to move.

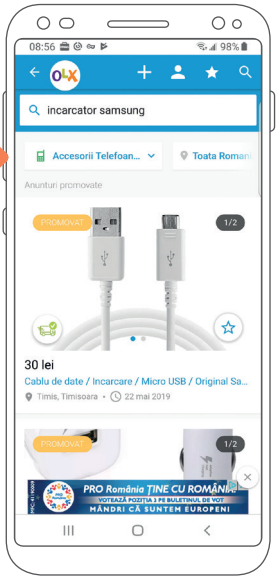
As we continue to discuss lists and categories, you might have questions about the menus. The answer should be to put them in the middle of the page. If the main goal of the page is for people to see the items in the menu first, then just place them in the center of the page. Not on every page, however, but on the home page or on a content landing page. The content is the menu. A lot of e-commerce sites do this, but AliExpress is a very good example that can easily be seen without scrolling too much.



The AliExpress website wants visitors to see the menu options straightaway, so displays them in the center of the screen.

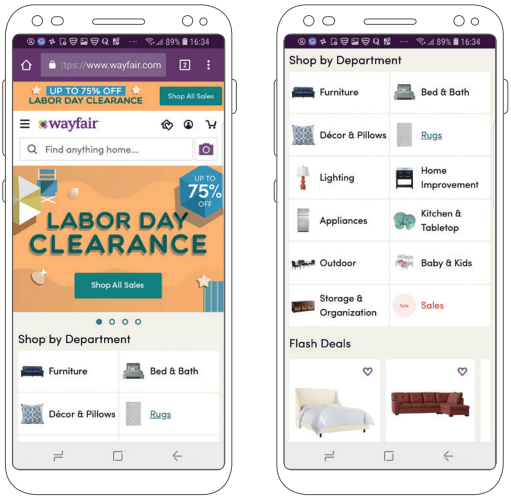
It is important to understand our users' needs and not think how designers or engineers might approach the product. Everyone on the product team understands the structure and will make assumptions about how it will work based on how it was designed or how it might solve the issues. Most visitors – to most sites, at least – do not come to the home page and then drill down by category. They arrive by search, referrals, and links from promotional emails or social media. We have to design the information architecture, the navigation, wayfinding, and information design to reflect how people really work.

The OLX product details pages orient the user with the product category right at the top of the page.



We cannot solve the biggest design issues at the page level, but we have to plan for people doing the unexpected. Spend time on category labels and other hints about which section customers are in – a practice called wayfinding – or offering things like related products. For example, OLX (OnLine eXchange, a multinational e-commerce marketplace) always leads with categories as part of the title. But not enough organizations do that.

Wayfair also has a category list on the home page, but note that it's allowed to take up more room and speaks well to the issue of "the fold." That is an old newspaper term: articles on page one above the fold meant they were visible on the

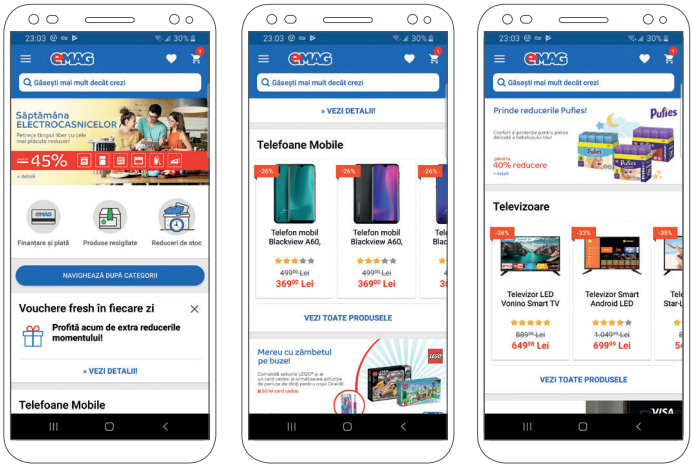


The Wayfair product category list, at the bottom of the page on entry, and scrolled to see all the categories.

newsstand. What is above the fold in digital means what is visible in the viewport when first loaded.

The problem, of course, is that it's meaningless. There are an infinite number of screen sizes, browser configurations, and so on, so we can't possibly predict viewport size. Nor should we bother. Wayfair has simply made sure the "Shop by Department" title and items are high enough on the page that most people will see a few items and then scroll to see the rest if they want to explore that way.

The Romanian e-commerce app eMag also has a few key bits of information, and then as the user scrolls they see actual products in little horizontal lists.



Product categories shown after scrolling down in the eMag app.

In this chapter we reviewed the principles of information design, touched on how human vision and visual processing works, and used all we know about where people view and touch to create a simple hierarchy of template design.

Next, let's finish off this tour of design tips with a review of the pros and cons of some of the design elements I've shown off here, such as menus, lists, floating bars, and tabs, to see how they can integrate with the concept of information design for center-out touchscreen products.

Checklist

- [x] Employ a visual hierarchy to organize information, by: position, then size, shape, contrast, color, and lastly form.
- [x] Use type sizes to communicate hierarchies, such as title level. Assure they are sufficiently different in size to be clearly understood.
- [x] People can only see a small area at high resolution and in full color, and only pay attention to so many inputs at once. Don't overload users with too many inputs.

- ☑ Organize product templates to place content into three zones:
  - **Primary** content or functionality in the middle,
  - **Secondary** information or controls visible along the edges as buttons or tabs, and
  - **Tertiary** items hidden behind menus in the corners.
- ☑ Don't hide primary information. If a primary function is navigation, place it in the center of the page.

CHAPTER TWELVE

Progressive Disclosure

CHAPTER 12

Progressive Disclosure

In the last chapter, I presented a comprehensive theory on the practice of information design, and introduced a new hierarchy of design specific to mobile touchscreens you can use immediately. Now I am going to expand a little more on that, with an overview of the principles of how we offer up the ocean of information available to us so people can consume it, and review the key tactics to make that work, along with the pros and cons of each.

Interaction Is Hypermedia

We need to approach information design in a proactive way. We need to make conscious decisions about what information needs to be visible and what can be a click or a scroll away, and how much we need to make the user aware of these choices.

In the early 2000s, one of the big buzzwords was *progressive disclosure*. A lot of what occupied our time back then were either dead ends or turned into other issues. But that one principle of progressive disclosure really stuck with me, and over time I've realized it is one of the deepest principles not

only of the web but of interactive systems in general. Internet pioneer Ted Nelson, summarizing his seminal work on hypermedia way back in 1977, wrote:

> *The hypertext concept is obvious. The thinkertoy concept may not seem obvious at first. Think yourself to a world of only screens, though, and keeping track of what you are looking at and thinking about becomes the fundamental problem.*
> —From "To the reader" in "Selected Papers 1965–77"

The problem we try to solve every day is not just about providing functionality or cutting it down for simplicity, but of making the functionality we do offer apparent, clear, and easy for users to understand. Desktop web design set aside a lot of this thinking as the size of screens kept increasing, allowing every widget to be on the portal and every column in the table. Then the user was left to work out the problems.

By far the least interesting part of what makes mobile computing interesting is how it's smaller – but it is a huge constraint sometimes. And for presenting information it's one of the most important parts. Generally, mobile phones present one bit of information at a time. OK, maybe not one piece but one type. The user sees a list, and to see what's under each list item they have to tap to see more.

Most of the designers I have worked with call the user's action to get more details *drilling down*. What happens when you tap? There are many solutions. The most common and useful are:

- Pop-ups
- Drawers
- Accordions
- Tabs
- New pages
- Scrolling

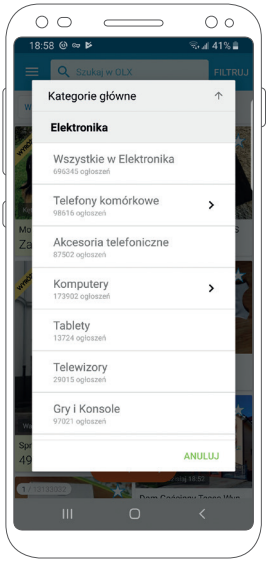
Let's review each of those in turn, and examine how they work or present problems for mobile touchscreen interfaces.

Pop-Ups

The pop-up is very popular. Most weeks I get a requirement to add a pop-up to some existing part of one of my projects. However, I rarely implement them. The design world has figured out pop-ups are problematic, and a non-scientific survey indicates there are fewer pop-ups than just a few

years ago. In fact, it's starting to get hard to find enough current examples on otherwise good websites or apps when I want to talk about real-world cases.

Pop-ups or *dialogs* are fine for some purposes, but drilling down to see more content or functionality is rarely one of them. The pop-up came into heavy use because it preserves context – the original page remains in the background – but on a practical level their context is vague, and the controls within the pop-up are often limited. This pop-up may have launched from one link in the list behind, but which one? The user can't easily tell.



A category selector to allow drilling down into a product list, using a pop-up dialog.

In addition, users see far too many pop-ups for signing in, displaying error messages, and especially for pushing unwanted promotions and sales. As we move to proper use of them, the improper uses are taking over, and users are becoming more and more accustomed to automatically dismissing pop-ups now more than ever.

A lightbox with a two-item slideshow and available zooming controls.

One way I have successfully used pop-ups is in the form of lightboxes; that is, pop-ups optimized for image viewing. The name is derived from the old term for backlit boxes used to view batches of film negatives or slides. On mobile

devices, it can be especially hard to display large enough images on a page, or to enable zooming without interfering with the rest of the page controls. Using a lightbox lets the user tap an image to open it in a full-screen dialog, which then allows them to pinch-zoom and scroll, as the only other accessible content is the dismiss or close button.

For complex cases such as diagrams, I have even implemented sketching tools to help mitigate the small-screen issues, which also can work in a lightbox.

BENEFITS

- Pop-ups are easy and quick to implement, often with the smallest amount of code. Native dialogs in mobile apps are very capable and are pre-styled, so easy to build.

- If loading page-wide content or functionality, the dialog context is usually suitable.

- There is no need for a masthead, title bar, navigation, wayfinding breadcrumbs, or action bar. Full-screen pop-ups are especially useful. Aside from the title, most of the viewing area is dedicated to the content or function.

DOWNSIDES

- Pop-ups are not as contextual as you'd think. Since they hover over the center of the whole page, users can forget what specific element they clicked to initiate them.
- Pop-ups scroll poorly. While scrolling is technically possible, it can be confusing, causing users simply to miss overflow content.
- Pop-ups are overused, so people often dismiss them without reading their content.

Drawers (and Hamburger Menus)

The much more contextual cousin of the pop-up is the drawer. Drawers are so named because they are layers that slide out. The most important aspect of drawers is that they remain docked to their origin, giving a sense of specific context. They are similar to pop-ups in that the drawer is noticeably smaller than the device's viewport.

Drawers are often used in places where combo boxes or pulldown lists would be used on desktop applications or

web interfaces. OS-default mobile pickers and selectors are generally drawn as page-level pop-ups or drawers that appear from the lower edge of the viewport and so lose the context normally expected. A custom drawer control restores this contextuality.

These may seem simple, but we need to stop and talk about drawers more than other components. One implementation has become both almost universal and very divisive among the design community over the last few years.

A drawer used to select the sort order for a list of products.

That's because the hamburger icon typically launches its menu as a drawer. Much of the conventional wisdom is that drawer menus, and the hamburger menu specifically, is hard for users to understand, and the options are hidden.

The three-line hamburger menu in a mobile application.

