

MOBILE DEVELOPER'S GUIDE TO THE FIFTH DIMENSION



Published by:



Wireless Industry Partnership Connector Inc.
Vancouver, BC, Canada
Austin, TX, USA
London, UK

www.wip.org

Please send your feedback, questions or sponsorship inquiries to:
developer@wipconnector.com
Follow us on Twitter: @WIPJam

1st Edition September 2013

This Guide is licensed under
the Creative Commons Some Rights Reserved License

Design and Artwork by:

Wiebke Becker, Allegra Schneider (www.koop-bremen.de)

Content by:

Contributors from our great mobile community
(see inside for details)



About this Cute Little Guide Book

Welcome to the 1st Edition of the **Developer's Guide to the 5th Dimension**. This Guidebook, the 3rd in a collection, takes a look into the design, user experience (UX) and user interface (UI) in mobile development.

The intent of all the Guide books is to give those new to mobile development (and those experienced too), a little primer on important mobile development topics in a handy easy to read and easy to access format. And they have certainly been popular! We can't seem to keep them in print, and are so tickled when we see them in a prime spot on a developer's desk.

The first booklet was produced in 2009, as the brainchild of Robert and Marco from **Enough Software** from Germany. The **Mobile Developer's Guide to the Galaxy**, on technical topics is now in its 13th edition as of this writing.

The second booklet produced by WIP, is the **Developer's Guide to the Parallel Universe: Warning Marketing Stuff**, is on its 3rd edition.

All of the Guides, are community projects and created with contributions from many members of the community, and are licensed under Creative Commons. Feel free to share and use what you like from the Guide, but be sure to give proper attribution of where the good bits came from!

We hope you will find the Guide to the 5th Dimension just as helpful and enjoyable as the earlier Guides. If you would like to contribute to future editions of any of the guides, help distribute and promote them, or sponsor the printing costs, please let us know.



UI, UX, Design— What’s the Big Hairy Deal?

So you think you’re a kick ass coder do you? You can implement a lazy loading list. And you know your *manifest.xml* from your *Info.plist*. OAuth—no problem. REST—piece of cake.

But let’s get serious for a moment. Are your apps in hot demand? Are you making any money? Or are you still living in Mom’s basement? (It’s ok if she still does your laundry—we all get Mom to do our laundry!).

Here’s a secret. It’s not the code that sells apps—it’s really the design and how your user experiences it that tends to make the sale and keep them coming back for more. So kick up your feet for a bit, we’re going to share a few knowledge nuggets to get your app to look as good as the code that is under the hood.

What is Design?

So what is design anyway? Think about this—there were many apps capable of editing and sharing photos before Instagram. What was it about Instagram that earned it an enormous, passionate user base and a billion-dollar acquisition? Or take Evernote—there are many apps that have a similar feature list, so why is Evernote such a runaway success?

The answer is of course, that people will pay—a lot—for good design. If there had been any doubt about this, Apple has sure put those doubts to rest by building a half-trillion dollar business based on technology supported by world-class design.

Hairy?



Design is not a final step taken to polish up an experience. Rather it's a way of thinking about an experience that takes into account emotion and aesthetics. If it isn't designed well, it doesn't matter whether or not it "works."

"Aesthetics matter. Attractive things work better."

Don Norman

The Difference between UX and UI Design

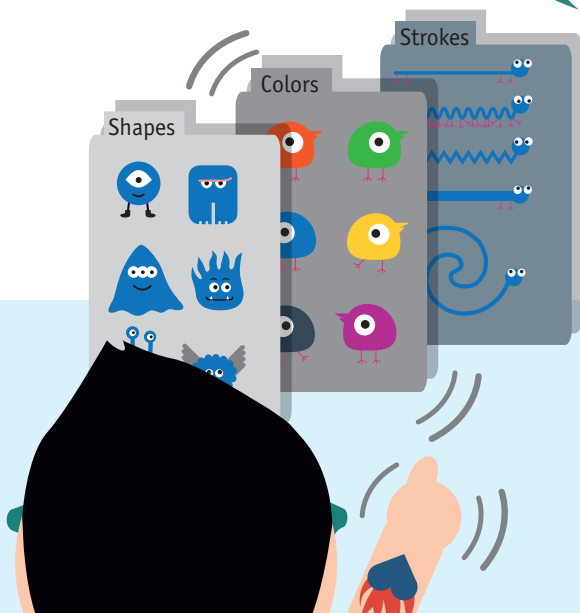
So you may be a little confused about UX and UI—where exactly do they fit into the design world?

UX or User Experience is a user's subjective opinion about the application. This opinion is based on how user feels every time he or she interacts with an app and the service environment that surrounds it (i.e.: while visiting company website to find out more about the product, getting in touch with customer care etc.). No one else owns this experience apart from the end user. UX designers work on influencing it by coming up with products and improvements based on the business and users' needs. Design projects can range from increasing content and interface quality, looking at better ways of assisting users in resolving issues, adding new features to an existing product or designing a new application from scratch.

User Experience Design is a broad discipline, and depending on how big or structured the team is, one person can carry more

responsibilities or switch between roles. UX designers concentrate on investigating the “why?” for what needs to be designed, while UI experts are focused on the details that answer the “how?” question. The clarity of how different design activities will inform each other and come together in the final product is very important in a group-working environment. And for the team of one, clarity can help focus on solving the right problem at the right time. UX designers are involved in all project stages to make sure that the final product meets initial goals.

More than aesthetics, design is a “utility enhanced by significance.” Daniel Pink





Why is Mobile Different?

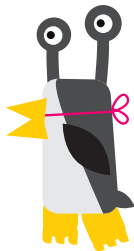
Designing for mobile devices is different than designing for other human-computer interfaces for a number of reasons. Don't overlook the differences—it will definitely affect the usability of your application.

Fragmentation

There are so many different types of mobile devices running on many different **screen sizes** and **operating systems**, there is a natural fragmentation that designers must contend with if they want their designs to function well and reach as large an audience as possible. Even if you limit your design to one specific operating system there are a number of mobile-centric design considerations to account for.

Mobile Means Everywhere

This means your design should take into consideration location; use of your app in public vs. private areas, online vs. offline, inside vs. outside. Take a game as an example. We can still keep users engaged even when the sound is turned off by providing touch feedback effects. Also, allowing different online and offline modes is also a good practice to take into consideration.



Mobile Means Touchscreens

Touchscreens double as input and display surfaces. Because of this, touch interactions are always obscuring the view of the user, so strict attention must be given to your touch-control design. Touch and gesture control points, visible or hidden, should be as simple to use as possible. A user should never have to repeatedly stab at the UI for the desired response. Because the display is obscured during touch interactions and audio cannot be relied on for touch confirmations, consider using touch feedback to inform the user of successful inputs or accepted gestures.

Mobile Means Social

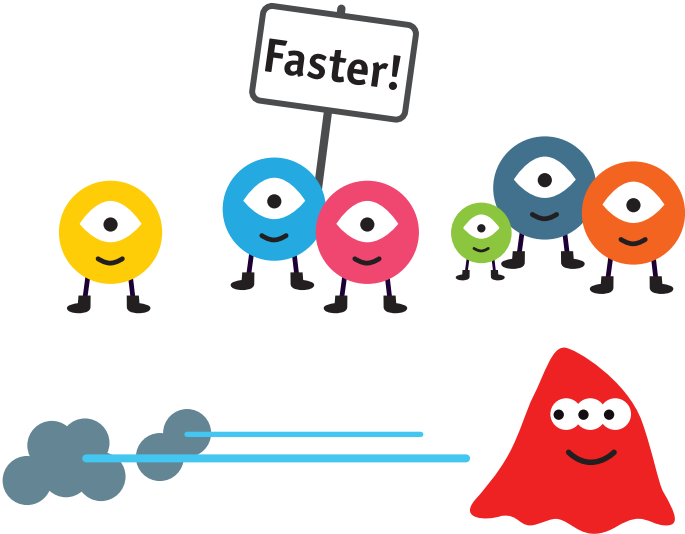
There are many social networking apps for mobile devices. Facebook, Twitter and Google+ are only a few that many people use daily. Incorporating an interface to these social apps into your design should always be considered as a method of increasing the visibility and sharing of your app.

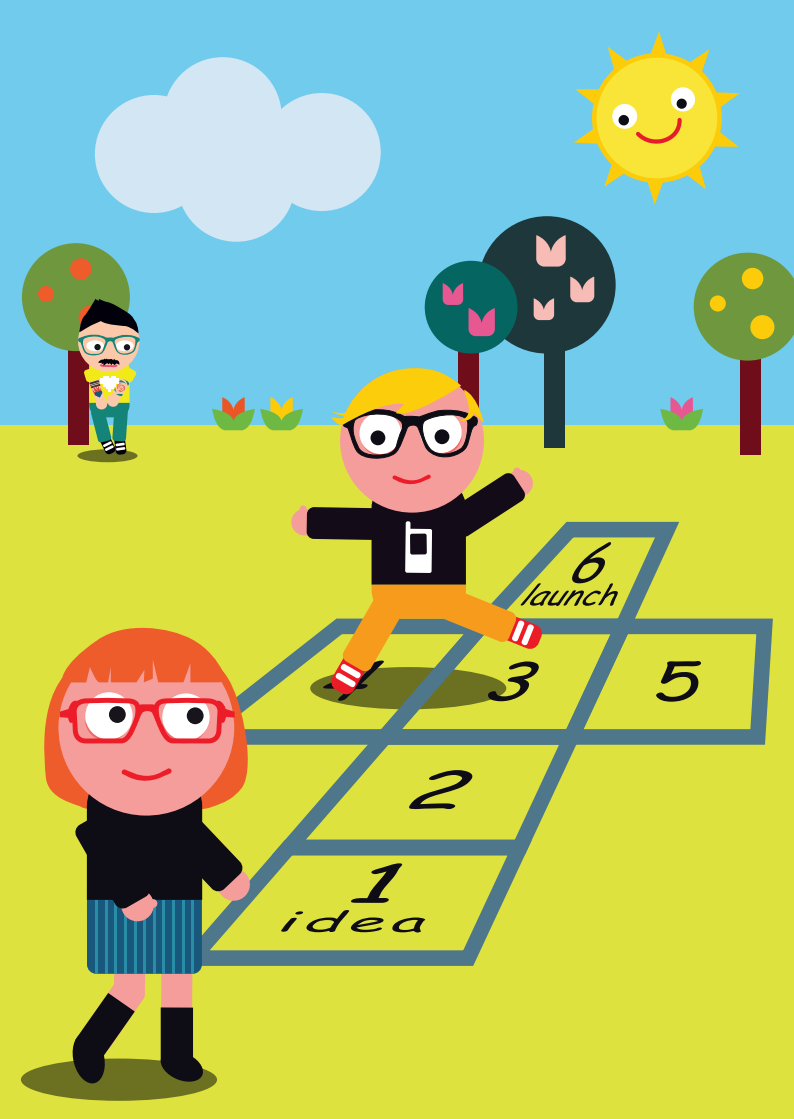
Latency

Mobile doesn't work the same way as a device hard wired to a network. Data access involves wireless transmission time as well as cloud server response time, so some latency (lag time) is inevitable compared with desktop apps. And speed matters. Scientific studies concluded that to keep a user engaged, simple user-input must be acknowledged within ~100 milliseconds. To keep the user engaged, the task must complete within 1000 milliseconds. In the web world, a 1 second delay has led to a 7% loss in conversions, 11% fewer page views and 16% decrease in customer satisfaction.

