Learning iOS Design - A Hands-On Guide for Programmers and Designers
William Van Hecke

Design is hard work. It takes relentless dedication to detail and a ridiculous work ethic to craft software and interfaces that enable people to do great things. Not only should software and interfaces be beautifully designed, they should also be intuitive to use and a pleasure to work with. It's all about the little things that distinguish a good experience from an exceptional one. Think of the things your software enable to do. Like in Human Engineering, the software that we build enable Human Factors Engineers to do their research, which help define the revolutionary and innovative products Apple sells, which in turn enable people around the globe to accomplish incredible things.

When the SJ introduced the iPhone in 2007, Steve Ballmer predicted that there would be no chance it would get any market share. Joke on you, Microsoft. The iPhone became an icon of a redefined smartphone. Instead of extended feature bullet points and legacy system support, the iPhone and iOS focused on design - and that arguably is what put Apple on top. Design is cultivating a relationship with the users - making sure they have the best user experience possible. Design is an art. Design is a science. Design is neither. You won't always find all the definitive answers to a design problem and you will probably never finish. But that's okay. SJ used to say Apple is at the intersection of science and liberal arts. Sometimes you need all the usability metrics; that's the science. Sometimes you need the aesthetic appeals; that's the art. Which one is better? It depends.

Turning Ideas into Software

- Software is complicated. Don't draw the entire mental picture in your head; throw it on something you can see
- The design process goes something like this: outline -> sketch -> wireframe -> mockup -> prototype. There needs to be some order, but they don't need to be linear. In fact, a more chaotic order is encouraged
- Understand what the app is for, who the audience is. Also think about anti requirements: who is this app definitely not for? Who isn't part of the audience? Figure out *what* the app does first and *how* later
- Try squashing repeating features or features that try to accomplish too much

- Always sketch out the ideas; that's where conversations start. It only takes a moment, so don't be lazy. Remember you're sketching, not wire framing yet, so don't let the perfect straight lines slow you down. Sketches don't address actual colors, textures, or other styling. The sizes aren't exact and the wording can just be "lorem ipsum"
- Sketches are also a good way to prove a point or to decide which design is superior
- The workflow is arguably a more important design problem at the sketching stage, since you don't want to go down a rabbit hole of designing something so convoluted and so complicated to use. A sketch with ugly individual screens and an intuitive workflow is almost always easier to fix than the opposite

- Navigation from screen to screen
    - NavigationController is the most common way to navigate iOS apps. Since 2001, users have been trained to look for the navigation bar to locate where they are and how to go forward/backward with the iPod
        - Very good for presenting tree-like hierarchy, but not when there are too many levels
        - Navigation items must look tappable
    - Split View provides a way to flatten the navigation hierarchy by displaying content and navigation side by side
    - Tab View: good when you have 3-4 screens that need to be always present. Beware when you have 5+ screens; it's going to haunt you if you make users decide which screen is more important
    - Model View: temporarily disables navigation. This view is attention-hogging and can be annoying if the users don't think it is worth their attention
    - Popover: iPad-specific, tooltip-like view. Convenient for small, infrequently-used bits of data. Has a very lightweight feel to it
- Native elements
    - Status bar (20 pts): don't hide it unless an immersive experience is more important than utility (like time and battery)
    - Navigation bar, toolbar (44 pts on portrait, 32 on landscape)
    - Tab bar (49 pts): always will show at the bottom
    - Alert: the easiest way to drive your users crazy. Use them with great caution
- Go buy an iPhone if you're not familiar with iOS native elements. You don't necessarily need to know their technical names, but at least develop a feel for when to use (and not to use) which and how

- Wire frames are more defined than sketches. Most of the elements are represented proportionally and their importance are also weighted via shades or sizes. Wire frames are not yet interactive
- 1 point is 1 pixel on regular screen and 2 pixels on Retina displays. Resolution shouldn't affect your wire frames