This is narrowly true, and I agree that we should not hide navigation behind a menu. Don't do that, and all our problems with this pattern are solved.

If we are building something like an e-commerce site and a critical function is to reveal categories to users who don't know what is sold there, then that's the primary content for the page: stick the category list right in the middle of the page, as I discussed and illustrated during the three-step hierarchy discussion in chapter 11.

However, if we have lots of tertiary functions – as we often do – and we expect users to know they exist to seek out as options, then menus work great. By "menus work great" I mean that in my usability testing they work great. When the tasks users need to find are behind a hamburger menu, they have found them every time in my experience. My favorite anecdote involves a test participant who only carried a feature phone, rarely used the computer at the repair shop, had no computer at home, whose children are long grown, and had essentially no familiarity with the conventions of mobile devices like menu icons: he found it in approximately three seconds.[1]

But where do we put menu icons, and how do we label them? I am hard-pressed to say any one thing works much better or worse than another. Part of the research I reviewed in chapter 6 involved testing some menu variations. I found no statistical difference in use or speed whether the menu was on the right or left side of the title bar. Then and in subsequent usability tests and product analytics, I have also found no performance issues with just a menu icon. However, other data indicates it is better to always use a text label alongside any icon, so I'd recommend that for menus as well.

Labels often worry product teams during early phases without translation. Luke Wroblewski, a product director

1. https://smashed.by/hamburger

at Google and writer on interaction design, has helpfully crowdsourced a very large set of translations of the term "menu" in a Google Document, which is useful for our purposes.[2] The document reveals that a huge percentage of the contributions are "menu" or something so close that most users will be able to figure it out, which is very useful for an important label.

Speaking of easy representation, the symbol that is now in most common use for representing a menu – the three lines that supposedly look like a bun and patty – even has a Unicode character (U+2630) that we can type to insert a simple glyphicon on our product masthead.

Note that this is not actually coded as a "menu" icon; it just happens to look that way. It is the Yijing (I Ching) trigram symbol for *heaven*, and screen readers will either indicate that or not read it at all, so it is doubly important to include a text label with this.

Don't go overboard with the sliding animation to make the expanded menu seem like a drawer, as this may slow down users and annoy them. In addition, some people can become disoriented or even ill because of onscreen motion. At the very least, code it to respect the OS accessibility settings when set to reduce motion.

2. https://smashed.by/menu

BENEFITS

- Drawers are highly contextual and expand directly to show more information or functionality.
- Drawers are most effective if there is only a small, fixed amount of content.
- People are used to looking inside boxes, folders, and books to find more information, and are increasingly familiar with the convention in the digital world.

DOWNSIDES

- The menu items are inherently hidden. Drawer menus are not a good way to expose info that users do not expect or know to look for.
- If there is too much content or a variable amount of content, users can miss that the additional content exists at all. Scrolling can be confusing or interfere with page scroll if implemented.
- Drawers work best if they emerge from a fixed element such as a masthead. They are not as convincing when they open from the middle of a page or a list.

- Hidden items only work when the label on the outside is clear and accurate. If they are difficult or impossible to label accurately, or some items inside are unexpected, they won't be found.

Accordions

An expanding content area is generally called an accordion, though in the past they had other names like *window shades*, and I have had teams who call them nothing specific and vaguely refer to them as "open/close areas."

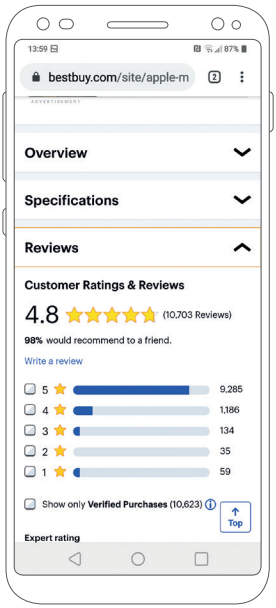
At its most basic, an accordion is a single row title that when tapped will expand downward to reveal the information within. Multiple accordions can be stacked up using the list format style, each with their own bit of content. This is similar to a tab in some ways, but there's room for an arbitrary number of category rows and much longer names.

Accordions are useful, but they are also risky.[3]

The biggest problem with accordions is that users can get lost while scrolling. This is true even on the desktop, but much more likely on mobile devices. An accordion that opens to an area that is shorter than the target screen height

3. https://smashed.by/mobileaccordions

is fine and especially useful to replace dialogs with short lists, or grids of actions or options. Users initially see a list of items and tap one to see its details. Then they can either close it or scroll past it to the next item.[4]



An accordion to reveal details on a product page.

But accordions are often used simply to tidy up a lot of content or long lists. And as users start to scroll, they can easily become lost in the middle and have no idea what section they are in.

If we customize the accordion to be something like the classic Windows tree view, then it is only a little better.

4. https://smashed.by/accordionchecklist

The connecting lines certainly help indicate to users that they're in a subsection, but the accordion still doesn't tell them what subsection they're in. And the lines can be a little dense and become hard to decipher at a glance. On the other hand, the indent levels of that tree view work very well (I always indent my accordion views). Keeping to one or two levels works well, and only the indent is needed to make it easy for people to tell how far down in the list they are. (A solution to section context is discussed below in "Floating Headers and Chyrons.")

A great way to reduce the user's propensity to lose their way is to only have one accordion open at a time: when the user opens another accordion, the first one closes. This annoys a lot of product teams, but we just need to remind everyone that it's easy to get lost so users can't possibly scroll to compare. One accordion open at a time is much more tab-like behavior, so if it were a toss-up between accordion and tab, this is much easier to justify.

In theory, the icons that control the accordion don't matter. In practice, there's too much chance for confusion. Most interactive systems are functional, so it's likely some part of your app or website will already be using the plus and minus signs to mean something else, such as add and subtract, or zoom. There are also readability issues with the minus sign; I have come across users who can't find it, confuse it with the Windows minimize control, or think it means the

partial-selection checkbox. So I always use arrows. And only arrows pointing up and down. A right-pointing arrow on a row means load a new page; down means to expand this area; up means to collapse.

BENEFITS

- Accordions work well to show hierarchies of content, such as nested lists.
- On mobile devices, where individual steps in a process are almost always taller than the viewport, accordions allow a more natural progression than tabs from one step to the next.
- Accordions allow display of additional content or functionality anywhere in the page.
- When used with a small amount of content, we can be confident users will not get lost while scrolling.
- Even larger amounts of expanded content can avoid confusion as long as only one accordion is open at a time.

DOWNSIDES

- If an open accordion is much taller than the target viewport, users can get lost when scrolling.

- If each accordion's contents is very similar to that of others, users can become confused about what section they're in. Users would need to see the title to differentiate each section, but in long lists the title would scroll off the page.

- Including all accordion content at page load can impact performance, and has SEO and accessibility implications. Even though the content is present, users cannot find it by scanning or through in-page search. And for users working without style sheets unclosable accordions can overload the page content.

Tabs

I regularly use tabs in my designs. They are terrific for switching views and especially to provide quick switches when multi-encoding large amounts of information, such as a graphical view and a chart (text) view. But for navigation? Well, the issues we see on mobile are the same as those we encountered over twenty years ago as the desktop web began to grow.

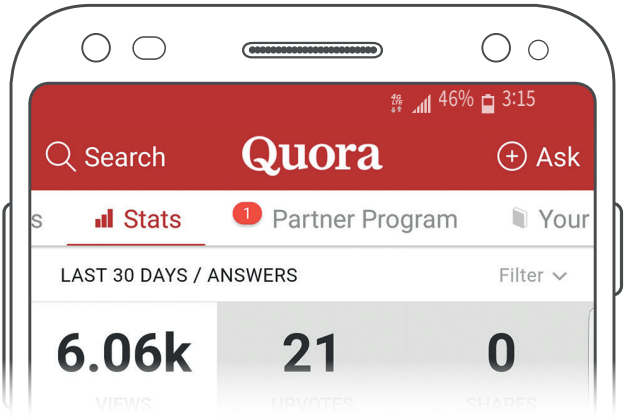
For a great example, see how Amazon tried to change its navigation when it started to grow past being just a book and music store. Amazon started by adding tabs, and then when they didn't fit, it took odd steps, like stacking the tabs.

For a while it offered an overflow button to "see more stores," a common fallback for those who promote tabs. Of course, the button opens a menu and we're back to wondering why any tabs are shown at all. Mobile just makes this worse, as the small screen means we will run out of room faster. Even with short labels, I can't fit more than three or four tabs safely. Personally, I never use tabs for primary navigation.

Some of the Amazon tab variations between 1999 and 2002.

Let's discuss the right way to display tabs. It applies the lessons from chapter 10: make interactive items apparent, readable, and afford action, and to give the user confidence.

As much as I am a fan of OS or framework defaults, tabs are another issue, like form inputs, where the lightweight, open default style doesn't work very well.



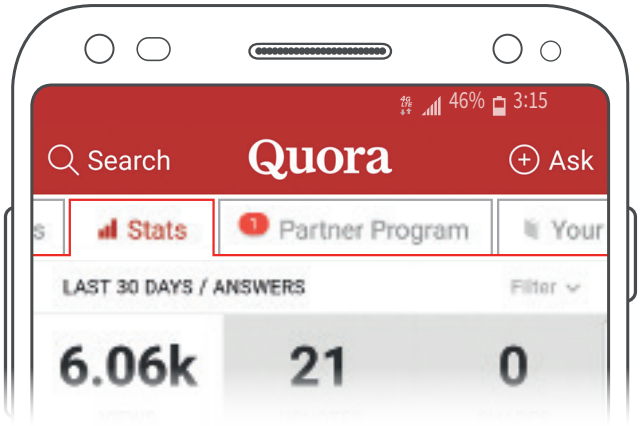
Tabs in the Quora app for Android.

Bold labels, highlighting, and underlining are good ways of differentiating the content of the current tab, but tab labels still require interpretation. Having only two tabs can be problematic: how does the user know which tab is highlighted? Even with three or more, there's often some confusion for users who can delay in identifying their position or taking action.

The original way of showing tabs in digital environments emulates the form of actual, physical folder tabs – this is the

reason we call them *tabs*. This is one case where skeuomorphism isn't bad; tabs are not an ornamental feature but take advantage of a core organizing principle.

Good tab designs work well because the tab label is contiguous with the tab's content, just as a card-stock tab sticks up above the physical tab's content. The label is also actually on the tab, so the background of the selected tab continues into the content area below with no color change or divider. When we do this, tab labels serve as very effective titles or subtitles for the content, as shown in the example I have mocked up below.



A mock-up of tab-shaped tabs in Quora.

Users expect physical laws to remain in force in their digital experiences, so the contiguous tab nature meets an existing

mental model and is easy to understand. The non-selected tabs are also understood to be attached to their content, so tapping them reveals that content.

Tabs should always be placed at the top. The bottom tab bar popularized by iOS is not perceived by users as comprising tabs, but more like a set of actions. I suspect this is due to both their lack of tab-like appearance, and the top-to-bottom nature of hierarchical content relationships.[5]

The project experiments I have conducted with more robust and tab-like highlighting of the bottom didn't solve the perceptual issue, so I simply avoid them entirely now. This works well for me as I like to use the bottom area for actions in a floating chyron, which we'll discuss in a little bit.

BENEFITS

- Properly built tabs are easily understood by their nature, with no need for special icons or labels to explain the function.

- Tabs are a good solution for items of similar prominence and importance.