In mobile, users are even less forgiving. According to Strange-loop's research, 85% of mobile users expect sites to load faster or faster than sites on their desktop! The corollary to speed, is hungry apps also tend to eat up your user's batteries. If you don't consider the issues around mobile, and take care with things like image sizing and caching, your app will be doomed before it gets out there!

One last thing about mobile to keep in mind. Not all users are on high speed networks or smart phones. Many users in developing countries and even in western countries still use feature phones and are on different networks with varying bandwidth. It will pay dividends if you get to know the devices and networks your customers use, and design accordingly.





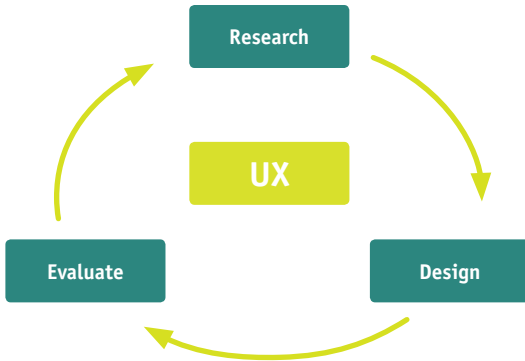
When Does Design Come into the Planning Cycle?

One of the biggest mistakes that teams make is viewing design as a block of work that is limited to the planning or development phase. **In reality design has to be fully integrated into every phase.** Why? Because it is not possible to anticipate how users will actually interact with a system without actually having them interact with the system. By employing methodologies like Lean UX, (see <http://www.leanuxbook.com/>) design is treated as an iterative exercise. Design is not allowed to “reach perfection” before it sees the light of day with customers (or the rest of the team). Frequent usability studies allow customer feedback to influence the design. And continuous deployment of the current product to customers in live production provide the necessary “voice of the customer” to modify design along the way. Continuous customer feedback becomes a bedrock principle.

Design also should not be thought of as strictly a function of the design team. Instead using concepts from the balanced team initiative (<http://www.balancedteam.org/>) and Lean UX , the right recipe is to involve product owners, designers and engineers throughout the process. It is critical for this balanced/lean team to reach a shared understanding and develop a collaborative cadence in order to bring the best experiences to life.

Design does not come in any particular place in the planning cycle. Instead it happens continuously throughout your development process.





UX design cycle—can be repeated several times before the design is passed on to the development team.

The design process however, typically starts with methods like: market research, competitor analysis, creating user profiles, understanding unique content and the value that the organization or product provides, getting insights from in-house experts etc. These research findings are then used to inform designers. In an app design project, **user experience is delivered via UI—User Interface**. Once the required features are listed, journeys can be captured in flows, user stories and wireframes—sketchy layout versions of the final screens.

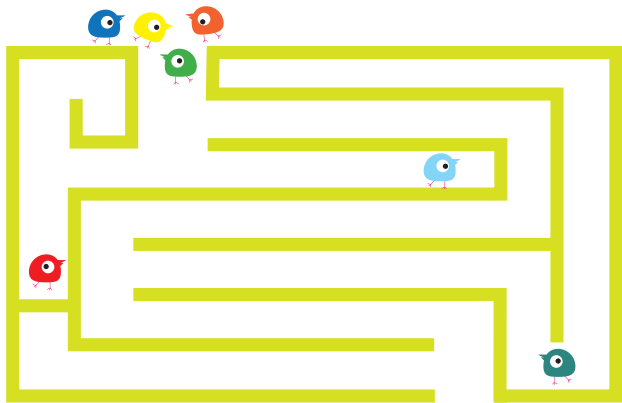
UI designers concentrate on the details of each user's journey through the application:

- **information architecture**—how the data is organized
- **interaction styles**—how users will interact with the interface

- **usability**—whether UI is understandable and easy to use
- **layout and visual design**—content arrangement on the screen, styles and branding.

UI designers also look at platform specific guidelines and UI patterns—common solutions to a certain problem, and work closely with developers to get their input early on. An **App's final User Interface is a combination of all these concepts.**

Interface ideas are best communicated with interactive prototypes—a clickable walkthrough that shows how all elements work together. Prototypes are not only for internal use. A “dummy” product put in front of users is a powerful tool to get feedback and evaluate the design early, without spending too much time on polishing the visuals. Good old pen and paper can also convey early design ideas. Both of these activities can help avoid major design changes in the implementation stage and save a lot of development time.



IMAGINE



FORM FOLLOWS
FUNCTION

AND NEVER FORGET

GOOD DESIGN STARTS
IN YOUR HEAD

REALLY?



Getting Design Direction and Inspiration

You've read this far, so chances are you are beginning to agree that design is an essential part of any mobile project. But when you sit down to work on the design of your app, what if you're not feeling inspired? Welcome to the club! Designers usually don't get ideas from flashes of insight either. Instead they **use specific techniques to trigger inspiration or establish direction.**

When design elements aren't based on user feedback, chances are they're based on something called "**design patterns.**" A design pattern is a "trending paradigm that describes invariant qualities, referencing history and similar solutions" (Jon Kolko, Design Synthesis). For example, the news aggregating and reading app **Pulse** uses a design pattern that has become known as a "**help overlay.**" As Figure 1 shows, a help overlay is a semi-opaque screen of hints and guidance, laid on top of the functional UI. The overlay is typically dismissed by touching it.



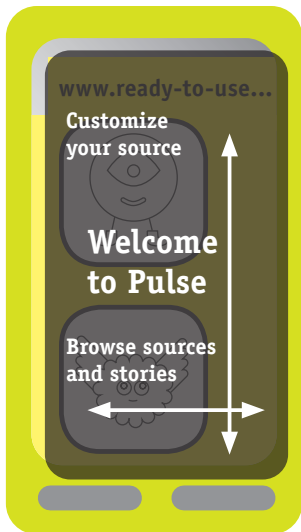


Figure 1:
Pulse’s “help overlay” screen

Keeping up-to-date with design patterns can make your life a lot easier because a lot of hard design problems that crop up have already been solved.

Another source for potentially useful design ideas is to **look beyond mobile**, and even beyond software. Industrial design, furniture, architecture, print, and other design disciplines may offer interesting ideas or at least starting points. There’s an iPad app called **Design Scene** that simply showcases images from design-centric websites.

For UI design, a great resource is *Dribbble.com*, a place for designers to “show and tell” what they’re working on. In addition to inspiration, you may even find the perfect designer to hire for your project.

If all you need is a set of buttons and other UI elements, you can purchase any of hundreds of vectorized, matched sets of UI widgets from websites like iStockPhoto.

If you have a small visual design project that needs a custom creation, you might consider a website like *99designs.com*, which allows you to post small job offers and offers a community of designers to bid on your project.



does
not fit!



Designing for Multiple Screen Sizes and Fragmentation

As mentioned earlier, in today's mobile world, apps need to run on many different devices, which means designing for multiple screen sizes. Yikes! It's important that your app makes the best use of scarce screen space on smaller devices, while still looking good on large screen devices. On a PC, developers can always assume a baseline screen size of 800×600 pixels is supported. That doesn't work for mobile devices! Low end phones might have a screen as small as quarter-VGA, which is 320×240 pixels.

With multiple screen sizes, developers are confronted with **two obvious choices, neither of which is satisfactory**:

1. Design your app to run well on really small screens. This wastes screen space and gives a poor user experience on larger-screen devices, risking customer loss.
2. Design your app to run only on medium and large-screen devices. This gives up on a potentially large number of customers who use low-end devices, guaranteeing some customer loss.

The wide variation in screen size is one aspect of the “fragmentation” issue that captivated large numbers of journalists and bloggers a while ago and was generally seen as a significant barrier to BYOD (Bring Your Own Device—the adoption of personal devices for corporate purposes).

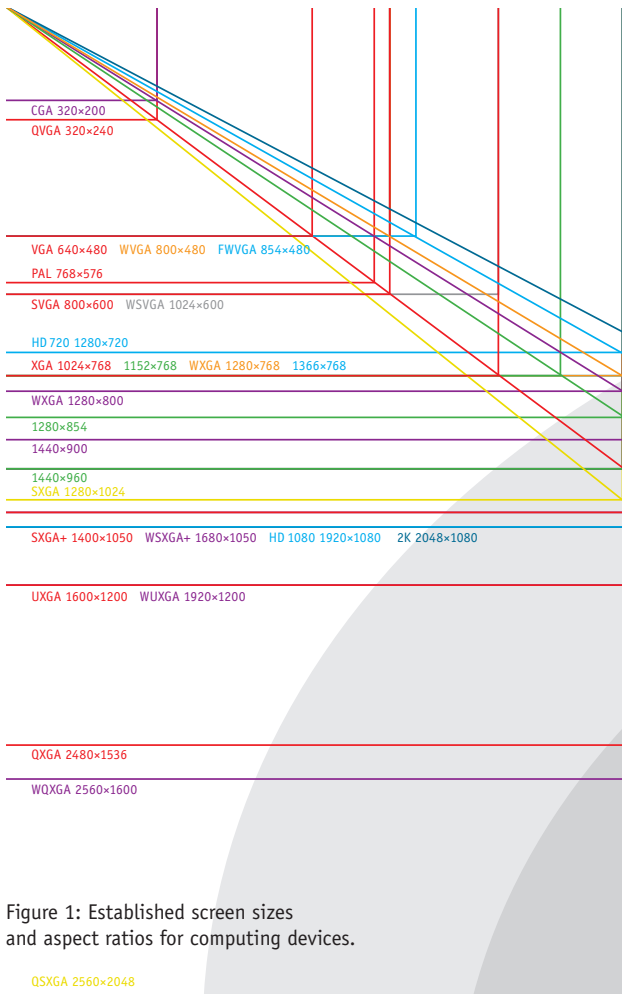
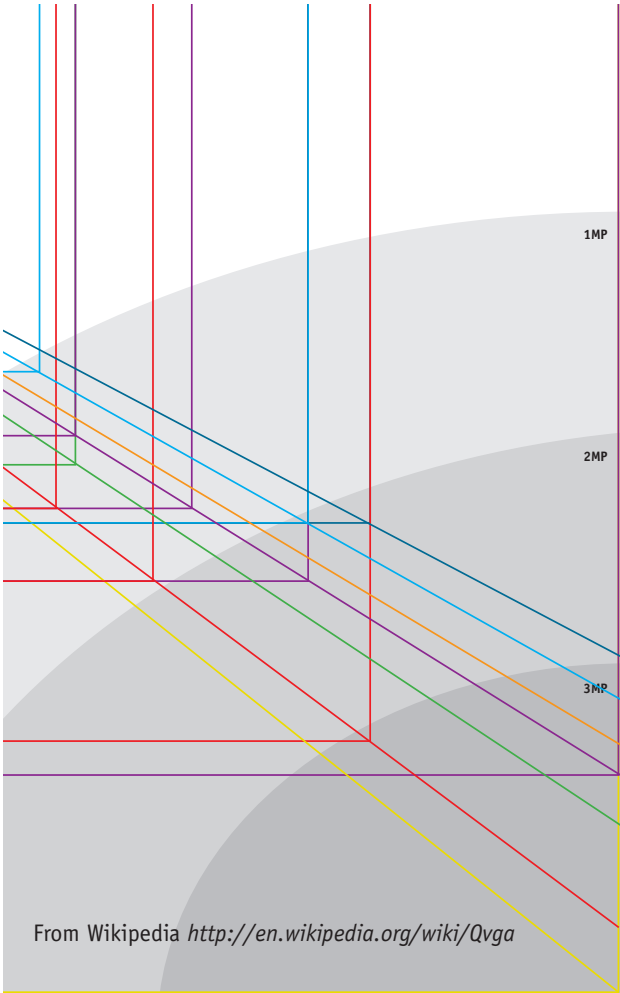


Figure 1: Established screen sizes and aspect ratios for computing devices.