- Gestalt psychology -- a popular psychological framework that describes how humans perceive things: the human mind forms a global whole with self-organizing tendencies. Use this underlying framework when crafting your layout
  - Unity is the goal. The funny thing about unity is that if you do it right, people will NOT notice. Since the human mind tends to group/categorize things together, the users don't feel challenged when your app is well organized and united. Their brains work naturally. However, if the opposite happens, the human brains can pick out the mistakes very easily
  - Some elements are ought to stand out for good reasons. The best way to achieve that is to use visual weight, which is defined by relative size and background contrast. The bigger an element is and the more it stands out from the background, the more visual weight it carries
  - Similarity and distinction: things that look similar should do similar things. Things that stand out better do because of good reasons. People look for reasons why things stand out
  - Proximity and distance: closer elements are implied to carry similar functionalities and vice versa
  - Alignment:
    - Edge alignment works best for rectilinear (box-like) elements
    - Center alignment works best for irregular shapes like blobs or text or bare graphics
    - Align text to other text rather than to other boxes
  - Rhythm:
    - Use visual rhythm to communicate a sense of unity and order.
    - Basic scale: 10, 20, 44, 52: normal distances, extra separation, bigger spaces, and humongous spaces
    - Modular scale: 7, 10, 14, 20, 28, 40, 56, 80,... this is the result of doubling every 2 iterations.

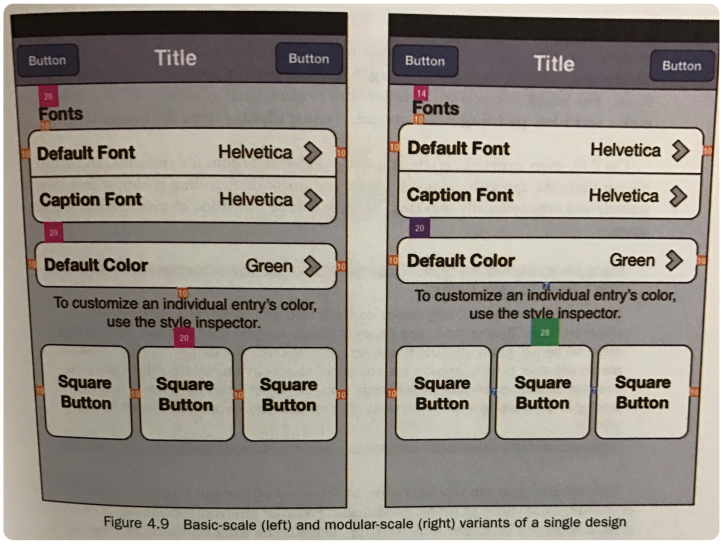
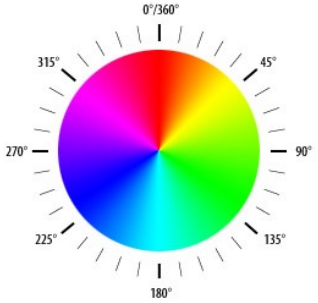


Figure 4.9 Basic-scale (left) and modular-scale (right) variants of a single design