- Titles can be removed from most tab content, as the tab itself is the label. Much like accordion titles, this saves vertical space.

5. https://smashed.by/intrinsicunderstanding

DOWNSIDES

- One tab must be the default, and this must be the first tab.
- Regardless of the space available in the design, the label must be short and clear. Long labels won't fit in tabs; consider an accordion instead.
- Only a relatively small number of tabs works well. On mobile phones, no more than three to five, depending on how they are labeled, will even fit. And even with larger screens, too many tabs become hard to scan and hard to find.
- Tabs work best for entire page changes only. When used lower on the page, keeping track of when the user scrolls outside the tab area can become confusing.
- When placed at the bottom or side, users will not always perceive the tabs representing equal choices, understanding them instead as discrete functions or actions.

New Pages

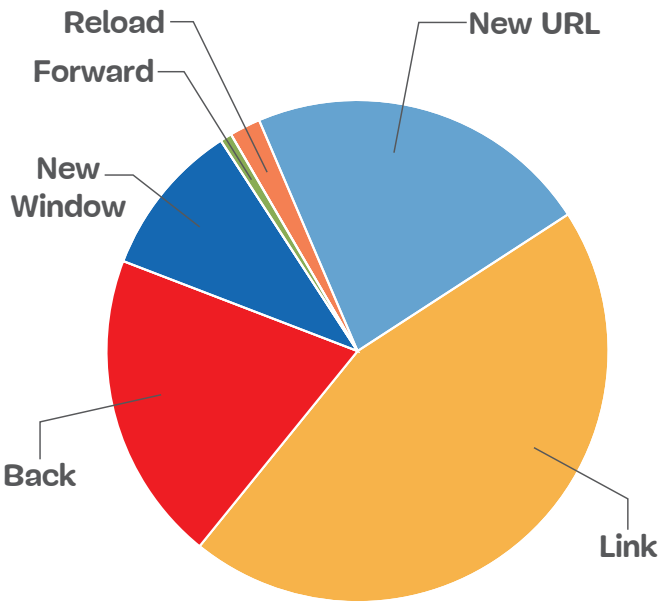
Clicking a link still performs basically the same action today as it did when pioneers like Ted Nelson were expounding on

the early concept of hypermedia. Clicking or tapping a link to load a new page works as well now as it did in the 1960s, or in the late 1990s when the web entered popular awareness.

When we're looking at a diagram of a website or an app, it might seem like there are a lot of pages. This often worries project teams, who seek to add states, to go with "one-page" solutions, or otherwise to simplify the design. But remember that users don't see the entire diagram, only the pages they view. Hub-and-spoke navigation or drilling down to pages is generally much less confusing than many of the other methods we have discussed. Take accordions, for example. A page's title can be anchored to the top of the viewport, preventing users from accidentally visiting a new page. Users know where they are and cannot get lost.

We know hub-and-spoke navigation works well from observing users over the decades. It also makes logical sense, in the same way that tabs work. People expect physical laws to work in digital spaces and start their experiences acting like digital space is real. As much as it helps discovery to expose cross-links, users seem very comfortable just finding content then backing out to change categories or tasks. And we've long known that the most used single button in browsers and mobile apps is Back.[6]

6. https://smashed.by/firefoxbutton



Frequency of use of the types of actions in web browsing.

Earlier studies, mostly of the desktop web, showed that the back button was the most used single action in browsing (the concept of clicking a link is higher, but that's among all links).[7] [8] My own product research indicates a similar rate today of use of Back on the mobile web, and in apps as well; current academic research likewise assumes Back is well used and seeks ways to improve it or adapt to more complex contexts.[9]

7. https://smashed.by/mistakes
8. https://smashed.by/revisitation
9. https://smashed.by/crosswalks

People don't use functions they don't like. If the back button was grudgingly used to hack around system issues, we'd see usage drop off, but that hasn't happened in the thirty years it has been studied. We can trust that users are comfortable with this method of navigation. But code it right. Respect the back stack and history states. Don't use fake pages that require the user to work with in-page navigation, because they won't notice it and will break their session. Follow standard, built-in OS or browser conventions whenever possible.

BENEFITS

- All users understand pages and that Back moves to the previous one.
- It is universally supported, and the default method when coding up links.
- The back button, even when in the far corner, is tapped by all users, with no issues or complaints.
- Pages can do anything and hold any content and so are the most flexible method.

DOWNSIDES

- If our product or platform or framework is already built as a single-page app or carries out processes as single pages, page changes can be confusing or cause actions that might break the process.

- If there is live information or user-entered data on a page, leaving the page would risk losing context or clearing the information the user has entered.

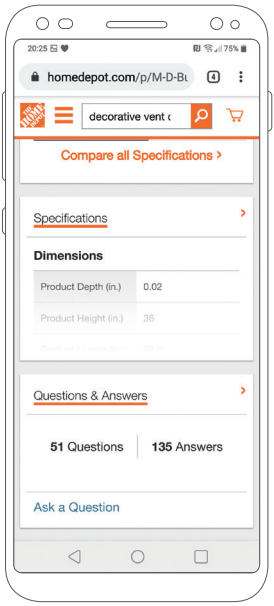
- New pages may induce new data calls, especially for connected sessions, which may reduce overall performance. This can be alleviated, however, and might not be different from competing methods like tabs, depending on their implementation.

Scrolling

As I mentioned in the previous chapter, a common question – even today – from product teams and marketing is "Where is the fold?" with the assumption we must keep everything above it and visible on screen. But we know that's essentially meaningless both because screen sizes vary and since people scroll to discover new content. Can we take advantage of that in design?

Absolutely. Amazon was one of the first to really figure this out by skipping almost all linking, accordions, and use of internal tabs on their product pages. There's no real reason a page needs to be any particular length, so we can simply add more information and functionality below.

It is easy to mix this method with others. The Home Depot product details page has summaries of all the information when the user scrolls, but accordions to reveal the full details of each of those sections.



A long, scrolling page on the Home Depot site.

Since users scroll, there is essentially no downside to using very long pages, or even one-page websites. Navigation can be provided by links anchoring to the relevant part of the page.

The only factor that prevents many teams from using scrolling is a concern about performance. It's not a real-world problem, but it is a metrics problem. Amazon is a famously analytical company so clearly gets around it – but how? By making its metrics adjust to reality.

The pages indeed take a very long time to load, but they are built so the top parts that users see first load almost instantly. The parts of the page outside the viewport can load in the background.

If built properly, the structure can also load so jump navigation works, and very clever implementations can even change the loading to make sure users who quickly click to view the last item on the page have it loaded immediately, instead of being made to wait.[10]

However, by traditional measures of performance this will be logged as slow-loading pages. If the company has trouble with changing their thinking, this can cause managers to have to defend the speed to their bosses, and they probably will not want to do this. Let's not lament the state of business today but include an understanding of what drives our

10. https://smashed.by/pagingscrolling

organization or client in the design phase, and not propose solutions that will cause internal, political trouble.

BENEFITS

- Single pages work well with a robust, well-labeled hierarchy so users understand what they are seeing as they scroll.

- Latency – the delay for new items to be loaded – is the biggest speed killer, especially on mobile networks, so performance to the end user for a piece of information can be very good.

- Simplicity is high for single pages, as users don't even need to use the back button and click, but they can just scroll up and down to find information.

DOWNSIDES

- When users are likely to take specific actions or view specific information on widely separated parts of a page, scrolling becomes less effective. Use another method to display summary information, and allow the user to reveal it then hide it – or quickly jump there and back.

- If the product's navigation is not floating, users first have to scroll all the way back up to the top – either to see key information, such as their signed-in state or cart status, or to use the navigation bar to move around the site or app. Even if users don't often use these features, moving away from them can seem like a loss of safety and prevent users from scrolling as much or for as long.

I have made several references in this chapter to items other than navigation or drill-down, such as fixed headers and list views. Let's now review some mobile best practices for display technologies. We'll especially focus on those that are more mobile and not quite the same as what is traditionally used for desktop web browsing.

Information Display

Back in chapter 6 I talked about how mobile is different from the classic model of desktop digital, and how we've over-applied and misinterpreted a lot of the standard models like the F-pattern. I followed up with a lot of the principles of how users really view and interact with content in mobile interactive systems.

But now let's finish off our discussion with some display tactics that are especially optimized for and effective with mobile.

- Floating headers and chyrons
- Lists
- Tables
- Category carousels

FLOATING HEADERS AND CHYRONS

Zones 2 and 3 from chapter 11, where I suggest we all put controls and menus along the edges, only really work well when they are along the edges of the *viewport*, not the page. If we put items at the edge of the page, scrolling will make their position change, and in many cases they will disappear. Even when just talking about labels, scrolling makes the user lose context because the header is outside the viewport.

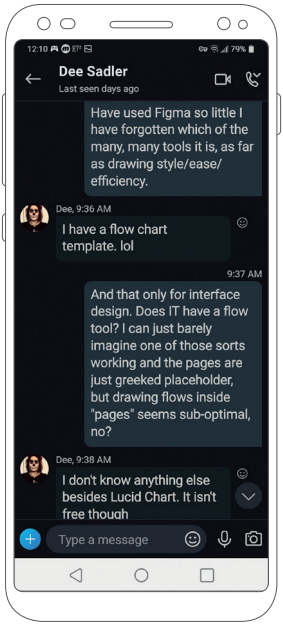
The answer isn't shorter pages – which I am often told to create on projects – but *floating* controls along the edge. These floating elements have a fixed position, right at the top or bottom edge of the viewport. Page content scrolls as usual and is visible in the large middle area between the two floating items.

The *masthead* is the title bar at the top of the viewport. It need not be just the branding and app name, but can extend to include page titles, tabs, and other information that is important to maintain context.

Content scrolling behind a fixed header and chyron.

The *chyron* is a footer, at the bottom of the viewport. However, it should never include the normal elements of a traditional website footer (one reason I adopted the video production industry term for the bar of captions instead). A chyron should remain at the bottom of the viewport only if it provides status, buttons, or control functions, but can disappear when not needed on a particular page or view.

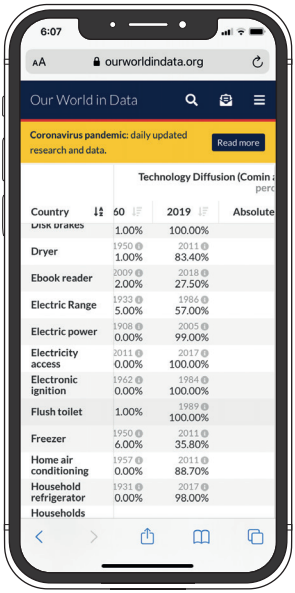
I design my apps and websites with floating mastheads so users are able to tell at a glance where they are, no matter how far they have scrolled. It also provides immediate access to the menu and any other functions, like search, that reside there.



Many conversation views, like the Skype app here, use a chyron to add to the conversation contextually, and while viewing any part of the thread.

Chyrons are also useful to provide access to contextual functions, such as submitting a form, adding a new post to the thread, or to open a drawer of options. Some uses of masthead and chyron are determined by their position: some items should be at the top and some at the bottom.

But the two areas provide us with options for design when we need to put more than a handful of elements along the edges. Remember that touch target sizes are very large along the edges. For phone-sized devices, only as few as four items might fit, so we will run out of room fast trying to cram it all into one of those.