In addition to screen sizes, the other aspects of fragmentation are:

- different revision levels within one mobile operating system, and
- mutually incompatible operating systems (such as iOS, Android, Blackberry 10, etc).

Additional information on operating systems and platforms can be found in the Developer's Guide to the Galaxy.

Figure 2 illustrates the range of screen sizes that are seen on mobile and other devices. Devices can be built with any of the screen sizes shown, or at any other point on or off the diagonal lines.

What is Responsive Design

So where does responsive web design fit in? You've probably heard a lot about responsive design lately, one of the most popular design topics these days, and an important one too! Wikipedia describes responsive design as the practice of using fluid grids, flexible images, and media queries to progressively enhance a web page for different viewing contexts. Top web design and development publications such as A List Apart, Smashing Magazine, and CSS Tricks have published a wealth of information on the subject—and with good reason, too: a responsive design workflow is a fantastic way to build tailored web experiences for different screen sizes and resolutions.

Why do Mobile Web Developers Love Responsive Design?

Responsive design has become the approach-of-choice for web designers and developers because of its flexibility in supporting the creation of websites that are optimized for a multitude of screen sizes and resolutions. With the number of screen sizes and

resolutions growing rapidly, responsive design solves the problem of how to make websites look great on every device (and also applies to apps that can be implemented as a web app).



So rather than building a website specifically for a desktop, Blackberry, iPhone, iPad, phablet, smart TV, Google Glass etc. etc., responsive design lets developers publish one set of markup to adapt the website for the user's device.

Unlike former mobile solutions—like “m-dot” which redirects (e.g. <http://m.example.com>) or proxies, **responsive design maintains one URL**. This means that users will always be able to access your website, no matter where they start or end their journey.

Design approaches tend to gain traction only when a community of people gets behind them. Responsive design is no different, and there are a ton of helpful resources[1] available to developers who are looking to get started. There are also several easy to use responsive front-end frameworks, such as Twitter Bootstrap[2], which make it very easy for front-end designers and developers to create new responsive projects and learn the techniques necessary for making inspirational responsive websites[3] that adapt beautifully to every screen.

That's Great! Where's the Catch?

OK, so Responsive Design does have a couple of skeletons in its closet. First and foremost, if you're undertaking a responsive project, it's important to factor mobile performance into your approach.

The reason for this: you're cramming a whole lot more content into your markup, so you'll need to prepare for the fact that the page size will go up and the image size will go up (in order to cater to top end Retina screens), but mobile users remain on limited bandwidth.

There are two main ways you can make sure that the mobile user experience of your responsive project doesn't crash and burn due to performance:

1. Optimize Images

Since a responsive design uses one markup across devices, you need to ensure that only big, beautiful images are served to your Retina Macbook Pro, while a standard definition smartphone is sent smaller images that make sense for its lower-resolution screen. Images represent the majority of kilobytes in a page, so you can also make the biggest performance savings by optimizing them!

2. Minify Scripts

One of the most effective ways to quickly speed up the performance on your responsive website is to make sure that only the necessary resources are sent to smartphones, tablets and desktops. By optimizing the number of HTTP requests for the end device, you can reduce the amount of time a user spends waiting for the DOM to load.

Here are a few resources to check out:

1. <http://bradfrost.github.io/this-is-responsive/resources.html>
2. <http://getbootstrap.com/>
3. <http://www.mobify.com/blog/70-stunning-responsive-sites-for-your-inspiration/>

iOS Support for Screen Size

The iOS approach to address screen size fragmentation is simple and direct. Apple has rigid central control over all aspects of software and hardware, including screen size. When Apple introduces a device with a new screen size, developers have to upgrade their apps to support it. In the current iOS lineup, there are two different screen pixel counts for phones, and three for tablets. The

retina (high resolution) screens are available on some devices and not on others. The iOS screen sizes are illustrated in figure 2.

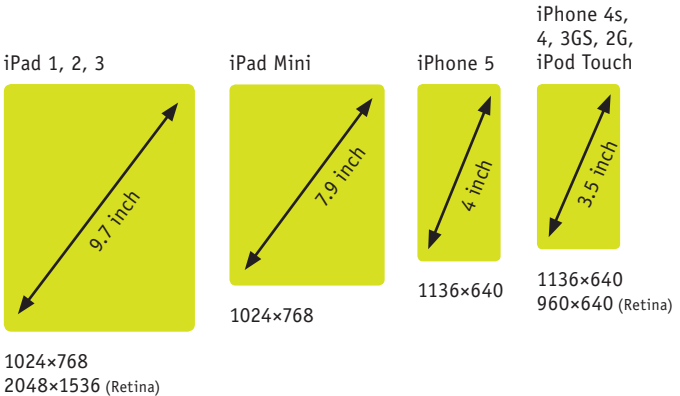


Figure 2: iOS screen sizes used with permission of <http://www.jaysquared.com>

If you want your app to work across all five device types, you need to create 5 layouts for the app. The framework knows what hardware it is running on, and will choose the corresponding layout at runtime allowing you to satisfy a larger pool of users. If you don't provide a layout suitable for a specific form factor, the iTunes app market will not offer your app to users with a device of that form factor.

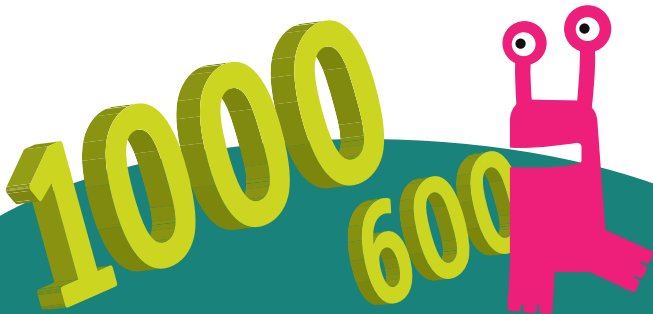
For the high resolution ("retina") screens, developers don't have to modify their code. You simply provide a double-sized version of each image file, with the suffix "@2x". The iOS framework knows to use this version of the file on devices with retina screens. Since the iOS coordinate system uses an abstraction

of “points” rather than pixels, the dimension and location of UI elements does not change when executing on a high or low resolution device. **Points are the Apple term for “density independent pixels” used in Android development.**

Android Support for Screen Size

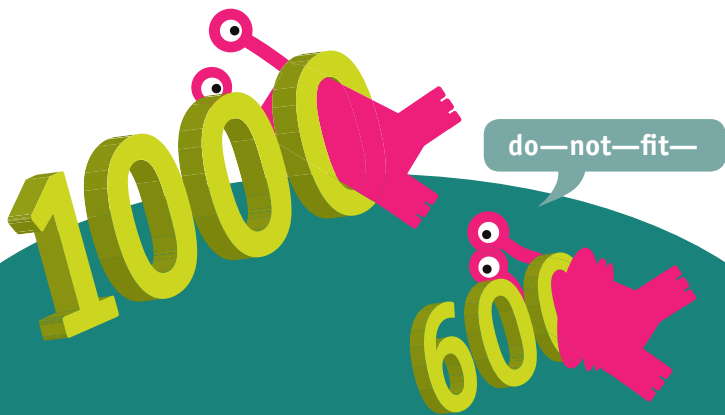
Originally, Android featured only rudimentary support for screen sizes. All screens were grouped into four categories of small, normal, large, or xlarge. However, the boundaries between sizes were not well-defined, and a device classified as “large screen” could be smaller than some other device classified as “normal screen”. The “4 categories” approach broke down completely with the introduction of Android tablets. All tablets were xlarge, but so were some phones, and this nomenclature did not adequately characterize the situation. Because of this and other limitations, screen size support was comprehensively revised in the tablet-only release of Android (Android 3.2, API 13).

Ever since that summer 2011 release, the Android framework has featured excellent support for laying out apps for all screens, down to the level of individual pixels. This is done in the usual way, by using the Android “alternative resources” feature. **“Alternative resources”** uses the pathname to a resource file to encode the details of the hardware for which that file is to be used.



For example, the technique allows you to have a file called *“mylayout.xml”* in a folder called *“layout”*. (The actual pathnames are more obscure than this, but you get the idea). You may have a second version of *mylayout.xml* in a folder called *“layoutsmallwidth600pixels”*. When you are running on a device that is at least 600 pixels wide, the framework will automatically use the second file for layout. On other, smaller devices, the first layout file will be used. Remember **screen size is dynamic**—the screen width will change as you rotate a device from landscape to portrait, potentially causing a different layout file to be used.

This “alternative resources” technique of mapping hardware features onto pathnames is very powerful. It allows hardware-specific versions of every type of resource: layouts, strings, shapes, colors, images, and so on. That in turn means that it is very easy to localize apps for other geographic regions. Just put the locale-specific files in the correct pathname under the resource folder. Further, “alternative resources” supports localized versions of **any** resource (like pictures), not just text strings.



Android also supports versions of image files for different resolution screens. If you don't provide alternative image files for high resolution devices, images that look normal on a normal resolution screen, look tiny on a high resolution screen. **Different screen resolutions** are supported the same way—through encoding the resolution size into the pathname to the file. Screen resolutions are still classified into 4 groups (as screen sizes used to be). The groups are low dpi (~120 dpi), medium dpi (160), high dpi (240) and extra high (320 dpi) resolutions. This approach to support different resolutions seems adequate for current purposes.

The older approach of 4 different “buckets” for screen size has been deprecated in full. The `Fragment` class provides a way to build up a large screen layout, re-using for part of that screen, a layout designed for a smaller screen. The **biggest challenge for Android designers is figuring out the right timing to drop support for pre-Honeycomb devices altogether**, and use the new screen size features in all apps going forward. To make that decision, review the Play market statistics that Google maintains at: <http://developer.android.com/about/dashboards/index.html>. At the time of writing (summer 2013) it shows Android 4.x devices are approaching a 2/3rds market share—not yet enough to cut off all older releases.