  - Margin and padding: white spaces help make the app feel modern and breathable. Scrolling is almost cognitively and ergonomically free, so don't be afraid to add white spaces
    - Let the height of the capital letter E be x
      - For a single text line, give about 100% of x as vertical padding
      - For multiple lines, give about 50 to 100% of x as vertical padding
      - Horizontal padding should also be about 50% to 100% of x
      - Images can have a little bit less padding than text. 25% - 50% of x is good
  - Don't always rely on Xcode's Interface builder to position things for you
  - Use margin, padding, white spaces, scales, whatever, to separate and organize your elements. The key is not to include things you don't need. That sounds obvious, but can easily be violated. Up your spacing scale instead of drawing a border line to separate things.
- Typograhy
  - Almost all of the time, the system font (Helvetica Neue and San Francisco) is all you need. It's a fantastic all-purpose font given the size and resolution of iOS displays. It's fine if you need a fancier, more stand-out font for your logo. Or when your context is text-heavy and you want to ensure a font that is meant for reading, then Georgia and Verdana are solid choices
  - Use real characters, not ASCII approximations:
    - *option* + *]* for ‘ instead of '
    - *option* + *shift* + *]* for ’ instead of '
    - *option* + *[* for “ instead of "
    - *option* + *shift* + *[* for ” instead of "
    - *option* + *minus* for – (en dash) and *option* + *shift* + *minus* for — (em dash) instead of --
    - *option* + *;* for … (ellipsis) instead of ...
  - Columns of the correct width (about wide enough to type the alphabet once or twice) are easier to read than two-wide ones (more than 2 alphabets wide) or one very narrow one (less than one alphabet wide). If column is not an option, use more space between the lines

    - Left-alignment for content text. Center-alignment for titles and headlines that need to stand out
- Physical heights
    - Unless you're hiding the status bar for especially immersive experience, it'll always eat up 20 pts
        - Status bar grows to 40 pts when there's a phone call or voice recording or navigation session is active
    - Keyboard (162 pts on iPhone landscape, 216 pts on iPhone portrait, 352 pts on iPad landscape, and 264 pts on iPad portrait) will need to be accounted for
        - Some languages like Japanese need a completion bar to support text input, so add 36 pts to the keyboard on iPhone and 54 pts on iPad)

- Mockups is the final frontier. Strive for pixel-perfect if possible, but don't get too invested or attached to it. You can easily overlook the flaws because the mockups look so good. For simple screen, mockups phase can be skipped
- "It's not just what it looks like and feels like. Design is how it works." – SJ
    - People often mistake picking color schemes and putting together shiny buttons for the sole definition of "design". Those are merely specific phases of design
- You could be the most qualified person for the job, but do people perceive you when you walk into an interview with pajamas on? Or you could dash in with a custom fitted suit but you have absolutely no qualification for the job you're interviewing for. Software is the same: it should look good AND be functional
- Styling
    - Rendering: How all the elements look on the app, down to the smallest pixel. If done well, this can show how much the designer actually care for the app
    - Communication: What do the elements' appearances say or imply? What feelings do they provoke?
- RGB doesn't make sense psychologically. For example, if you want to brighten a dark orange, it doesn't intuitively make sense to add red and green at a 2:1 ratio.
    - HSB works closer to how your mind does
        - Hue, expressed in degrees from 0 - 360, represents a position around the color wheel
        - Saturation, expressed in %, represents how much hue is present. Higher % means brighter and more vivid colors, while lower % (desaturated) look more muted and gray. At 0%, it's grayscale
        - Brightness, expressed in %, represents how much light the screen is emitting. 0% is black



    - Humans, for evolutionary reasons, can't see the subtleties of blue hues, but can detect red- and green-hues very well
    - If you intend 2 elements to have different hues but similar value, nudge the brightness or saturation up or down to keep the value consistent
    - W3C's formula to calculate perceived brightness: (299*R + 587*G + 114*B) / 1000
- Give the elements on your screen great contrast to create clear contours. Although the human brains can detect camouflaged elements, don't strain them
    - Every element has the anatomical components: the border, interior, and the contents. The more contrasting these components are with the background, the more readable the element is
    - The more internal contrast, the more weight the element has
    - The more saturated the color of an element, the more weight it has
    - Most important of all, whatever you do, don't distort or weaken the contour of the elements
- Backgrounds
    - Good backgrounds should have low internal contrast
    - If you use real photos as background, better make sure the internal contrast is not too much. Achieve this by idealizing the texture. For example, Apple's old iBook wooden texture was idealized. Real wooden texture looks way more complicated and contrasting
- Posterize
    - Change your entire interface to only black, white, and 50% gray to see how readable it is still