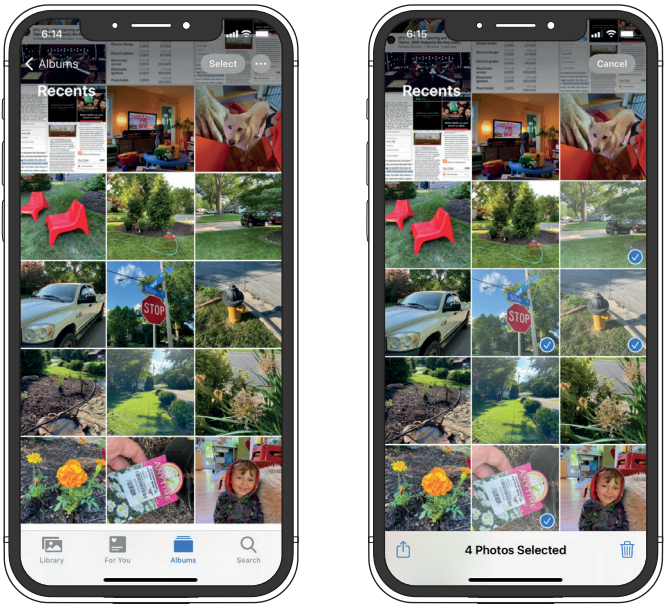


Tables in the Our World In Data website have captured headers on scroll.

As well as acting as the masthead, floating headers can capture other items as the user scrolls. A title in the middle of the page, especially for an item that is taller than the viewport, will scroll with content until it gets to the top

of the content area, where it will stop and become part of the floating header. Table headers and accordion titles are excellent candidates for capture during scroll to avoid issues of loss of context.

A similar variable visibility behavior exists for chyrons, where they can appear as soon as scrolling begins, but so far I have found no value in any of the implementations I have seen.

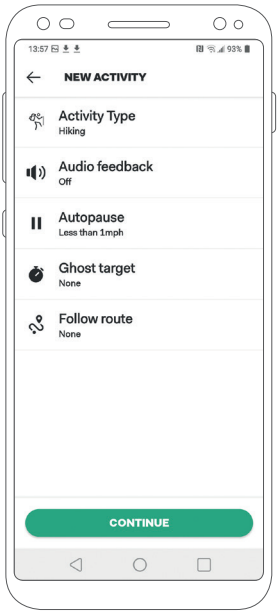


A chyron with count of selections and actions to take overlays the tab bar on the Photos app for iOS.

Chyrons can work well when based on context. One good use of them is for batch functions. When a list of items is selected, a chyron appears (or overlays the existing one) with a count of selections and the options that can be taken on those items. But there's no need to display it all the time.

Lists

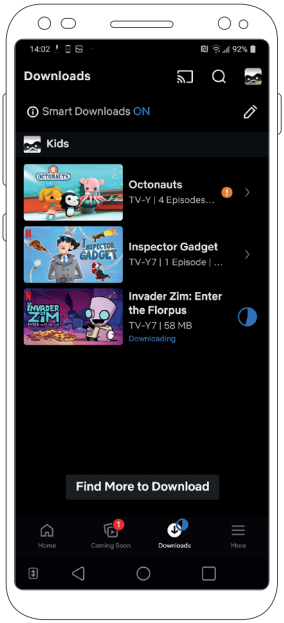
Mobile design is about lists. Look around at the apps and sites on your mobile phone (or just flip back through the



A list view is used for selection of workout features in the Sports Tracker app.

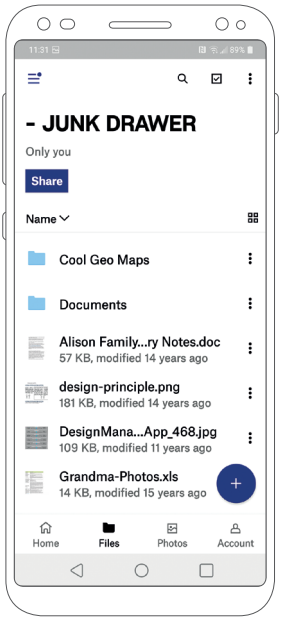
screenshots in this book) and you will see they are largely composed of list views. Certainly, there are news sources and blogging tools that are occupied with long blocks of text; there are image viewers, maps, and other display items as well. But if we are building an interactive product, what gets displayed and clicked is best served most of the time by being a list.

Once we realize that the default display mode is a list view, the easier our design decisions become. Now we can get down to designing the best possible list display for our content.



List view of videos downloaded to the phone from Netflix.

Lists are often best displayed with 1-pixel dividing lines between each row, but never vertical lines to bound them entirely, so they can breathe and we can align items all the way to the sides. Without dividing lines, users may become confused as to what constitutes a row or not be aware the row is selectable. If large graphics are part of each list row, they can sometimes serve as indicators of the top and bottom bounds of the row, as shown in the Netflix download example.



List view in classic files-then-folders desktop view in Dropbox.

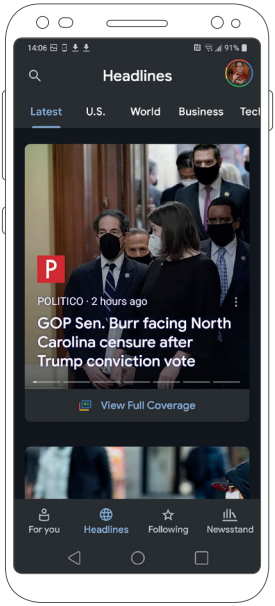
List views can be quite complex, with multiple types of content in each row, subsidiary dividing lines, and multiple icons and functions. As discussed in chapter 10, reveal arrows to the right should point to the right when selectable list items load another page, but when they do not, no arrow should appear.

Try to make list contents contain predictable empty space, even if it is small and irregular. The gap provides users with a place to feel comfortable scrolling, as mentioned in chapter 7.

When testing the layout of lists – to find out if there's room to tap and whether hands might cover critical info – remember that people will mostly try to tap the first few letters of the text label for a row. Not everyone, and many will miss, but it's a good guideline to start with.

CARDS

As a deeply analytical type, I find cards to be basically stylized lists. Instead of rows immediately adjacent to others, each row is styled as a box. Much like tabs are derived from the look of file folders, cards are similar to Post-it notes or index cards taped to a wall.



Parts of Google News use a card view with one card for each story.

I am not excited about the value of cards, mostly because of the wasted space between each card and the edges taking up room and being "trapped," which can stop users from scanning smoothly between the items.

While in principle the separation into cards allows more internal formatting, in practice I only find myself using cards instead of other design solutions not for any solid design reasons but because the client's design principles or technical framework require it.

Tables

The primary alternative to the list is a table – and I have a terrible secret to tell you: in most app frameworks, pretty much everything is a table. The right-hand column of reveal arrows stays aligned because it's a table column. To make a list view, we just don't show the table's vertical borders. In many design systems, we even label the component a *List Table* and define the differences between simple list-like displays, complex table-like displays, and when they get functional.

In reality, a huge percentage of the designed elements on pages overlap. A text list with icons is a table with two columns. A text list with two icons for each row is even more a table, now with three columns. When does it become a "table"? Hard to say, but once we start thinking of lists and tables as two ends of a continuum, it is easy to just pick and choose features as needed.

That can also work on the web from a design point of view, even though the technologies do not overlap. And they really do not overlap, because tables are all but banned in some places I have worked. In the early days of the web, there was no attention at all paid by the standards bodies and the browser developers to how content was laid out. No thought

was given to separating content and style because there was no thought given to style at all.

Early web designers used tables to organize their web page layout. Which is unsemantic, of course, so once the semantic web move to meaningful elements for each item really nailed it, we were not supposed to use tables anymore.[11] Except that this went too far and now developers code tabular data into web pages using `div`s and CSS.

Tabular data, however, should almost always be displayed using a plain old table. Using tables properly, for the display of data, is a good and necessary action. However, small-screen tables are an entirely different matter and have raised this issue again.[12]

The various small-screen table tactics[13] and their key problems are, in brief:

- [x] Allow horizontal scrolling: even with clever bits like locking the row titles, this is just confusing and users never put it to good use.

- [x] Convert tables to images: seriously, I see this a lot. There's nothing good to say about it.

11. https://smashed.by/tablelayouts
12. https://smashed.by/datatables
13. https://smashed.by/mobiletables

- ☑ Convert tables to graphs: I have already stated how multi-encoding is good, and if the tabular data will work as an easy to understand graph, great! But only letting mobile users see a graph while desktop users get tables is weird and a bad idea.

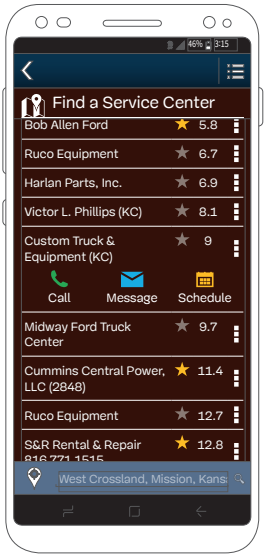
- ☑ Responsive tables: I mean the style where each row is turned into a tiny table of its own. Each column header becomes a row label. This meets the requirement to display the content but entirely misses the point of a table. Users cannot compare row by row at all, so we might as well not even have a table at all. These never work.[14]

The other tactic I see, usually hand-in-hand with horizontal scrolling, is letting users select columns to show and hide.[15] Usually the table starts way too big and, "if they want," users can cut it down to fit. Of course, this misses many points, but worst of all is the project team admitting that users don't need all those columns. So the real tactic is: only show what is needed.[16]

Lots of tables repeat information that is the same in every row, show unimportant information, and show verbose content the user doesn't need. Let's say a user is searching for the nearest location of a shop to their house. They don't need the address yet, just the distance. Distance as a num-

14. https://smashed.by/responsivetables
15. https://smashed.by/tabledesignguide
16. https://smashed.by/ordereddata

ber is a very small and easy to use field, and we can very clearly show the list in distance order. Parsing a column of addresses would be much slower and less effective.[17]



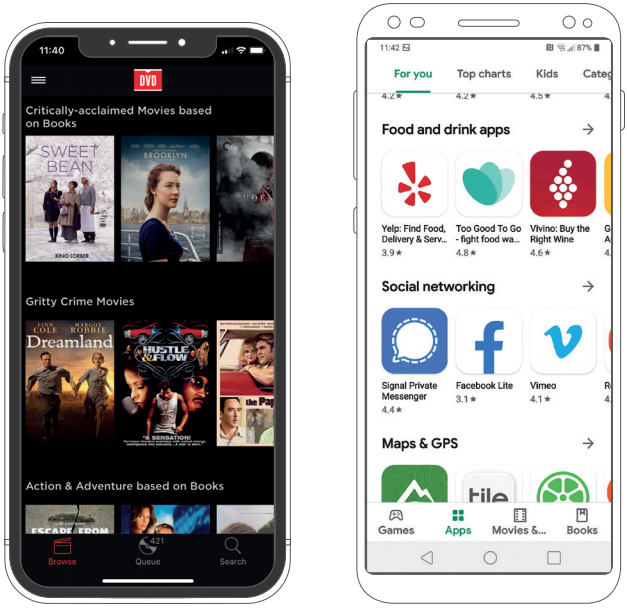
A table displaying only the information absolutely required to make decisions, which expands to show options when tapped.

Think about how people will use the content, and very often we'll find there's a process. They can select one to see details or take action. Then let them get a new page, dialog, or accordion to get that extra info. A table small enough for mobile and simple enough for users on all platforms will almost always naturally emerge.