Conclusion

The difference in the way Android and iOS handle multiple screen sizes is this: iOS regards every new device as a special case, to be coded uniquely. In contrast, Android uses the framework to select at run time the correct resource files for this specific combination of hardware. In both cases, of course, the designer has to pay painstaking attention to detail to obtain satisfactory results on the widest variety of hardware.

UI Guidelines for Popular OSs

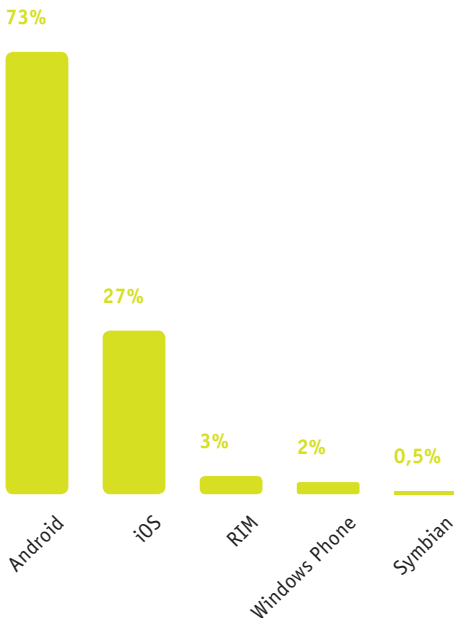
A blank touchscreen is like an empty canvas, but that does not mean a designer can paint on it with complete abandon. All popular mobile operating systems (see figure 1 for a vivid illustration of «popular») have guidelines constraining how UI elements are to be used on the screen.

Let's distinguish between UI and UX, by relating the concepts to the automobile world. UX—User Experience—is the overall feeling you get from sitting in a car and driving it. It's the sum of the car interior and everything you see, feel, hear, and interact with. The UI—User Interface—is a subset of this. UI is the set of controls that operate the car—the transmission shifter, the pedals, the door lock, the steering wheel and so on. Knowledge of the UI enables you to drive. How much you enjoy that drive depends on your UX experience, combined with your personal preferences.

The UI controls for cars are mostly standardized now, but in their first few years some automobiles used a steering tiller instead of a steering wheel, analogous to the rudder on small boats. The UI for mobile devices is currently at the same place: we are still pioneering new and better controls. Some UI controls are common to most platforms. Everyone has scroll bars, radio buttons, text entry fields, support for scrolling lists, and so on. **But even where there are common controls, each platform has different names and APIs for these components.** The website <http://kintek.com.au/blog/portkit-ux-metaphor-equivalents-for-ios-and-android> has an excellent side-by-side comparison of similar widgets available in Android and iOS.

Every platform also has some unique UI features. Make sure you work with designers and artists who are familiar with the UI features of the platforms you are targeting. Designers are always keen to invent new UI components (sounds suspiciously similar to developers!) But without a good knowledge of all the components that are already in the toolkit, newly-invented UI components might not fit gracefully with the platform. A designer who is unfamiliar with the native controls on the platform of interest may not achieve the results you want.

Figure 1: Worldwide smartphone sales by OS, Q1 2013



While there are interesting new entrants to the mobile OS space, like Samsung's Tizen, the Ubuntu Phone, and Mozilla's Firefox OS, the smartphone market is at best a two horse race, with Android currently taking 75% of smartphone global sales, and Apple picking up half of what's left. Note that "current sales" is a leading indicator of where the market is going, while "installed base" is a trailing indicator of where sales have been in the past.

(source: http://en.wikipedia.org/wiki/Mobile_operating_system)

UI Guidelines for iOS

The iOS UI guidelines can be found at <http://bit.ly/iOSguidelines>

The guidelines are well-written and easy to access. They contain solid "do's and don'ts" such this one: "Don't create a multi-segment back button." So there is no doubt about what a multi-segment back button is, Apple provides an illustration (figure 2).



Figure 2: A multi-segment back button. Don't create one!

UI Guidelines for Android

The Android UI guidelines have been updated repeatedly in the last couple of years, and developers are still adjusting to the changes. The UI guidelines improved greatly after Google put a senior director, not an engineer, in charge of the user experience, for the Honeycomb release for tablets in May 2011. Significant new UI components have appeared, such as:

- ViewPager (added to the compatibility support package late in 2011, but still not in the main framework),
- Fragment (added in the Honeycomb tablet release, to simplify re-use of chunks of UI)
- GridLayout (in the main framework from late 2011), and
- Space View (added in API level 14).

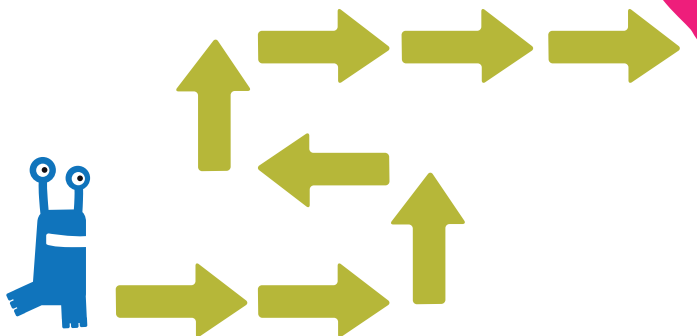
The easily overlooked and often useless menu button is out, replaced by the action bar. Still part of the UI, but with radically new semantics, is “long press” on a key. In Android API level 1, a long press on the home key displayed the activity stack. In API level 2, a long press on the search key initiated voice search. In API level 3, a long press on the menu key forced the software keyboard onto the screen. In API level 7, we got support for virtual (i.e. on screen) hard keys with a proper key event API and callbacks for long presses.

In Honeycomb (API level 11) the long press semantics abruptly changed from “offer something from a context menu” to be a universal gesture for selecting data, using a new component called a “contextual action bar”. The “exception which proves the rule”—using a long press on the back key to bring up recently-viewed pages in a browser—still works in Ice Cream Sandwich but was removed for the Jelly Bean release. The UX will continue to be tweaked and improved on; UI and UX are never really completely finished.

The landing page for Android UI Guidelines is at <http://developer.android.com/guide/topics/ui/index.html>. From there, you can reach hundreds of webpages with detailed advice on how to use components of every variety. One problem is that these guidelines try to be exhaustive, and that sometimes makes it difficult for a developer wanting to find a small needle of information, in a giant haystack of documentation. Goldilocks notwithstanding, it's better to have too much documentation than too little, we suppose.

UI Guidelines for BlackBerry

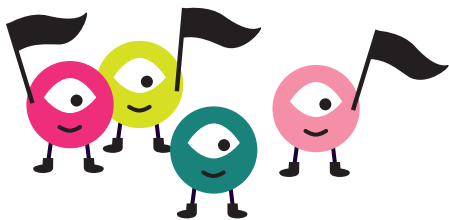
The UI Guidelines for the BlackBerry 10 platform may be found at <http://developer.blackberry.com/design/bb10>. The BlackBerry platform is in the middle of an SDK transition at the time of writing (summer 2013). The former platform, BlackBerry 7 OS, supported development in a variety of languages, including C, HTML5, and Java. The current platform, BlackBerry 10, drops Java from the supported implementation languages. So, however you follow the UI guidelines for BlackBerry, you won't be following them using Java as your implementation language.



UI Guidelines for Windows Phone

The biggest thing to consider when designing for the Windows Phone platform is which of the 2 standard navigations to use: Pivots vs. Panorama. For both it is best to limit the menu items to a max of 5 or else the user tends to get lost as the items become hidden. It is also recommended to only use Panorama once in your app and to go from Panorama to Pivots rather than Panorama to Panorama to avoid confusion.

Apps are starting to stray from the WP8 standard UI and stay more consistent with their iOS and Android counterparts. There is a trade off when doing this with WP8 users that being inconsistent with the rest of the OS can hurt their experience and you won't be taking advantage of standard OS UI components.



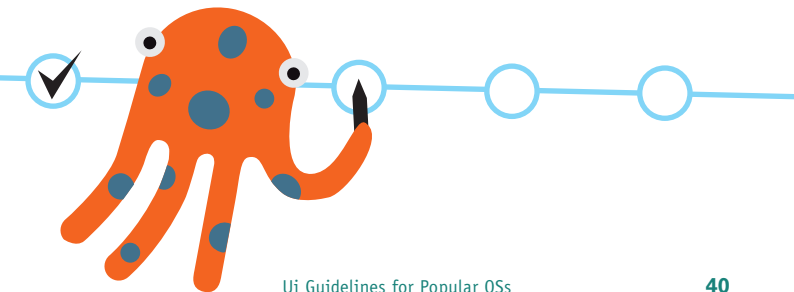
Conclusion

On both iOS and Android, a large selection of custom controls is available. Developers “make their bones” by writing custom UI controls. For example, a scrolling list is a standard control. A custom version of that might have the scrolling behavior, but also the equivalent of a magnifying glass across the middle of the screen. As list items pass under it, they are magnified, making them easier to read.

Custom controls are a little more popular, and a little more available on Android, because the framework code is open sourced, and available for all developers to review. You can search the Android source online at <http://androidxref.com/source/>.

> *See also the later section on “Tips on Custom Controls”.*

The bottom line with UI guidelines for all platforms is that you need to know in detail what the guidelines say. If you decide to ignore the guidelines, make certain it is an informed decision. All mobile platforms have their design quirks and revisions. It’s much quicker to work around these in application code, than it is to wait for the framework to acknowledge the issues and address them. But if you ignore the guidelines in your designs, it is your responsibility to make it work, and make the new interactions easily understandable by your user community.





go forward!

IMPROVE

The Sky is the Limit

↑

Tips on Dealing with Custom Controls

When there are almost 1 million apps in Apple's App Store and on Google Play, being noticed can be rather difficult. Creating custom controls is one way to differentiate your app from the multitude of others that exist around it.

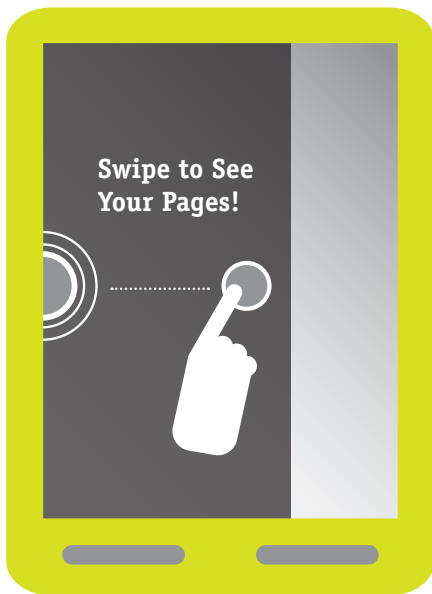
Here are five key points about creating custom controls:

1. Ask Yourself if it is Necessary. "It's very easy to be different, but very difficult to be better," said Jony Ive. You shouldn't create a new control just for the sake of being different. Instead, do so only to complement your app's design and create more fluid usability.