- Prototyping
    - There are many options to create software prototypes: videos, paper interactive cutouts, etc. I prefer just opening up Storyboard and wire the damn thing up. It doesn't need to be perfect; it's a prototype. At Apple we move really fast, so going through all these formal steps won't be feasible. I find Storyboarding the easiest and most time conscious approach to prototyping
- Usability testing

    - Tell the participant that whatever he/she does, it's the app's fault
    - Have the participant think aloud as they walk through the app
    - Resist the urge to explain/direct them when your design screws up and they can't achieve a task
    - Take good notes

- Psychologist Donald Norman identified 3 levels of cognitive processing that are related to design, the first of which is called the visceral level (both the unconscious, automatically, mechanical interaction). This is the “gut feeling” when you first interact with a design. The goal is to make that graceful. Note that a design can be graceful and useless at the same time
- iOS provides a very magical experience that users take for granted. It’s magical when you flick a finger or tap on a piece of glass, the software underneath responds accordingly. Let it be magical and forgotten that way because the moment they realize it’s not, your design is 100% shit
- Provide instantaneous feedback to touch inputs to maintain that magical illusion of fast software and minimize the moment of uncertainty. 500 ms or less will get you that instantaneous feel.
    - Make sure feedback is not covered in the hand shadow areas
    - Stick to the standard gestures for control. Better have a good reason why users need 2 fingers tap to do something
    - Hysteresis: an area around a tap that the finger can stray into without becoming a drag
    - Make the tap target as generous as possible. 44x44 pts is the suggested amount
        - This area could be smaller for dangerous actions like buying things or deleting stuff

- A graceful interface makes the users feel good about the software. A gracious interface makes users forget about how easy it is to do things with the app (i.e. they don’t constantly think about how to do a certain task). This aligns with Donald Norman’s 2nd level of the 3 levels of cognitive processing: the behavioral level
- Beware of wording connotation in the context of different cultures.
- Affordances, or interaction cues, should be used to help conveying meaning and subtle directions
    - The appearance (shadow, gradient, highlights) suggests a button can be tapped
    - Horizontal grooves on table cells suggest it can be rearranged
    - Imagery affordances:
        - Use system icons because those have been tested
        - At least use the same object in those icons if you want to use your own
    - Textual affordances
        - Sometimes a word is more valuable and expressive than a symbol. If that’s the case, use words even though symbols and icons feel cooler and more elegant
        - Words are cognitively and visually loud. Use with caution
        - Use plain English. Keep it as concise as possible
- Defensive design
    - Users make mistakes. A lot of them. So design a fool-proof system where recovering from mistakes isn’t so bad

- The last and top layer of Donald Norman's 3-layer cognitive model: the reflective layer. This is the most nebulous layer. It's hard to write or talk about and it's also the most important. The reflective layer dictates what the users thinks and feel when the app has been closed and the iPhone/iPad has been put away. How did the app or experience make their lives better, in whatever small way?
- Remember that you are designing your software to enrich people's lives. Each and every app should get the users a step closer to whatever goal they hope to achieve. It doesn't matter how small or big that step is, as long as the users' lives are enriched by your software, you've done your job well
- Resist the temptation to include tutorial screens. They're just as intimidating and unwelcoming as the interface the tutorial was meant for. Instead, create an interface that triggers a sense of adventure. You can then send help along the way, bits by bits. The best kind of tutorial is the interactive one, where users can follow along at their own pace. This is harder to do than a static page type of tutorial, but it's well worth the effort
- Beware of localized string length. A paragraph written in English can be easily 50% longer in character-rich languages like German or Dutch and can be substantially shorter in Japanese or Korean, in which long English words can be written in a single character. Design your interface to accommodate this if you're targeting an international audience

- An app can be
    - Focused: functionally simple
    - Versatile: functionally complex