17. https://smashed.by/typographytables

Category Carousels

A lot of content discovery systems are moving to a somewhat new display method that it seems no one is discussing, so I get to name it. I'll call it the *category carousel* to differentiate it by purpose and design from the generic slider or carousel too often used for marketing banners.[18]

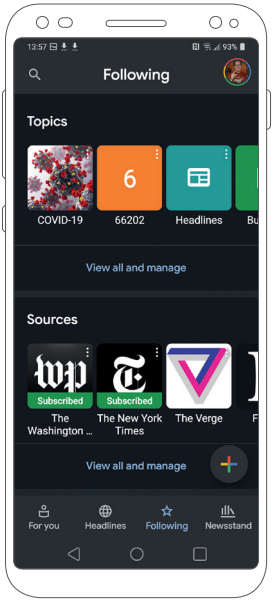


Netflix showing box covers, and the Google Play Store showing app icons and additional information in category carousels.

18. https://smashed.by/carouselslider

Envision a list where each row is a category with a label. Instead of the content being fixed to fit inside the width of the viewport or wrapping, it overflows. Scrolling left or right will show more content within that row. Scroll up or down to see other rows, and then go explore their content, row by row.

These are increasingly common in media such as streaming video as those are visual. Typically, a title will have a "box cover" as though for sale on a shelf, a visual representation with a readable title. These do not work well with pure text



A category carousel used with icons for news services.

content but can suffice adequately with app icons, brand marks, or other identifiers.

Make absolutely sure that the rightmost item "falls off" the edge of the viewport. An overdependence on gridded design makes many category carousels not work correctly as the designers make it all fit perfectly. No one knows there are more items if they all fit, so use the old trick of partial showing to indicate that. Partially shown items invite users to scroll. Never let these mix with marketing carousel banners with automatic scrolling.

Each card in the carousel should always be the same size and format. I strongly encourage you to apply text labels for accessibility reasons. Even for sighted users under ideal conditions, the brand mark or video box cover is functionally unreadable. Since the category title is above, it seems to work best to lead with the image and put all text labels and other data below that.

Make sure to support all input methods. Don't require users only to gesture scroll, but also allow tapping to scroll. Better yet, offer them category pages. Clicking something like "More" on the row header could load a whole page of these as a grid or list.

I am in favor of full exploration, but many systems have technical or performance limitations and can only show a subset of the items within each category. If that happens, be sure to have a prominent "More" function at the end so users don't think your product only offers ten kids movies. If you can make it fit, be sure to show a count like "See all 14,890 kids movies."

The value of this pattern is immediate drill-down at the same time as category exposure. Don't make people click on a category then have to select the specific item they want again.

In chapter 11 we presented a way to create consistent, touch-friendly information design based on what we know about how people interact with their mobile devices. In this chapter we went into greater detail to discuss a variety of UI and interactive components you can use to achieve the 1, 2, 3 principles and their best practices for design.

CHOOSING A DRILL-DOWN METHOD

- **Pop-ups** are quick and easy, but not as contextual as we think, and they cannot handle large or complex amounts of information or interaction. Pop-ups are overused and will often be dismissed without reading.

- **Drawers** are very contextual, and when the proper information is loaded they are extremely effective. Have no fear of using hamburger menus. But don't rely on drawers for discovery, or for large amounts of information or long lists.

- **Accordions** are a great way to expand a list to show details right where the user needs them, or can show steps or hierarchies without loading new pages. However, we have to be careful with design to alleviate problems of getting lost inside lists longer than the viewport.

- **Tabs** are well understood and work well to display equal-weight content. But space in mobile devices is horizontally limited more than anything, and large numbers of tabs are hard to read regardless of screen size.

- **New pages** can do anything without limits, except for loss of context or previous entry. Users understand the back button, so hub-and-spoke navigation works very well, as long as our whole product works that way.

- **Scrolling** single pages work very well because people like to scroll up and down, so there's no need to load and reload content. But they only work if we can tell a story and have a well-defined and well-designed hierarchy. Scrolling also isn't the solution for many processes and don't work so well without floating navigation.

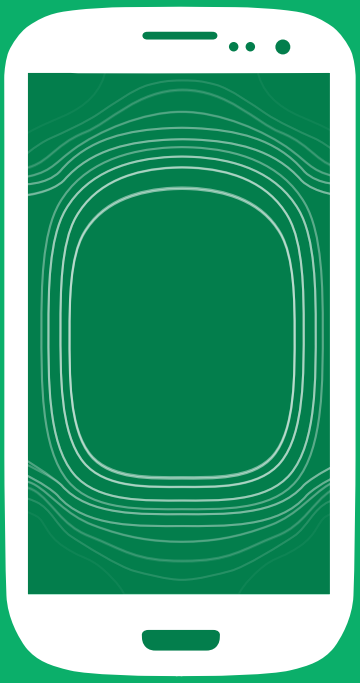
CHOOSING MOBILE-FRIENDLY DISPLAY METHODS

- **Fixed mastheads and chyrons** make the context and key navigation, function, and action items visible at all times. They also help with several of the other progressive disclosure methods by preserving internal context as users scroll.

- **Lists** are what the mobile interactive world is made of, because they are simple, easy to use, and easily expanded to cover many use cases, from display to selection.

- **Tables** are slightly more complex lists, allowing us to show multiple columns of actions or functions. Don't let tables get out of hand; always show only what is absolutely needed.

- **Category carousels** are an up-and-coming way to allow discovery, disclosure, and selection of multiple categories in one screen, while leveraging mobile-native methods of tap and gesture.



CHAPTER THIRTEEN

Practical Mobile Touchscreen Design

CHAPTER 13

Practical Mobile Touchscreen Design

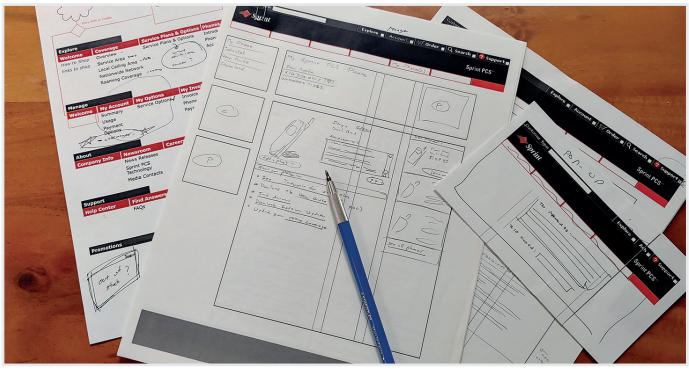
I want to conclude by discussing briefly how I use the information throughout this book as part of my design approach and design process, and revisiting and summarizing the major points.

I moved from print to interactive design before user-centered design was a common term and years before someone put the then-new chutes-and-ladders design process chart on the wall.[1] I developed my design process over time, by reading, trying things out, failing massively here and there, and learning from my mistakes.

While in graphic design school, we weren't given a process overview. I believe the thought was that you'd go to work for an agency, and the account manager would talk to the client, give you a *design brief* summarizing the needs and constraints, and then you would just come up with ideas. But in the digital world I rapidly learned that timelines are short, so options are rarely needed, and I learned to quickly sketch solutions in meetings. When I moved on from small agencies to a large corporation, we built up a large team with analytics and usability testing, so I knew this usually worked.

1. https://smashed.by/throwback

However, when we did a complete, end-to-end redesign of the website, it got entirely out of hand. I implemented what we'd now call an information architecture design, laid out a template, and then met with stakeholders for months sketching their ideas.



Drawing designs, page by page, in the early 2000s.

Just a few weeks in, it was clear this was not going to be a long-term solution. Once it finished and I had drawn every page in Photoshop, I set off to create a better process. I read all I could find about design and engineering systems and the philosophies of design at the time. The process I came up with wasn't too far from what we consider best practice for user experience today.

The main thing I have always remembered from this work, as a drawing-oriented person, is that we must hold off as

long as we can from drawing. First listen, research, write down what you find, develop ideas in words, bullet lists, and boxes. And only once everything is well understood, start to actually draw solutions.

1. Understand Your Audience

Your next project will probably start out with a list of requirements, but those tell us nothing about who is using the product and how it might help improve their life, or even just how it makes your organization more valuable to them.

First we need to identify who will use the product, on what systems, in what environment, and in pursuit of what actual end state or goals. The best way to find this out is not to ask others, but to watch how people work. Get in the field and see how actual users (or similar types of people) interact with the class of systems you are designing for – and how they don't. Only by watching people in their natural environments will you understand where your system ends, how people use other technology, write codes on Post-it notes, or just turn and talk to other people.

Before you go out into the field though, sit down and read or research a little from the comfort of your desk. I have set out a lot of basic data about how people work, or with what devices, and you can find out a lot more by looking at

web analytics, understanding the jobs behind the job titles, reading user complaints, and much more.

EVERYTHING IS NOW MOBILE

Mobile phones are by far the most used connected, interactive devices. I shared a chart of device class usage in the introduction ("Everything Is Now Mobile"). Keep up to date on general trends by regularly checking reliable sources. Scientia Mobile releases its mobile overview report (MOVR)[2] regularly, full of statistics and insights on mobile usage, based on actual clicks through their industry-leading WURFL device detection repository.

The statistical aggregator website Our World in Data[3] is one of the best, most comprehensive, and easiest to use resources for all sorts of statistics. Its page on internet access is very useful and, unlike too many sources, is not specifically about one country.

In chapter 1 ("Defining Mobile Devices") I discussed how PCs themselves are becoming mobile, with most now incorporating touchscreens. Many computers are not computers at all: the best-selling devices are Chromebooks, based on the mobile-first Android operating system. These are good guidelines to start with, but you should try to find usage rates by industry trend or analytics for your specific targeted regions and users.

2. https://smashed.by/movr
3. https://smashed.by/internetdata

KNOW WHICH MOBILE DEVICE

I defined a lot of device classes in chapter 1, but mostly discussed that people use many different devices. In chapter 5 ("Finding Out How People Hold and Touch") I discussed how device diversity reflects human diversity. Don't assume, stereotype, or judge; find out what people really use and design for that, and for everything likely, not just your favorite phone.

Also don't forget that almost half the planet – including a lot of people in the West – doesn't have a smartphone (see introduction). Depending on the region and type of users, you may be missing a lot of them by assuming iOS or Android only.

UNDERSTAND THE USER'S ENVIRONMENT

We've seen how people use mobile devices in new ways, to connect all the time instead of only sitting in an office (chapter 1). Be sure you don't make any assumptions about where, when, and how people interact with your product.

It's important to be aware of overall trends in how people use different devices in different ways (chapter 5). Remember that larger devices are used farther away from the eyes and are more often landscape oriented. This is all critical for design later on.

I shared some information about how interaction varies based on where the user is or how their phone is configured (chapter 9, "Phones Are Not Flat"). People have vastly less accuracy when moving or carrying items, and difficulty touching the edges when protective cases are used. Do you know your users' environment?

Perform Field Research

Get out to the field and be prepared to gather information outside of your original plans and expectations.