2. Master the Default UI Before Creating a New Custom Control. Other than gaming programs, almost every app is simply ListView with a detail view. Although all these apps have a similar baseline, within them we see millions of different ways to manipulate the generic UI. This type of operation improves on the rules rather than breaking them. Twitter pull-to-refresh is a great example—it is still ListView, but upon an interaction, a pull-to-refresh shows up.



3. Take Advantage of Intuitive Gestures. Although gestures such as swiping and pinch-to-zoom are almost second nature to people who use today's mobile devices, these gestures are rarely implemented in most apps. Taking advantage of these intuitive gestures can really improve an app's interface. One excellent example is the Pulse News app, which allows for both horizontal and vertical swipes.

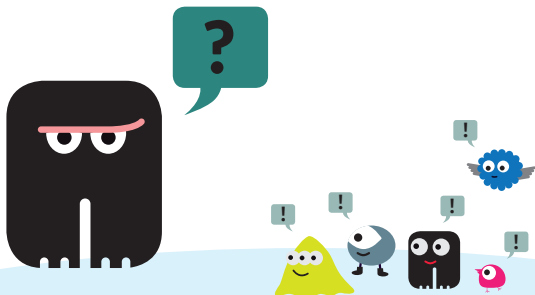


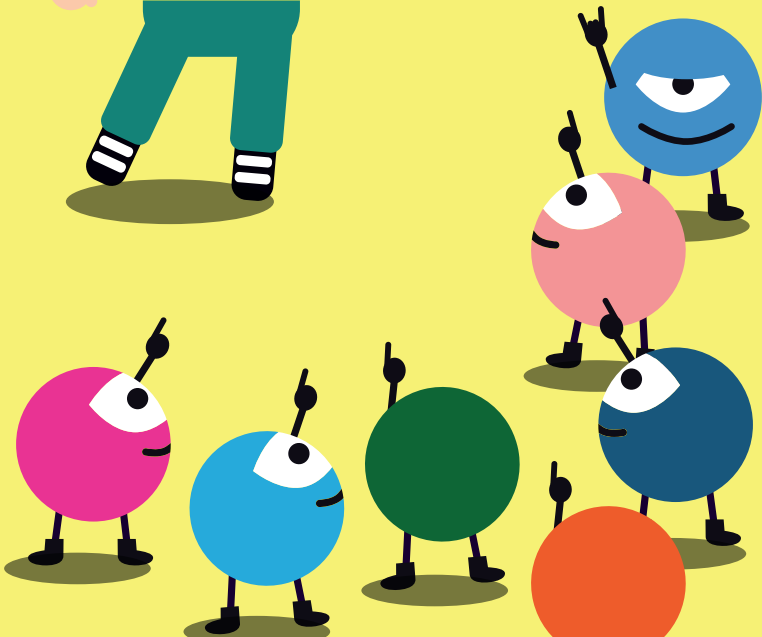
“Simplicity is usually the result of much complex thinking.”

Lee Clow

4. Simplicity. It is important to be simple and represent only essential information, but become too simplistic and the point can be lost entirely. Apps must use controls effectively in order to display the right information at the right time. Otherwise every app would use a boring ListView or Table View (iOS), and display only a minimal amount of information. The look and design of an app is important, but its true essence lies in its utility.

5. Creative Testing. Use your intuition, and test your apps with children who have never used it before. A six-year-old can usually use an iPad without any prior knowledge or training. If your app’s custom control can obtain a 12-year-old’s “approval” without any prior information, it will be more than successful on the market.



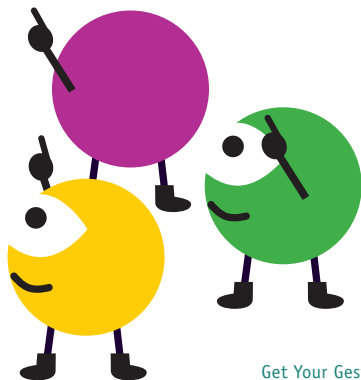


Get Your Gestures On

People love using mobile apps that are satisfying and fun to use. A lot of that satisfaction and fun come from gestures. People love touching and feeling what they're using, and intuitive gestures plug into our instincts about how interacting with objects works. A great gesture is obvious, direct, and instant.

Obvious gestures are obvious. Mobile users learn some obvious gestures very fast when learning their mobile device, just by trying to use it. Tapping, scrolling, panning, and zooming are all obvious enough that you don't need to teach users about these gestures. Nine times out of ten, these gestures are the ones you should use—obviousness beats novelty every time.

Direct manipulation is joyous. Scrolling by dragging content with your finger is inherently satisfying, since you're directly manipulating what you see. All the obvious types of gestures involve this direct manipulation. More obscure gestures like tap-and-hold or swiping on objects are okay for advanced features, sort of like a keyboard shortcut.

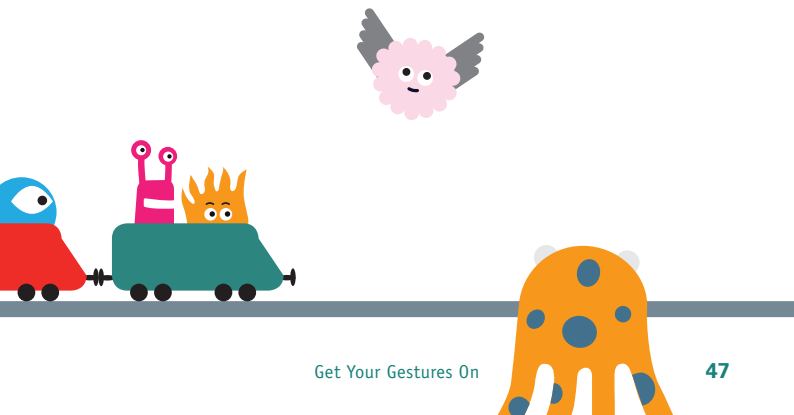


A great example of direct manipulation is the new “delete” shortcut on the iPhone and iPad. If you quickly swipe your finger across an email message on iOS 6, it will detect a “swipe” and then display a delete button. It’s handy, but hard to discover so not very satisfying. The fix with iOS 7 takes the same gesture but pans the mail message underneath your finger, revealing a delete button in one smooth motion. Because the new gesture involves directly manipulating what’s under your finger, it’s more obvious and more usable.

Instant responses give instant karma. Gestures are only fun and satisfying if they keep up with your finger. Even the fundamental scrolling gesture feels wrong if it can’t keep up. For animations and transitions that aren’t tracking users’ fingers, people are somewhat forgiving of jitter or lag. For direct manipulation though, your app will immediately feel slow or “janky” if there are delays or pauses as short as 1/30 of a second. If you commit to adding gestures to your application, it is your responsibility to make them snappy.

A few more basic tips

Here are a few tips and resources for raising the level of design in your project.



- **Color matters.** Depending on your color palette, you can make your app feel fashion forward or retro; reserved and dependable, or ebullient and playful. A good resource where people from around the world create and share colors, palettes and patterns, and discuss the latest trends in color is: www.colourlovers.com
- **Legibility: Typefaces** really do matter. Try to use a very restricted number of typefaces—like, one, or maybe two.
- **Layout** matters too. Leaving space around text makes it more legible and more pleasurable to read.





LOUCHİ

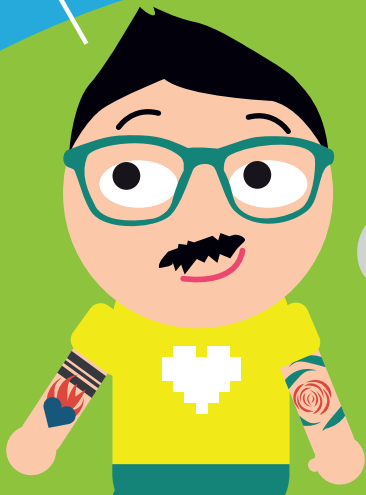
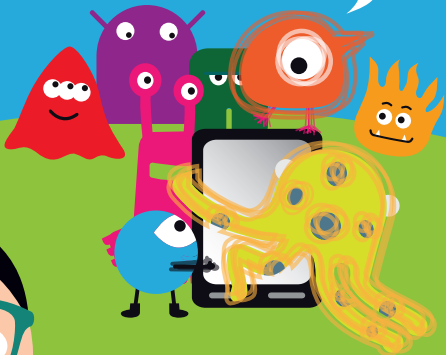
$\downarrow \sqrt{\Omega \circ \xi} \equiv \pi?$



TOU-U-U-UCH?

T@UCHİ

sigh



Beyond the Visuals

Most people think that UX and design is just about what the user sees. But what about getting them to use their other senses too? Well at least touch and sound—it will be awhile until we see apps that let us use our sense of smell and taste!

Touch or Haptic Design

Haptics refers to touch feedback technology, and is used to create the vibration response of software keyboards, vibration patterns for ringtones, and the “rumble” effects in console games. Because our sense of touch is so central to our everyday experience, integrating haptics into your app can make it more engaging, intuitive, and emotional. A few haptic effects used sparingly and on-target can dramatically increase the usability and perceived production value of your app.

So you’ve decided to follow the leaders and utilize haptic effects in your game or app. Now what? Here are some guidelines that should help you decide when and where to engage your users’ sense of touch.

Motion and Magnitude

Haptics engage our sense of motion and action. When possible, ensure that you synchronize haptics to animations. Animations can include visual actions in games, transitions between UI views, or scrolling menus. Remember to make the magnitude of the haptic sensation match the magnitude of the visual sensation. A tiny visual event should not have a strong haptic effect associated with it.

Audio

Haptics and sound are a match made in the brain. In the physical world, we rarely experience a haptic sensation without an accompanying audible sound. (Try tapping or sliding your finger on any object around you and notice the sound that accompanies the touch sensation). Your users' immense number of past haptic-audio experiences, creates an expectation that sounds should feel a certain way, and vice versa. Make sure your choice of sound and haptic effects play into these expectations.

Sync

These principles can be summed up as, "it should feel like it sounds like it looks." But there is one final ingredient that, if left out, will cause the experience to fall apart: synchronization. User studies have indicated that when haptics are out of sync with other modes of feedback by more than about 50 milliseconds, the user experience substantially degrades.

Resources

The leading company in Haptics is Immersion Corporation of San Jose, California. Immersion has specialized in haptic software and hardware for around two decades. Immersion licensed its technology to game console manufacturers for use in rumble packs. Immersion currently offers a haptic library for Android. The library is licensed without cost to Android game developers, and provides a much finer granularity of control over the vibration actuators than the Android framework API does. Developers can thus place subtle and nuanced haptic effects in their games, giving a more engaging overall UX. And a richer game experience for users translates to better monetization opportunity for developers.

Sound Design

Audio can be overlooked, but is one of the most fascinating parts of software development and is a must in games. It adds one more dimension to a world of flat screens and small displays on mobile devices. Try watching a movie on mute and you'll quickly see that sound makes up at least half of the user experience. Even if you're not in the games industry, sound can help give the user feedback such as, what they are navigating through in an interface, give hints on when something is done incorrectly, or can simply indicate that something is being interacted with.