    - Presented quietly: appears simpler
    - Presented forthcomingly: appears complex
    - Just because an app is presented quietly doesn't mean it's not versatile
- iOS apps can be focused: do one one thing and do it very well
    - Preserve energy and recourses to craft beautiful UI and pleasant UX
    - Is not as useful as versatile apps
- iOS apps can be versatile, because versatile iOS apps are still more focused than their macOS counterpart
    - Will take lots of patience and resources to come up with the right design

- iOS is the perfect place for quiet apps because you don't have a lot of screen real estates. Your experience usually follows a tree-like, hierarchical path with very few branches and many levels. You don't have a lot of options at once, creating an impression that the app is "lightweight" and not as featureful, that's not necessarily true
    - Stacked in time is the term used to describe how your navigation elements are not all seen at once. As you tap on one element, many more appears depending on the context. This is the opposite of "adjacent in space", where a lot of controls are displayed side by side and appears kinda like a command center
    - Progressive disclosure: only show more items and options when the user has explicitly asked for them
        - Group the items by meaning and arrange by importance
- Taps are cheap, so set the number of taps required to do something accordingly
- Hide instead of disabling rarely used options

- Add friction to dangerous/alarming task can be very helpful in preventing users from doing something disastrous. Software can make people very happy. It can also make people equally upset, so better safe than sorry
    - Add more step, require more gestures, or add heavy animation to make the point
    - Make sure your friction is intentional and logical
    - Streamlining and reducing inputs are almost always good; they create friction
- Provide defaults that make sense
    - Make sane assumptions to avoid asking the users to many questions
        - Ex: automatically save a document when the user creates it instead of asking to save or delete at the end. When they want to delete it, they will. Don't ask
    - One option is another good choice when you require the user to explicitly start doing something
    - If more than one options is absolutely necessary, provide pre-filled values and rank the most useful, anticipated ones on top

- "By all means leave the road when you wish. That is precisely the use of a road: to reach individually chosen points of departure" -- Robert Bringhurst on following design guidelines
- Specialization - top layer
    - Trends - consistent with some other apps
        - Conventions - consistent with many other apps (and maybe other platforms)
            - Guidelines - consistent with all other guidelines-compliant apps. This is the foundation
- HIG - iOS Human Interface Guideline. Every good iOS designer reads this document and uses it as a foundation. Specialization is the unique piece that they bring to the table while still sitting on the time-tested foundation
- Consistent designs mean every elements behave in a uniform manner
    - All animations flow the same direction when performing similar tasks
    - All buttons, links, or clickable elements have the same tint color
    - Equally-important elements have similar visual weight, etc
- Avoid the cargo cult design
    - Anthropologic term coined in WWII used to describe Pacific Islanders who made contact with the outsider world for the first time and mimicked their advancements without understanding why
    - Don't adopt a cool design without understanding why it is so cool in the first place and why it works/doesn't work with your software
- "Good artists borrow; great artists steal". If you're merely good, you'll take something that someone else came up with and use it for your own purposes. If you're great, you'll recognize the bets ideas and make them your own by building them up to their true potential, far beyond what the originators imagined
- Completely specialized designs take exponential amount of effort to create relatively to their completely consistent counterparts
    - Mediocre completely consistent apps are more delightful than completely specialized apps, so dread the water carefully

- Rich vs. plain
  - Color vs. monochrome
    - Use hue
      - Blue and green: safe and encouraging
      - red and amber: elicit caution
      - Brown and beige: organic
      - Blacks and grays: industrial and cold
      - Safe hues include blue (215°), hueless blacks, grays, whites
    - Avoid totally saturated colors
  - Depth vs. flatness
    - Skeuomorphism: mimicking real world counterparts - feels very heavy in modern app
    - Flat UI: hell yeah
  - Realism vs. digitality
- Don't worry about people who are just upset because your vision is different from theirs, but listen to constructive complaints to help refine your vision