Ethnography is a whole area of study in which people earn college degrees. If you have the budget, consider hiring one. Jan Chipchase was a traveling ethnographer for Nokia when it was at its peak, and he provided a lot of good information to the community. He has more recently published a comprehensive book on the subject, *The Field Study Handbook* (2017), which I highly recommend.[4]

Jessica Weber and Jon Cheng's UX Magazine article "Making the Most of Ethnographic Research" has a good overview of the tactics needed to do good ethnography, specifically avoiding many pitfalls in areas like recruiting.[5] There's also my own article about field research and pragmatic methods to squeeze ethnography into your visits and usability tests.[6]

4. https://smashed.by/fieldstudy
5. https://smashed.by/ethnographic
6. https://smashed.by/fieldusability

Define the Audience

Take all the info you know and start defining who the audience will be, then share that with everyone in the project team.

Personas is the buzzword around this, but they are often and easily misused. UX Studio is a Hungarian design studio that blogs about the pros and cons of personas. Anikó Kocsis' post, "User Personas: Traps and How to Overcome Them," contains helpful tactics to make sure you create good and useful definitions, and do not just slot people into marketing segments.[7]

Design ethnographer Kelly Goto is one of the smartest people on this topic. Her slide deck on understanding users, in the guise of accessibility and aging populations in this case, includes some great details on how to analyze and understand the audience without falling into many of the traps of personas.[8]

2. Understand the Technology

In some ways understanding the technology is an offshoot of understanding the audience and their devices. But far too often there's a focus on screen size and nothing else. Make sure you understand the technology involved, from

7. https://smashed.by/userpersona
8. https://smashed.by/beyondusability

touchscreen to radios and cameras. If your product relies on a particular technology, don't make any assumptions, but go find out how it really works. I have found over time that everything is always slightly more complex than you think it is. Recursively, as deep as you go.

THE HISTORY OF COMPUTERS AND SMARTPHONES

While it concludes before we reach anything really about mobile, most of the documentary "The Machine that Changed the World" (1992) is something I believe everyone in technology should be required to watch annually.[9] Lots of early computers were no such thing but, rather, tabulation machines – and that matters. (Note: don't accidentally watch the documentary or read the book of the same name about the development of lean systems at Toyota.) I try to keep a live copy on my YouTube channel.

Understanding how we got to where we are today is not just a matter of respect, but it also helps you understand the current environment better. Why are smartphones the way they are today? There are lots of bits and pieces of history out there, and YouTube videos are a good way to see how early concepts and devices work. I have gathered a number at the 4ourth Mobile channel.[10]

But for a single narrative, the first few chapters of Elizabeth Woyke's *The Smartphone: Anatomy of an Industry* (2014) pro-

9. https://smashed.by/machine
10. https://smashed.by/4ourthmobile

vide a good overview, including a retrospective of the legal actions that drive too much of the industry. The manufacturing sections are also interesting, and a reminder that you can't just write code, but that "hardware is hard" and also drives a lot of our behavior.[11]

TOUCH TECHNOLOGY

If you're designing for any specific device instead of smartphones generally – automotive controls, point of sale terminals, seatback entertainment, and so on – don't make assumptions about how touchscreens work (chapter 2, "The History and Technology of Touch"). Find out which technology is used and then look for a primer and ask detailed questions about the hardware your design will be appearing on.

Since you are probably designing for a device such as a smartphone or tablet with a capacitive touchscreen, understand the details of how that works, what it does, and what it doesn't do (chapter 3, "Capacitive Touch"). Multitouch and pressure sensing are not what they appear to be, but you can take advantage of them if you understand the underlying technologies.

While I mostly talk about the contact patch in terms of how it impacts users (chapter 7, "How Fingers Get In the Way"), be aware of how multitouch screens sense contact patch size so it can be used as a proxy for pressure. And

11. https://smashed.by/industryanatomy

recognize how hardware design can influence parallax to impact apparent accuracy.

PLATFORMS

Decide on platforms as a group: even an "app" is not that easy and simple a choice. Understand the technologies, constraints, and values each one offers. Don't assume you are building an app or a website, but think critically about it. I discuss the pros and cons of various app technologies in a UX Matters article, "Mobile Apps: Native, Hybrid, and WebViews."[12]

OTHER INPUTS

Remember how many devices are touch dependent, but also remember that computers come with keyboards, and mouse or trackpad to point with as well (chapter 2). And remember that Apple wants people to use the iPad as a computer so sells keyboards and trackpads for those as well.

While around 15% of the world's population, or about billion people, suffer from some sort of identifiable long-term disability, almost anyone can suffer a short-term injury, be blinded by glare, be unable to hear over loud traffic or machinery, have a hand occupied controlling a small child, or any number of things (chapter 9). People use alternative

12. https://smashed.by/mobileapps

methods such as the keyboard control or even accessibility input options to allow them to interact anyway.

Among the best current resources for understanding universal access and accessibility across platforms is Microsoft. They have a resource center full of information, tips, guides, articles, and research that are very much worth reading.[13]

MOBILE NETWORKS ARE NOT WI-FI

Remember that Wi-Fi is just wireless Ethernet, but mobile networks are very different, with lots of additional signals for coordination and security, and inherently high latency due to the distance the signal travels. Testing your app or website on Wi-Fi won't accurately reflect how it works on mobile networks and can encourage bad coding and data design practices.[14] This may be changing as 5G networks roll out, because physically they are more like Wi-Fi networks.[15]

Always remember that "airplane mode" is not a feature but a way of thinking about how to design for poor or variable network connectivity, which everyone encounters.[16]

LOCATION IS NOT JUST GPS

Providing location is not simply about turning on the GPS. Aside from the *fine* accuracy of satellite navigation using

13. https://smashed.by/microsoftaccessibility
14. https://smashed.by/mobilenetworks
15. https://smashed.by/mobilevswifi (PDF)
16. https://smashed.by/beyondairplane

at least two systems – GPS is only one – there are issues of speed, power consumption, and locational privacy. There is a *coarse* location, which provides plenty of precision for news, sports, and weather, without risking privacy violations.[17]

The general advice has often been that only individual data can be exploited; as long as things like location data are anonymized, you are safe to share it, especially in aggregate. But that's not always true: in early 2018 it was revealed that Strava fitness tracker data could be used to find the location of isolated, supposedly secret US special forces bases.[18]

3. Collaborate and Share

I haven't written about how to build a team, and I've only touched on how to make strategic decisions and set project-level goals. But these are critically important. Pursuing the wrong goals will never turn out right, no matter how well you execute.

The old derisive term of "throwing it over the wall" for design applies at every step of the process. The design team can't accept vague hopes and dreams from executive PowerPoint, but need to be involved in building and detailing them out.

Project teams need to help create requirements and success measures. Share what you know not only from design

17. https://smashed.by/understandinglocation
18. https://smashed.by/strava

solutions but also about the trends in how people work, or actual data gathered about your customers. If you don't tell everyone else on the team, how will they know?

UNDERSTAND THE ORGANIZATION

Read your company's principles. You'd be surprised how often they are forgotten, and how often there's plenty about being collaborative, being one global organization, and being customer-centric. If your organization embraces some higher-level process, read that and think about it. Six Sigma, for example, aligns very well with UX design principles. The first of the seven principles is "Always focus on the customer," and others include "Get buy-in from the team through collaboration," and "Make your efforts systematic and scientific."[19]

CREATE DESIGN PRINCIPLES

There are many lists of heuristics and design principles. I have my own mobile-specific ones.[20] These principles are not in this book because while they are a good starting point, you need to set your own that are specific to your project. Remember these are not objectives but the design principles you always fall back on to help achieve those goals.

Whatever you come up with, build it into your process. Cascading design means you technically enforce the

19. https://smashed.by/sigma
20. https://smashed.by/mobileprinciples

design, and hopefully the code, to rely on higher-level definitions of style, widgets, so cannot easily deviate and have one-off pages.[21]

TEAMS AND GOALS

I cannot recommend highly enough everything that Christina Wodtke has to say about teams and product strategy, such as creating objectives and key results (OKRs). Just read her website and then buy her books.[22]

There are many, many discussions of process and methodology, and I have written up my own, based on my experiences managing several teams and mostly working within the process at dozens of client organizations, trying to optimize the product design on hundreds of projects.[23]

While there are many business books that are a bit like self-help books or extrapolate a single success, Anuj Mahajan's *The Billion Dollar App*, on strategizing and designing for mobile, might help pull together many of these threads.

DEVELOPMENT METHODOLOGIES AND PRODUCT DESIGN PROCESSES

Have you actually read the Agile manifesto? Many of us have undertaken lots of training that waters it down or em-

21. https://smashed.by/cascadingux
22. https://smashed.by/eleganthack
23. https://smashed.by/craftfordigital

braces the ritual without understanding. It turns out there's a lot more than "only working code counts." The very first line of the twelve principles is: "Our highest priority is to satisfy the customer..."[24]

Product design shouldn't have much to do with development methodologies, but it often will, so you have to help carve out a way to be useful. I am deeply opposed to design-every-two-weeks, though you may disagree.[25]

WORKPLACE AND REMOTE WORKPLACE TIPS

UX resources are massively underfunded, with far too many individual practitioners among a team of dozens or hundreds of developers.

After some questions lately, I wrote up my tips on carving out a space for your work, and creating the right environment for design-centric product development.[26] There are some special notes on mobile issues, and how to move your existing team from traditional desktop digital into the mobile space as well.

More people than ever have become remote lately, and this is likely to continue even as many offices reopen following the COVID-19 pandemic. I have mostly worked remotely for over a decade, and in 2017 wrote up much of what I'd

24. https://smashed.by/agilemanifesto
25. https://smashed.by/productdevelopment
26. https://smashed.by/onboardingyourself

learned, specifically about working as a remote designer for digital products.[27]

4. Design from the Outside In

The entire process I am outlining here is about designing from the highest to the lowest. Don't decide you have enough data and jump to UI design. Our industry is skipping a lot of steps, and design tools are almost entirely about UI design.

Instead, tie each step to the next, and at this point create a framework to support the eventual design so it is consistent and appropriate for the content, functionality, strategy, and audience.

INFORMATION ARCHITECTURE

The Polar Bear Book, as it's often called, is at the core of this practice area and belongs on your bookshelf: *Information Architecture for the World Wide Web*.[28] Now in its fourth edition (2015), you can trust most anything that Peter Morville and Louis Rosenfeld say on the topic.

To get going within minutes, or as a regular refresher, Dan Brown's eight principles of IA are useful to keep in

27. https://smashed.by/remotecollaboration
28. https://smashed.by/informationarchitecture

mind always.[29] For a broader view, and in a much thinner volume than the Polar Bear Book, Peter Morville's *Intertwingled: Information Changes Everything* (2014) is a very philosophical discussion of the whole field and concept of how we handle information.[30]

Christina Wodtke is another whom I also suggest you listen to on these topics, but a good start is to get her book *Information Architecture: Blueprints for the Web* (second edition, 2009)[31] and put it on the shelf next to the polar bear. But be sure to read it first.

INFORMATION DESIGN

I summarized the concept of information design as sense-making (chapter 11, "1, 2, 3: Designing by Zones") and applying that as building reusable structures or templates to help make sense of the organization of the information and functions of your digital products.

Much of the way I frame information design – and how I have long thought of design in general – comes from this great anthology of writings collected by Robert Jacobson, called simply *Information Design* (1999).[32] It is still one of my favorite books on any design topic, not least because the cover is a diagram of its contents.

29. https://smashed.by/iaprinciples
30. https://smashed.by/intertwingled
31. https://smashed.by/blueprints
32. https://smashed.by/informationdesignbook

In a great article, "The UX of LEGO Interface Panels," George Cave frames the overall design of things and explains and codifies structures, methods, and approaches to designing by analyzing fifty-two "2×2 decorated slope" Lego control panels.[33] Cute, sure, but also really well done and informative. By being divorced from the websites or apps we use every day, it's easier to grasp the basic concepts.