Because you are manipulating an intangible asset, finding the right sound can be a daunting task. You have to remember that "the sound is not the sound". Here's an example: most of us have seen a movie, TV show or game where there is a dinosaur roaring loudly on screen. As we have no tangible evidence of what dinosaurs sound like, we have to create their **voice** from other animals that we think they **might** sound like. Some sound designers have used lion sounds mixed with pig squeals, morphed with even jet sounds. Be creative with your ears!

Keep it Simple. Imagine you were making a productivity app that featured lists for the user to navigate through, with items on each. You will want to give audio feedback to the user, so the feel of a button that is created virtually is more tactile.

At the top level, the user will be sifting through a bunch of lists. Stick with something simple like the sound of a button click of a mouse, or a light switch. Those sounds have a very quick attack and quick release. Another aspect of these sounds is that they don't have a lot of low frequencies, so they will most likely sound fine on most devices and speakers.

When the user chooses a list that they want to open, you could add the sound of your fingers snapping on the “mouse down” event. This sound has a quick attack and release but more **body** or fullness than a mouse click. This gives the user feedback that they have chosen something and will bring them to a close up on a list. If they browse through the items on the list, you can use the same mouse click sound. This will help you save on footprint, resources and the time to find more interface sounds.

Keep your sounds organized as to not confuse the user. In this example, “mouse click sound” means that the user is browsing and the “finger snap” means that they chose what they were browsing through.

Paint with Sound. Mouse clicks and finger snaps are great for many pieces of software such as operating systems or productivity software, but lets say that you were making an interface for a puzzler game that used round fonts and light colours. You could stick with clicks and snaps, but you could further add to the user experience if you were to match what they see. For something like this, I would use “rounder” sounds. Sounds with a slower attack to give it a softer feel but, but not too slow, losing the perceived tactile feel. A great example of this is the Sony Playstation Store interface sounds. It uses short piano chord shots as well as clicks and snaps.

Sometimes what you see isn’t necessarily on screen either. Take the interface for Rockband for example. The theme of the game gives fantastic parameters to create a cohesive audio soundset out of musical instruments. Scrolling will give you the sound of light hi-hat and guitar, while choosing something that will take you to another list or screen will give you the sound of a full on rock stab that has a combination of bass, drums and guitar.

Even in the mobile world, games like this require the full spectrum of frequencies. Mobile devices may not have the size of speakers to accurately represent all frequencies, but you must consider that the user might be playing with headphones or earbuds. (You could even put an opening disclaimer to wear some!)

Ear Hunt There are many places to find sounds that range from royalty free to pay per download. Many of them have meta data searches that will make it easier for you to find what you're looking for and nail that extra level of polish that your app needs.

It's easier than you think to add sound to simple projects. All it takes is a little searching and some open ears!

Audio Logos

Having an audio logo helps solidify your brand and entrench a slogan or identifier. Great audio logos are so memorable, that I could write the slogan and you'd sing the melody in your head just by reading it. Let's try this as an example:

1. Call one eight hundred twooo six seven, two thousand and one.
2. I have a structured settlement, but I need cash now...
(this one is for our American friends)
3. I'm lovin' it.
4. Why buy a mattress anywhere else?



Of course, these are all sentences and we definitely don't want to hear such a thing everytime we open an app, so let's condense even further. Audio logos aren't necessarily slogans but help us attach a visual with a sound. Here are some great examples:

1. Apple startup sound
2. Intel Inside
3. by Mennen

You may be saying to yourself, "No problem! It's just a short melody! I can make that up in two seconds!" Yes it's short but remember, you have to live with this logo for awhile, so you should make sure its encompasses at least these two things:

1. It's easy to sing back and remember
2. It can easily work across different media

It's Easy to Sing Back and Remember. This is the first step in helping the user remember the audio logo. It's called the aural approach. We hear something, we sing it back, we learn. Having a catchy audio logo should be no longer than 2 seconds, max. We're looking at approximately 1 bar worth of music. Should you learn music theory? No! That's the best part about audio logos. TOO much information is bad. You need parameters. Here are a few to start with:

- No longer than 5 or 6 notes
- Use only 1 or 2 instruments or sounds (of course there are exceptions to this ie. movie/tv production company audio logos)
- Can be played with one finger only

Sit at piano and try it for yourself. Use just your index finger and plunk away at notes. See what you come up with. Not that easy is it? But if you keep going, I'm sure you'll end up with a memorable melody that just wrote itself. You also don't have to play a different note every time your finger jumps either. You can try playing the same note twice and in various rhythms. Usually those are the most interesting. If it's stuck in your head or in someone else's head, you've completed your task!

Don't let your imagination hold you back either. Just because you haven't written a symphony doesn't mean what you have isn't a great work. Using two instruments or sounds at the same time gives your audio logo more colour and texture. The apple startup logo changed many times, with many layers of synthesis over the decades adding more variety every time, but we still know it as the startup logo.

It Can Work Among Different Media. Digital audio is here to stay and almost every device is able to play it. But what about those times when a device can't play a digital audio file? Where would Nokia be if they designed an audio logo that didn't work on their earlier phones? This may not be the case today with their devices and the platform that you would be deploying to. But what if you were asked to have your audio logo in a game or fun little short film that was in the style of 8 bit consoles from back in the day. You'll have to simplify your audio logo to emulate your melody being run through analogue chips. This is easily done if you kept your audio logo simple! Think of the limitations the Nintendo composers had back then, make the most of what you have.

In conclusion, keep it simple, keep it memorable and keep it to one finger. A little goes a long way.

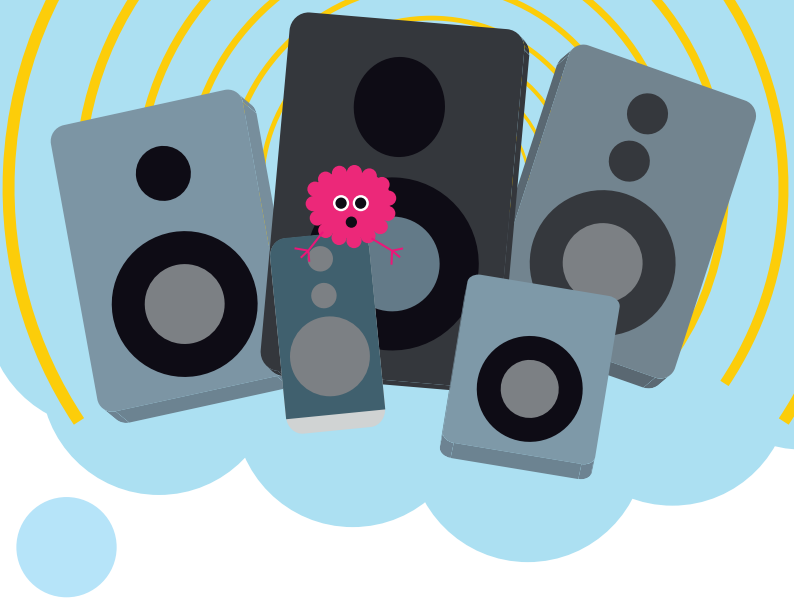
Music in Apps

Music is an excellent way to enhance the user experience. Music, as with sound, has the ability to create an environment that expands the reaches of a physical device. Music can create moods, stir up old memories; it brings an emotional connection to what it is applied to. This being the case, it is something to take full advantage of in the mobile world.

We are always thinking of how to improve our product and have great user retention. How many times have you had a tune stuck in your head for hours? Why not create a product with music that evokes a positive emotion and ultimately the desire to come back for more?

Finding the Perfect Fit. It's one thing to know that music is important to the user experience, it's another to actually bring it to a reality. This can be easily achieved by knowing exactly what you are trying to accomplish with the music and communicating this effectively.





What Are Your Options? It is necessary to consider the goals and limitations of your particular product. With this in mind, you can create a list of parameters that will allow for accuracy in your vision when hiring a composer or searching for material.

Parameters include:

- **Vision:** What is your overall theme/vision of the product? Does it take place during the Cold War? Is it a cute/happy twitch game?
- **Source material:** Will you be composing the music or will you be outsourcing?
- **Sample material:** Are there other games, movies, or songs with a similar theme?

- **Resource:** What resources do you have dedicated to music/audio?
- **Product Breakdown:** What do you want and where? Are there several menus where you would like different musical content than on a player selection page or map?

Once you have a list of parameters and sample material, not only is it easier to find material, but you can also effectively communicate these details to a composer.

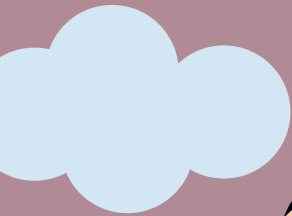
Communicating with a Composer. One of the most difficult parts of describing musical assets to a composer is the terminology. Musical terminology may be foreign to you, however there are many other ways of describing what you are looking for. Try going beyond general descriptions such as “mellow”, “exciting”, or “epic”. You can be more specific with genres and examples. For example, when you want epic, do you want Star Wars epic, Pirates epic, or Inception epic? Are you looking for hip hop, jazz, classic rock?

Also make sure to provide the composer with all of the parameters as discussed earlier as well as any visuals for your product. Once you have provided them with all details, it is your turn to trust that they will create what you asked for and more!

Finalizing Musical Content. While your musical content is in production, always make sure to listen while looking at the corresponding visuals for an accurate judgment of its effect. It is also important to recognize that because music evokes an emotional connection, it will likely affect not just you, but everybody! This is where trust in your composer comes into play, as well as the importance of maintaining objectivity. Remember that the composer has experience, which is why you hired them in the first place and can assist in reaching your target demographic and original vision.

→ VS. ←





1UP
1UP
1UP



Designing for Specific Uses

Now that we have the basics out of the way, it's time to look at specific use cases, and how to use design to get the best UX for your app.

Game Design

There are many types of mobile games. Think about the categories you find in an app store such as:

- Action
- Arcade
- Casual
- Educational
- Puzzle
- Racing
- Sports
- Strategy

While defining the game category is important for discoverability, initial game design should not dictate the final category. What we mean by that is, there are many instances where a designer starts a project with one type of game in mind and finds they have quite a different game experience in the final product. A game can cross multiple genres. When it does, it enjoys the luxury of app store placement options, allowing the game to be strategically placed in categories to maximize exposure. With that in mind, don't be overly concerned about game categorization at the start of the design process; **concentrate on creating the best game possible.**

Here are a few simple design guidelines for developing that next awesome mobile game.

Define Your Audience

Who are you designing this game for? An adult or child? A man or woman? A casual or hardcore gamer? It would be ideal to design a nirvana experience that satisfies all gamers, but it is much easier to concentrate efforts on a specific audience of gamers. You may have heard that being a jack of all trades makes you the master of none. This doesn't mean that developers can't make money from an average game that satisfies many types of gamers. However, games that have a few well designed elements generally do better in the market than games that try to satisfy everyone.

For example, take Rovio's Angry Birds game with their "3 Star" level reward mechanism. Users are awarded 1–3 stars for how well each level is played. Users advance to the next level with 1 or 2 stars, but must achieve 3 stars to really feel like they "beat" that level. This mechanism along with a super simple user interface helped achieve a homerun design the industry took note of and which is found in many other games today.

Keep it Simple

A game that is easy to play reaches a wider audience and has greater appeal. That's not to say games should be easy to beat. Game play should be straightforward, not overcomplicated and achievable allowing players to quickly get into the game and work towards its goals. Clicking through ads, sub-menus, and endless options only frustrates gamers.

Mobile games are generally played with a touch interface, so recognize the limitation of touch screen interactions: fingers obscure the screen content. Make sure touch areas are well defined, large and react intuitively to user inputs. This includes

touch gestures, like swipes, pinches, and multi-touch elements. While motion control can be a barrier in casual games, it is often perfect for games that need simple left/right control action without obscuring visual elements.

Make it Pretty

There are many good games out there that fail just because all the graphics look like placeholders for the finished artwork. Don't skimp on the look of your game. We are a visual society. You've heard of eye candy, right? Give players something satisfying to see.



Listen and Act

You love your game. Great. You should. It's your baby. But.... parents are not always the best judge of character. Continually monitor user feedback. The feedback may feel hurtful or complimentary, but whatever the comments, act on them to improve the experience. Listening can take a few forms. Comments in an app store or a support forum are one form, but game play analytics offer an in-depth way of listening to users. Reply directly to gamer comments and look to analytics to understand what's going on between-the-lines. For example, analytics may show that no one is accessing that game feature you thought was integral to game play, while everyone is using a feature you had no idea was important. For a successful design, listen to the audience and give them what they want.

Be Different

What makes a game different than all the others? Is it the look? Is it the storyline? Is it a social or MMO element? Make a concerted effort to stand out from other games in your category. For instance, the mobile game *Limbo* by Playdead has a wonderfully dark black and white visual style that matches morbid puzzle play. Rockstar Game's *Grand Theft Auto: Vice City* for mobile has a great soundtrack, but it also uses tactile feedback effects to bring their Rockstar console game feel to their mobile Android titles.



Design for Mobile Checkout and Payment

Here's a simple test. Take one of the business cards you have laying around at your desk. Judge its size. What you're seeing is roughly the frame size your mobile users have available to view your entire mobile site.

But when it comes to the mobile payment and checkout experience, they have even less space available, **as the touch keyboard will take up close to 50% of the available screen space**. And if they are in landscape mode, the **landscape keyboard will take up 70–80% of the screen**. Wow!

Core at the issue is an extreme lack of page overview. The user simply gets “lost in the page”—especially when filling in forms.

Therefore, to account for this lack of visible context and page overview in your mobile checkout and payment design, closely consider the following:

- As users lose the overview of a page, they often tend to focus on each form field as a separate task to complete, and not so much a part of a large whole (as they often do on desktop sites). Thus, it's very important that all fields labels can be read and understand when read completely out of context. For example: a label shouldn't simply read “Phone:”, not even if placed within a “Billing Info” section header, but instead in should be completely context independent and read “Billing Phone:”. Or “Gift Certificate Pin” instead of “Enter Pin”.
- Furthermore, any help and clarifications must be placed exactly where the doubt can occur. This makes inline help such as tooltips and form field descriptions vital.
- To simplify the initial page impression and set the right expectation of what the user should do at this given step, consider using the principles of progressive disclosure and

dynamically collapse content. For example at the account selection step, instead of having 3 separate headers for “Sign-in”, “Create account” and “Guest checkout” each with their respective fields shown, simply collapse all content and only show 3 clickable headers. This enables the user to get a complete overview of all their selection options. See example in section #6: uxdesign.smashingmagazine.com/2013/05/21/recommendations-mobile-commerce-web-sites/

- Typing on a small touch keyboard is both slow and very prone to errors. So whenever technically possible auto-detect and auto-complete as much data as possible. For country selection this can be IP geo-targeting or at the very least country auto-complete. Here’s a free plugin: uxdesign.smashingmagazine.com/2011/11/10/redesigning-the-country-selector/
- Furthermore, in the US and many other places, city and state/province name can be auto-detected based on the user’s ZIP or postal code - so simply ask for that first and pre-fill their State/Province and City fields. Two free plugin’s exist: daspecster.github.io/ziptastic/demo.html and zippopotam.us/

Besides the specific recommendations above, always make sure you test your designs on real end-users. There are always some site and audience specific hiccups you’ll never discover in any other way.



kerching



kerching



kerching

kerching





Testing and User Feedback

Types of Testing

To begin, here's a quick explanation on the most common types of tests you can perform with your projects:

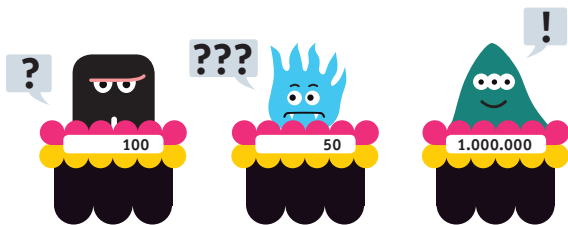
- **Installation Testing**—tests to ensure that your app installs onto the target platform(s)
- **Compatibility Testing (Device Compatibility Testing)**—tests to ensure your app works with the target platform(s) and its' typical environment
- **Smoke Testing (Sanity Checking)**—a quick functional test to make sure that the app runs and basically works in a variety of different use cases
- **Regression Testing**—ensures that an update or fix hasn't disrupted previous results
- **Performance Testing**—tests to ensure that app speed, data use and power consumption are acceptable
- **Usability Testing**—tests focused on the user interface and user interaction with the app.
- **Security Testing**—tests concerning the handling of personal or secure information
- **Internationalization and Localization Testing**—tests for language and styles across the territories targeted by the app. (hint: Never trust automatic translation...)
- **Accessibility Testing**—tests for access by disabled persons (The W3C is a good place to get more information on this).

Involve Users Early and Often.

User feedback is an essential part of interaction design. Involve users as early as possible. As soon as you have any feature functional in your app where showing it to other people is practical, show it to someone—your colleague, your roommate, a friend. Try to remain open to anything they might say. Watch their face as they use the app. Does it light up? Do expressions of confusion flash across their face? If so, when does this happen?

Also get verbal feedback. One common method to use is “think aloud method” which is really just asking the person to talk through what they are doing and thinking as they use the app. You will be surprised what surfaces.

It’s important to gather feedback without biasing your users. In other words don’t “show them” how to use the app. Just put it in front of them with minimal explanation and soak up their feedback like a (quiet) sponge. If they’re hopelessly stuck, say, on the login screen, and you really want to get their feedback on the home screen, then you could prompt them to get through a tough spot. But keep this to a minimum, and in the spirit of moving the interview forward.



User Testing Top Tips

Most user testing happens too little, too late and for all the wrong reasons. By following these simple tips you can add an extra weapon to your arsenal which will set you apart from the competition:

Test Early

After their first user test most teams exclaim: “Why didn’t we do this sooner?” So it’s never too early to start testing:

- Ask users to try competitors’ products before you start working on your own. Observe their reactions and record their comments to identify pain points and unmet needs.
- Test very early versions of your own product, using sketches on paper or wireframe mockups. Don’t be embarrassed to show incomplete or broken versions, the whole point of testing is to prevent you from getting down the wrong path before it is too late, risking delays and higher costs!

It’s much harder to change a product once it’s in use. By testing early, you avoid arguing endlessly with your colleagues about what is the “right” design and having to redo things later.

Test Often

Testing is an iterative process: you make a prototype, you ask people to use it while you observe them, you identify problems, you try to fix them, and then you test again.

- It’s more valuable to run tests with a small number of participants (typically 3-5 per iteration) frequently throughout the development process than doing one big test with tens

of participants near the end when most crucial decisions have already been made.

- As you do this frequently you'll reach a stage in which you will feel that the issues that you are now identifying for a particular interaction or even the whole product are minor (or at least much less severe than initially). It is fairly usable and you can now proceed to writing production code, or do some Beta testing in parallel to user tests.

Keep It Simple

Testing shouldn't be a big deal. This will make sure that you do enough of it! Start right now by asking a couple of friends to try out what you're working on in front of you while you take notes. Sketch on napkins if you don't have an interactive version to test:

- Don't try to sell the idea to them. Just show them the sketch and ask them to tell you what **they think you are working on**. If they can get it from the sketch you are on the right path! Then ask them to complete a typical task (e.g. book a hotel room for the weekend) by going through the sketches. Observe their reactions: Does the sequence of screens make sense to them? Is a step missing? Did they get lost? Any problem that you identify now will save you from a lot of effort later.
- Let them do the talking. Try not to guide them or become defensive if they run into trouble. Try to put yourself in their shoes instead of explaining to them what they should be doing.
- Postpone screen-recording and hiring a usability lab with a one-way mirror for later (after you've run a few sessions and have become convinced that you need a more sophisticated set-up).

- Your recruiting process should be equally simple. Although you want to make sure that the participants are not entirely unlike those whom your product targets, don't bother too much about their demographic details. E.g. if you are working on a game app don't rely only on input from your game-freak colleagues or on feedback from friends who have never played a game. Publish an advert on a social network and look for a few other gamers (both men and women) to try it out. Often free pizza and beer combined with the opportunity to play a new game will work wonders. If you test frequently, you'll be able to meet a good chunk of your target audience over time.
- Debrief with your team immediately after the test to identify the five most important observations and what you'll be fixing for the next version.

The only training you need to run such a user test is to read one of the brief How-to books listed below and then start doing it. You'll get better as you practice and you'll see its value immediately.

There is No Substitute

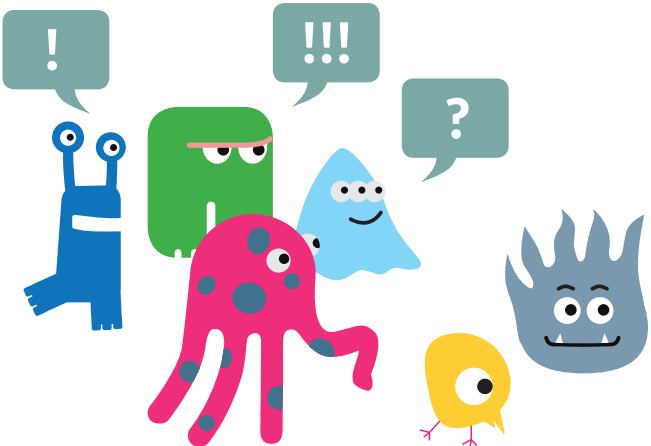
There truly isn't a substitute for user testing:

- After you've worked on a product for a while, you know how it works and what it is supposed to do. The best way to see it with fresh eyes is to ask someone else to use it while you are watching them. In fact, this may be your only opportunity to directly observe people using your product in order to understand whether they can figure out what it is, complete the most typical tasks with it and enjoy using it.

- Most tests are done to settle internal debates about very specific issues (e.g. should we use a search box or a pull-down menu here?). But they end up revealing much larger holes in the design. This helps the members of the team realize that not everyone thinks like them and that a lot of things that they take for granted are not obvious to everyone else.