Bill Buxton reviews a lot of methods to think about and get design on paper and cardboard in his guide to *Sketching User Experiences: Getting the Design Right and the Right Design* (2007).[34]

While more narrowly focused on screens and mostly mobiles, I discuss the principles and practices of using box-based design, with index cards and sticky notes, and how this level of design artifact creation fits into the design process.[35]

PROGRESSIVE DISCLOSURE

While this is deeply foundational stuff, it's not talked about openly enough anymore. For a discussion of progressive disclosure with some clear examples, see *The Universal Principles of Design* (revised edition, 2010),[36] and for another point of view see Joe Natolis's blog post, "The Power of Progressive Disclosure."[37]

33. https://smashed.by/legointerfaces
34. https://smashed.by/sketchingux
35. https://smashed.by/adaptiveinformation
36. https://smashed.by/universalprinciples
37. https://smashed.by/progressivedisclosure

The progresive disclosure methods most applicable to mobile were discussed in chapter 12 (“Progressive Disclosure”), as well as how to choose the most effective for your needs.

Choose a drill-down method to allow the user to find more information as they click or scroll, and choose a display method to show key information on the page.

5. Design for Touch

Of course, the whole design is for touch or you wouldn’t be reading these tips, but now that you’ve sketched down to the template level, recheck that you are following all the guidelines to make sure that your design is truly appropriate for portable touchscreen devices.

ACCOUNT FOR HOW PEOPLE HOLD AND TOUCH

Remember that phones and tablets are not held in any one way. People vary, and shift all the time, so design for every way people hold them (chapter 5). Orientation can also vary, but start with basic assumptions that phones are used more often vertically (portrait) while tablets and computers are more often landscape (horizontal).

DESIGN FOR TOUCH ACCURACY

The center is most important (chapter 6, "Touch Accuracy and the Center-Out Preference"). Place key actions in the middle of the screen, with at least 7 mm between tappable items like lists. Along the edges, make selectable items at least 12 mm across.

DESIGN FOR REAL FINGERS

Fingers are opaque, so make sure targets are large enough in at least one dimension for users to target it, and be sure to show them when the tap was successful (chapter 7). Users tend to tap the beginning of a text label or on the most prominent icon; to avoid accidents, plan for that and do not crowd these areas. Users like to scroll in empty areas, so keep comfortable spaces in tables, and use left-aligned text in lists so there's a little bit of space for them to scroll in.

DESIGN FOR IMPRECISION AND MISTAKES

Don't default to minimum sizes. Instead, make touch targets as large as possible, always using whole rows, buttons, and areas (chapter 8, "Imprecision and Probability"). Never make the interaction just a word or icon. Space out dangerous or destructive actions, and provide undo or other processes to allow users to correct mistakes. Avoid using guard or "Are you sure?" dialogs.

CONSIDER THE PHYSICAL DEVICE AND THE ENVIRONMENT

Remember that users are interacting with your digital experience in the real world (chapter 9). Be careful relying on sound, vibration, or transient messages as users may be in loud environments, or simply be distracted or doing other activities.

Effects akin to color vision deficiencies (color blindness) result from glare and odd-angle viewing. Design every interface to work without color.

Accidental double-clicks, drags, drops, and selections are common and unavoidable. Design to alleviate accidents by using a single action button, displacing confirmation actions, and assuring all actions are reversible.

Don't assume the whole screen is yours to design. Investigate the edge gestures or other items the OS or browser takes away from you, such as the Safari menu bar.

PLAN FOR PROGRESSIVE DISCLOSURE

Hand in hand with the design of the information architecture and the information design should be the interface and interactions used to let users tap, scroll, swipe, or otherwise find more information (chapter 12).

While your strategy, architecture, and information design should be as universal as possible across platforms, don't let that make users of any one platform struggle. Use the proper disclosure methods to account for small screens or to work best with touch and gesture on phones and tablets.

6. Have a Conversation with Your Users

Remember that people only touch what they see (chapter 10, "People Only Touch What They See"). Users who don't understand the interface won't interact with it. It's not much of an interactive product if no one interacts with it.

Don't tell your user to do things. Don't order them about or chide them, and don't assume they understand your process and speak in jargon and code. Instead, design the process from the ground up as though you are having a conversation: when they click, or type, or gesture, that's their input to you; then your system responds, and so on until you've solved their needs – or become the trusted friend they refer to multiple times a day.

CONVERSATIONAL DESIGN

While this is a long-standing concept particularly espoused by content designers, Erika Hall's *Conversational Design*

(2018) brought it to the ultimate conclusion as an overall design principle, not just a way to do the words and pictures.

Writers for digital have long done a lot more than writing, but it was during her work for the UK Government Digital Service that Sarah Richards and her team invented the discipline of content design. Her book, *Content Design* (2017), is a great start.

PEOPLE DON'T READ

Always remember that people don't read. They read long-form articles, of course, but when it comes to labels, instructions, or tooltips, most don't read. They may only scan as little as the first few letters of a phrase, so be sure to get to the point and avoid repetition.

As is often the case, the Nielsen Norman Group has some of the foundational research on this, and the same has been proven over and over again on products across digital domains.[38]

It is best to simply embrace that no one wants to read about how to use your product – they simply want to use it. Software developer Joel Spolsky has a delightful blog post about this principle, "Designing for People Who Have Better Things to Do with Their Lives."[39]

38. https://smashed.by/reading
39. https://smashed.by/respectingusers

Torrey Podmajersky's book *Strategic Writing for UX* (2019) is a complete guide to writing effective and informative instructional text.

People read words, and words are made of letters, so understanding typography is important for any designer. My favorite work on this is Ellen Lupton's *Thinking With Type*. Anything it doesn't cover or skims over is covered in James Felici's *The Complete Manual of Typography* (second edition, 2012).[40] I have both readily at hand on my bookshelf.

DESIGN FOR MOBILE VIEWING

Make sure all interactions are designed to attract the eye, afford action, are readable, and inspire confidence they can be safely tapped (chapter 10). Make interactive items appear as common and recognizable items: buttons, tabs, and inputs.

Angular resolution changes how large things appear, based on how far away the user's eye is. Since different devices are used at different distances, you need to know, either from direct research or by using the device usage classes I listed in chapter 10, which sizes to start with.

Remember that people and situations vary, so multi-encode all content to assure everyone can read it. For example, never just use icons, but add text labels so there is a fallback.

40. https://smashed.by/manualoftype

Use hierarchies of design – position, size, shape, contrast, color, and form – to arrange items by affinity and prominence (chapter 11). Contrast is always more important than color. Aside from what I have said in this book, I also wrote about this in the guise of discussing dark mode and specifically for mobile.[41]

Users who think they can't safely interact will wait till they are in a better environment but might never come back or lose overall confidence in your product. Make sure interactive items are bound (have borders or containers) and assure that items are clearly separated to increase confidence.

7. Plan for the Unexpected

Resilience is usually defined as the ability of a system to absorb disruptions without tipping over to a new kind of order. A building when exposed to too much lateral ground movement settles into a state of rubble on the ground unless it is designed to resist the disruption of earthquakes adequately (chapter 8).

There's a lot of engineering best practice around this, but I like to remember that for any system I design, the biggest constraint and the most difficult thing to change – by far – is users. They will do unexpected things, but

41. https://smashed.by/dark

never try to change them. Instead, accept the complexity and unpredictability.

Never ask people to type or select data that the computer already knows. Never require input in computer formats when it's easy to parse from arbitrary – or human – formats. Phone numbers do not need hyphens.

Don't make people think about what the computer is saying. Display data in human-centric ways. Engineers must not write error messages because only they will understand them. Avoid error conditions caused by unexpected user input. Constrain, parse, or just live with the data. Do not gripe at users because they are not computers.

YOU (PROBABLY) DO NOT WORK FOR APPLE, AMAZON, OR GOOGLE

Don't base your business model or tactical decisions on what they or your competitor does. Likewise, don't fear the competition will steal from you as they generally cannot, because they aren't you. The best way to induce failure is to work against your systems and your customers. First, consciously know your business model.[42]

I also have written about designing with the true nature of your product or the platform in mind.[43]

42. https://smashed.by/businessmodel
43. https://smashed.by/authenticdesign

Steven Spear's *Chasing the Rabbit* (2008) is not just a business book, but a self-help book.[44] At the core of every success story is that companies built their new processes, forged their new markets, or empowered their employees by recognizing the uniqueness of their organization.

RESILIENCE ENGINEERING AND INEXACT DESIGN

Have you noticed how the big tech websites never go down or have a maintenance window? At a deep engineering level they follow practices and procedures to ensure their systems are not brittle, avoid failure or fail gracefully, and are fixed easily even with power failures, network breaks, and typhoons. There are many books on this, but unless you are going to run a data center, Erik Hollnagel's overview is a deep enough dive.[45]

In my favorite writing on resilience engineering, "Inexact Design: Beyond Fault-Tolerance," Gary Anthes quickly moves from how imprecision was a necessary evil at the dawn of computing, to a great opportunity in broader systems design today.[46]

DESIGN FOR IMPERFECTION

I extended the principles of resilience engineering to UX design. I'm not calling for a sea change in design, but for

44. https://smashed.by/chasingtherabbit
45. https://smashed.by/resilienceengineering
46. https://smashed.by/inexactdesign

consideration in how we design.[47] Good design already does all this, but understanding it helps you avoid pitfalls.[48]

Did you know that it's possible to use a McDonald's self-checkout kiosk to order a single serving of ketchup? There are no guardrails, no logic, so they let users do dumb things, like removing any part of a hamburger order's ingredients. Many other modern digital designs without well-conceived logical limits have led to injury or death. I used some well-studied examples in an article on avoiding stupid, rude, destructive, and deadly design.[49]

8. Test and Measure

As soon as you have designs, and as much as you can throughout the product development cycle, test on real devices, and get out and test in the real world.

If some of this seems repetitive, and we already discussed it under "Understand Your Audience" a few pages back, that's right. Your project may seem to be over at launch, but the project is just starting its life. Gather information and monitor behavior to improve it, and make sure good ideas that come along later make sense and can be well integrated.

47. https://smashed.by/imperfection
48. https://smashed.by/imperfectioninaction
49. https://smashed.by/destructivedesign

PLANNING

Don't use whatever phone is in your pocket, or assume your product design and development teams are representative of your users. Build a mobile device lab, and make the devices and SIMs available to everyone for testing and familiarization.[50]

Try to set aside biases by identifying them and working with other teams or team members to perform research whenever possible.[51]

INSPECTION

There's no such thing as realistically reviewing or measuring mobile device screens on a computer, much less a projected PowerPoint in a meeting room. Put designs on actual mobile phones and tablets as soon as possible. You don't need anything fancy, but take photos of the whiteboards and pen-and-paper sketches. Email the image over and show it maximized on the device screen. Try to read the words and tap the items.[52]

Never trust your math or your eyes. Instead, measure items directly on the screen to make sure text sizes and tap areas meet the guidelines. I built a little tool to help with this.[53]

50. https://smashed.by/devicelab
51. https://smashed.by/assumptions
52. https://smashed.by/mobileuxtools
53. https://smashed.by/touchtemplates

At right, you'll see **Touch Templates** at 100% scale you can print out or photocopy onto transparency film and have as a quick inspector for free.

CREATE MOCK-UPS AND PROTOTYPES

Don't wait until a coded version is built to try clicking through or testing with users. Employ prototyping tools and techniques to try out concepts as you create them, share with the team for approval, and test with users as early as possible.[54]

WATCH PEOPLE USING IT

What people say they do and what they actually do are two different things. Usability testing methods observe users interacting with products to discover how they actually work and to measure actual performance. If you can't hire a usability testing team, a good practical guide to the principles behind it is Albert and Tullis's *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics* (second edition, 2013).[55]

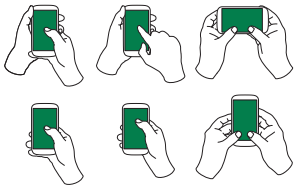
There's nothing I'm aware of that discusses the unique needs and methods of mobile, so I included a testing checklist in *Designing Mobile Interfaces*, and I have updated it a bit over time.[56]

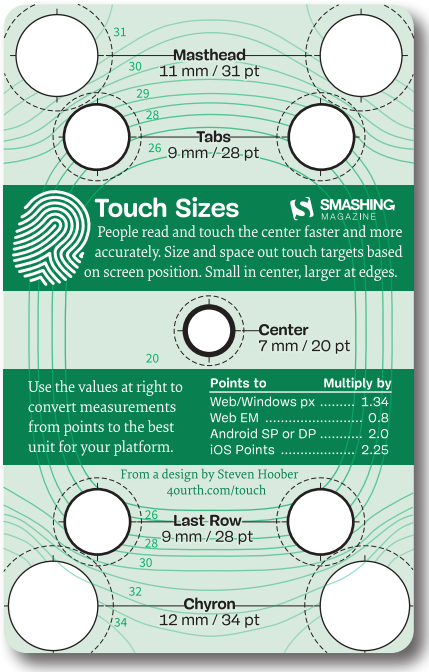
54. https://smashed.by/lowvshighfidelity
55. https://smashed.by/measuringux
56. https://smashed.by/evaluatingmobiledesign

Touch Sizes

Use this guide to determine if a touch target is safe, based on screen position. The four circle sizes cover most cases: on phones, tablets, and computers.

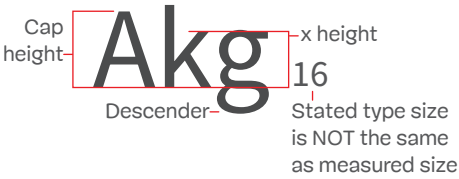
Center the target (or the most likely click area, like the first word in a label) in the circle. If any other clickable area is inside the circle, it is too close. Safer yet is no other targets inside the dashed outer ring.

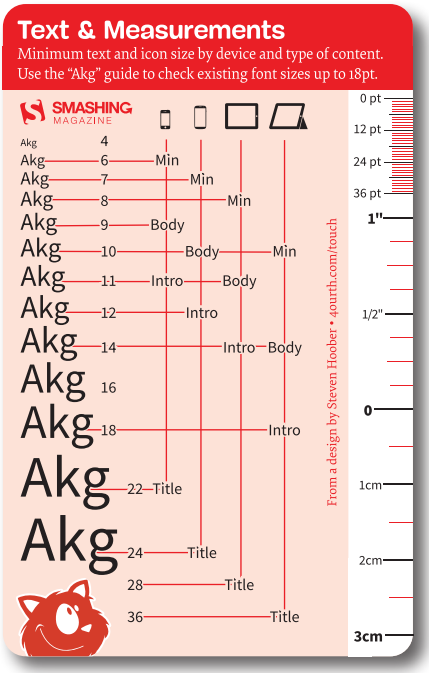




Text & Measurements

Use this guide to check your designs and measure text on the screen. Line up the guide text with the matching type size on the screen. The three letters give you a way to measure three types of letterforms:



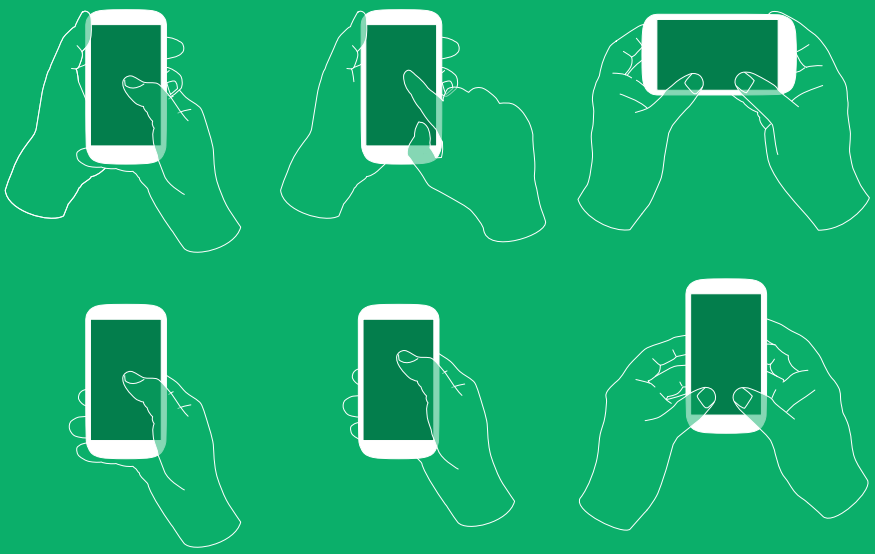


Remote testing is a good way to gather testing data quickly – especially as we try to avoid visiting people in person – and expand our reach to get more users from more regions. The quickest method is remote unmoderated test services such as UserZoom or UserTesting.[57]

This book is a comprehensive overview of my work on the topic of touchscreen use and designing for touchscreens. We have seen that most computing today is mobile: devices are everywhere, and mobile and touchscreen paradigms have been adopted into desktop and laptop computing.

We've covered the history of mobile and touchscreen technology, human factors, physiology, and cognitive psychology. Much of what we assume about mobile device use is either mistakenly based on outdated assumptions or just plain wrong. I've described methods of designing interactions that appeal to the ways people really use their touchscreen mobile phones and tablets.

57. https://smashed.by/unmoderatedtesting



MASTER CHECKLIST

Tips, Principles, and Best Practices

MASTER CHECKLIST

Tips, Principles, and Best Practices

This list presents a distillation of the tips, principles, and best practices for mobile touchscreen design found in chapters 5 through 11. See the relevant chapter for more detail.

> Remember that users are real people with their own lives, needs, and preferences.
> Design for every user.
> Accept that users change.

Chapter 5: Finding Out How People Hold and Touch

- [x] Plan and design screens to work well for every device that will access or install the product, not just our favorites. Design with both portrait and landscape orientation in mind.
- [x] People use their devices in many different ways, all of which are valid. Be sure to consider every method, not

just the easiest or most common. Design all interactions as though for mobile touchscreens.

- ☑ People shift and change how they hold and touch their devices based on environmental context and onscreen tasks.

Chapter 6: Touch Accuracy and the Center-Out Preference

- ☑ People touch and look at the center first, most, and best. Place key content and actions in the middle, and allow users to scroll content up to the center.

- ☑ Touch is less accurate along the edges. Provide enough room for people to tap controls placed in header and footer bars.

Chapter 7: How Fingers Get In the Way

- ☑ When actions are tapped or clicked, visual responses should be immediate and visibly big enough to be seen around fingers or pens. (For bonus points, add a haptic response.)

- ☑ Always use left-aligned text (or right-aligned for right-to-left languages) – never justified – to encourage

scrolling in empty areas. For tables and multicolumn lists include gutters and gaps.

- [x] Assume users will select near the beginning of label text, moving from the scroll position across the screen to select.

Chapter 8: Imprecision and Probability

- [x] Make touch targets as large as possible to give people the best chance to tap actions. Design interactivity using entire containers, like buttons or rows, never just a word or icon.

- [x] Design to avoid accidents by using single action buttons, displacing confirmation actions, and ensuring all actions are reversible.

- [x] Space out dangerous items so they are not near positive or commonly used items. Design processes to avoid destructive actions entirely. When not possible, provide undo functionality rather than asking for confirmation.

Chapter 9: Phones Are Not Flat

- [x] Environmental conditions, social norms, and individual abilities influence how people use devices. Don't assume audio or color is clearly available to every user

at every moment. Design onscreen cues, notices, labels, and captions to support audio.

- [x] Take into account accidental taps, double-taps, clicks, and drag-and-drops due to users' chance movements or jostling.

- [x] Ensure OS-specific gestures like edge swipes are not in conflict with the functionality of your product. Use margins wide enough to mitigate reduced reach and accuracy due to phone shields and cases. Don't use chyrons on the web, and with caution on Android or newer iOS devices.

Chapter 10: People Only Touch What They See

- [x] People process what they see in different and complex ways. Always multi-encode by adding text labels to all icons, adding shape and color to interactive text where appropriate, and so on.

- [x] Glare, unusual viewing angles, and other environmental factors (moisture, dirt, sweat) are universal. Icons and graphics should communicate their purpose

through shape and contrast more than color. Use the highest possible contrast to make content readable as widely as possible. Avoid outline icons and thin type weights except in large sizes.

- [x] Notices, warnings, and any other important data should be onscreen until dismissed by users. Avoid transient, temporary, and intermittently visible items.
- [x] All interactions should attract the eye, afford action, be readable, and inspire confidence they can be safely tapped.
- [x] Different devices are used at different distances from the eye, so use the best type size for each device class.
- [x] Give interactive items familiar, recognizable forms, like buttons, tabs, and form inputs.
- [x] Differentiate between interactive and static content by bounding the actionable elements or placing them within existing bound spaces already in the design, such as defined rows, bars, boxes, or buttons.
- [x] Make interactions large enough and isolated from other items to give users confidence in tapping without adverse consequences, even in difficult environments.

Chapter 11: Designing by Zones

- [x] Employ a visual hierarchy to organize information: position, size, shape, contrast, color, form. Use type sizes to communicate textual hierarchy.

- [x] Provide users with only the inputs and options they can focus on.

- [x] Organize screen templates into three zones: primary content or functionality in the middle; secondary information or controls visible along the edges as buttons or tabs; tertiary items hidden behind menus in the corners. Don't hide primary information, and if navigation is a primary function, place it in the center of the screen.

Index


The world is a miracle. So are you.
Thanks for being smashing.
SMASHING
MAGAZINE

"Steven's research for over 20 years and his extensive knowledge of all things mobile (not just phones) shines through. His insight makes you rethink assumptions and best practices that have not evolved alongside the technology. This book is a must-have for any designer, whether new to digital design, or not."

—***Jaee Likhite,*** *Aptiv*

"We live in a world of touch devices, and even those that don't have user interfaces are sure to be activated with touch. Reading Steven's book feels like you are taking a grand tour, full of wisdom and practical advice – and what to look for when designing and developing a real-world experience."

—***Mudassir Azeemi,*** *Ring Central*

"Many books written on technical topics run the risk of putting readers to sleep. Steven manages to make the dense topics contained within practical, relatable, and even fun. If you're looking for a good gateway into understanding design for touchscreen devices from a deeper perspective, look no further."

—***Mike LeDoux,*** *Accenture*

Steven Hoober shifted focus from graphic design to mobile UX in 1999. Since then, he designed the first Google mobile search, the first mobile app store, several mobile browsers, and numerous websites and apps for global brands. See more of Steven's research at 4ourthmobile.com

SMASHING MAGAZINE

9 783945 749975