Triangulate

User testing is a valuable source of information but it isn't the only one. Based on your experience, professional judgment and common sense you will use the input from user tests to make design choices with greater confidence. If you value evidence more than intuition and accept that your product is more likely to be successful if its design is based on trying to understand your intended users rather than relying on a rare moment of inspiration or working in isolation in your garage, then user testing will become a powerful and valuable guide for your decisions.



Here's some further readings on testing:

Steve Krug, *Don't Make Me Think: A Common Sense Approach to Web Usability*: Chapter 9 covers all the basics of usability testing in a fun and concise way. Also look at *Krug's website* for a sample script and more expert advice.

Saul Greenberg and others: *Sketching User Experiences, The Workbook*: Chapter 6 introduces user testing with sketches and other ways to get feedback before it is too late!

Karen Holtzblatt and others: *Rapid Contextual Design*: Chapters 13 and 14 prepare you to test with sketches in the actual environment in which people are meant to use your product. It's ideal for Agile teams interested in getting out of the usability lab to immerse in their user's experience.





WOW!

incredible!

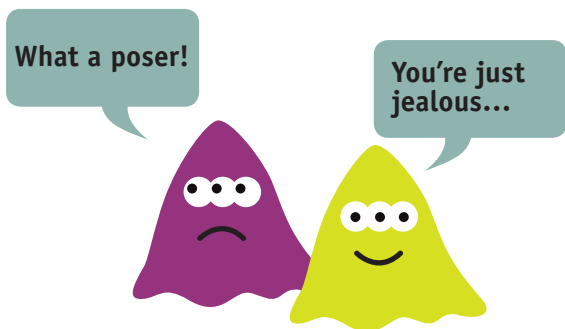
looks so real!


Conclusions

We hope this has given you a good idea of how important it is to consider design, user interface and user experience to the success of your mobile application.

There are some tough problems, like hardware fragmentation, touchscreen issues, platform incompatibility and mobile latency. But with proper planning and knowledge of the requirements and limitation of the various devices and operating systems; you should be forewarned and forearmed to create the experience your users will love.

Like the 5th Dimension, good design and usability is hard to define and even rarer to conquer but is truly vital to make a product or application truly great.





ooooh yeah!

more is more...

Contributors

As with the other Mobile Developer's Guides, the 5th Dimension is a community project. A big **thank you** to all the folks who contributed their time and expertise. Let us know if you would like to contribute to the next edition.

Allen Pike / Steamclock Software

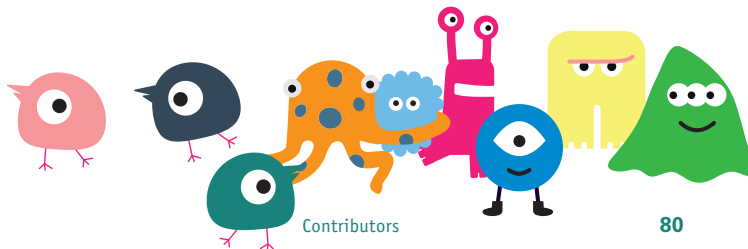
Allen Pike runs Steamclock Software, a mobile development studio in Vancouver, BC. His team focuses on designing and developing high quality apps, which have been featured on CNN, Daring Fireball, and Popular Science. Prior to founding Steamclock, he was a Software Engineer at Apple. He has also taught at Simon Fraser University, and plays on a hockey team comprised entirely of programmers.

[@apike](#) / www.steamclock.com

Anna Alfutka

Anna is a designer and illustrator passionate about software products. Based in London, she worked on UX & UI for projects in telecom, publishing and e-commerce; as well as her own software ideas.

www.alfutka.net



Ariana Bliak / A Thinking Ape

Ariana Bliak is a Music and Audio Designer at A Thinking Ape Technologies. Attending Selkirk College and the University of Lethbridge, she has background in both contemporary and classical music. Specializing in keyboard performance and composition, she has experience in composition and sound design for games, film scoring for short films, composition for commercials, and live performance. In her current position at A Thinking Ape, she is dedicated to enhancing the mobile gaming experience through music and audio.

[@athinkingape](#) / www.athinkingape.com

AquA

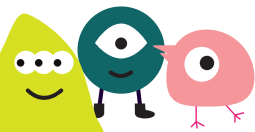
“App Quality Alliance (AQuA) is an independent, worldwide, non-profit mobile industry body funded by the members for the benefit of the industry. Its aim is to help the industry increase the quality of mobile applications. The Core Members are AT&T, LG, Motorola, Nokia, Oracle, Samsung and Sony Mobile.”

[@AppQuality](#) / www.appqualityalliance.org

Bill W. Scott / PayPal

Bill Scott is the Sr. Director of UI Engineering at PayPal. In a past life he co-created one of the first successful Macintosh games (GATO, 1985), built & designed wargaming interfaces for NATO, led user experience teams (Sabre, Meebo), co-wrote one of the first Ajax/JavaScript frameworks (OpenRico), managed user interface engineering organizations (Netflix, PayPal) and published a design pattern library (Yahoo!). Bill is also a frequent speaker at conferences & workshops worldwide as well as the co-author of the O'Reilly book Designing Web Interfaces.

[@billwscott](#) / www.looksgoodworkswell.com.



**Peter van der Linden, Bob Heubel, David Birnbaum,
Grace Franz, Suzanne Nguyen / Immersion Corporation**

Immersion Corporation is the world leader in Haptic technology that enhances the user experience in ways which is satisfying, intuitive and fun. The Immersion contributors have nearly a century of collective industry experiences. Bob, Sr. Haptic Technology Evangelist, is well known as the Haptic Effects Design Guru. You may have felt some of Bob's work in Rockstar Games' Grand Theft Auto III & Max Payne. Peter, Chief Android Technology Evangelist, is an award winning author for several programming books in Java and C. David, UX Design Manager, has garnered 20 patents in the fields of user experience, gaming, medical devices, mobile, and wearables. Grace, Marketing Specialist aka Graphics Design Genius, creates all the art works and designs for Immersion's advertisements, banners and collaterals. Suzanne, Director of Developer Relations, is well known for building various technology developer programs (Java and Android).

[@HapticsDev](#) / www.immersion.com

**Caroline Lewko, Carlo Longino, Rod Burns,
Sam Chan / WIP**

The WIPsters are based in Vancouver, London, and Austin and use their knowledge, charm and good looks to connect mobile developers to people, information and resources to increase innovation and market success. We also build mobile developer communities for many of the leading developer programs. Find us at our events such as WIPJams, Muthers!, DroidconUK; and check out our website for some great resources, developer tools, developer spotlights, and our WIP Calendar of the greatest developer events on the planet.

[@CarolineWIP](#) / [@Caaarlo](#) / [@rodburns](#) / [@anothersamchan](#) /
[@WIPJam](#) / www.wip.org



Christian Holst / Baymard Institute

Christian Holst is a usability engineer and has worked with designing user experiences in both the credit card, hearing aid, and consulting industry before co-founding Baymard Institute 2009. At Baymard Institute he conducts large scale web usability studies within e-commerce and mobile commerce—currently supplying industry research reports to 72% of all Fortune 500 companies within e-commerce, as well as publishing free bi-weekly articles on e-commerce and m-commerce research.

@baymard / www.baymard.com

David Fay / Mobify

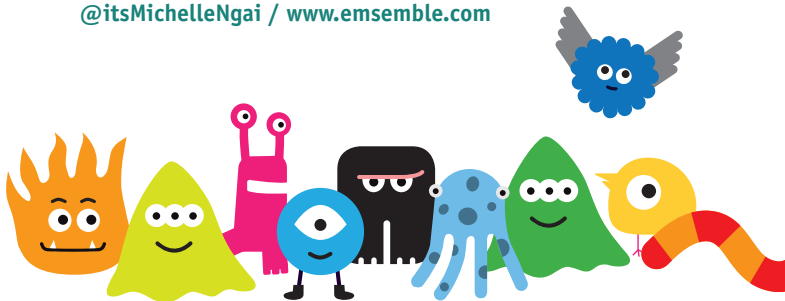
David's mission is to share the amazing stories that result from Mobify's research and development. He loves reading and writing about emerging trends in the responsive and adaptive web.

@mobify / www.mobify.com

Michelle Ngai / Ensemble Software

Michelle is a User Experience Designer at Ensemble Systems. She has a degree from SFU in Interaction Design and also studied design in Italy. She has also volunteered as a graphic/web designer and teacher in Peru. Since then she's also worked at various companies including a start up, ad agency and digital agency doing UX design across many different platforms.

@itsMichelleNgai / www.emsemble.com



Dr Nikiforos Karamanis / SwiftKey

Nikiforos is the User Experience Lead of SwiftKey where he designs, prototypes and evaluates ways to make typing on smartphones and tablets an even greater experience.

@uxniki / www.swiftkey.net

Peter Ma / adap

Peter Ma is the founder of adap, a mobile advertising platform. He has worked with multiple startups and started two of his own companies. He is currently residing in San Francisco working as an independent consultant that builds iOS, Android, and HTML5 applications.

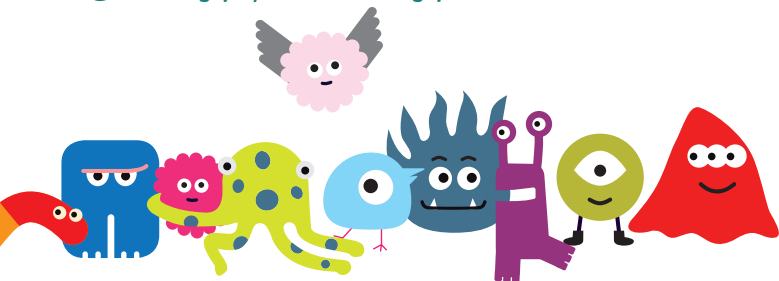
@nyceane / www.adtap.com



Terence Mazon / A Thinking Ape

Terence Mazon is the Music and Audio Director at A Thinking Ape (Gangs At War, Meego Village), one of the highest-grossing mobile developers in the world. Terence's background in professional DJing (he once won the prestigious West Coast DMC Championship) and latin percussion helps him push innovation in sound quality and production value in mobile games. His passion for recording started early: at the age of 6, he recorded his own talk shows on cassette tape, and he's been DJing and multi track editing since he was 12.

@athinkingape / www.athinkingape.com



APPS HAVE FEELINGS TOO*

Download **Immersion's Haptic SDK** and feel your apps come alive!
www.immersion.com/haptic/sdk

TOUCH me and
Feel me ROAR!



HAPTIC MUSE is a free animated effect preview app that illustrates the Haptic SDK's library of 124 pre-designed haptic effects in a gaming context.



immersion.



30 Rio Robles, San Jose, CA 95134

developer.immersion.com | HapticsDev@immersion.com



[/ImmersionDeveloper](https://www.facebook.com/ImmersionDeveloper)



[@HapticsDev](https://twitter.com/HapticsDev)



blog.immersion.com



A community project led by:



www.wip.org



immersion.

developer.immersion.com



**The Companion Guide of the
MOBILE DEVELOPER'S GUIDE TO THE GALAXY &
MOBILE DEVELOPER'S GUIDE TO THE PARALLEL UNIVERSE**