UI design principles

(extract from iOS Human Interface Guidelines)

Main principles

- **Deference:** "The content is boss!". The UI helps users understand and interact with the content, but never competes with it.
  - Let the content utilize the whole screen
  - Gradients, drop shadows etc. sometimes lead to heavier UI elements that can overpower or compete with the content. **Instead, focus on the content and let the UI play a supporting role**.
  - A translucent background help users see that more content is available

Main principles cont.

- **Clarity:** Providing clarity is another way to ensure that content is paramount in your app.
  - **Text is legible at every size**, icons are precise and lucid, adornments are subtle and appropriate, and a sharpened focus on functionality motivates the design.
  - **Use negative space** ("air") - Negative space makes important content and functionality more noticeable and easier to understand. Negative space can also impart a sense of calm and tranquility, and it can make an app look more focused and efficient.
  - Let color simplify the UI - A key color highlights important state and subtly indicates interactivity. It also gives an app a consistent visual theme.

Main principles cont.

- **Depth.** Distinct layers can convey hierarchy and position, and help users understand the relationships among onscreen objects.
  - Visual layers and realistic motion impart vitality and heighten users' delight and understanding.
  - Using a tranclucent/diffuse background separates objects from the background and makes them "pop out".

Start - stop

- **Try to avoid displaying a splash screen or other startup experience.** It's best when users can begin using your app immediately.
- **Avoid asking people to supply setup information.** Instead:
  - **Focus on the needs of 80 percent of your users.** If there is functionality that only a few users might want—or that most users might want only once—leave it out.
  - **Get as much information as possible from other sources.** If you can use any of the information people supply in built-in app or device settings, query the system for these values; don't ask people to enter them again.
  - If you must ask for setup information, prompt people to enter it **within your app – when it's needed.** This way, people aren't forced to switch to Settings before they get the chance to enjoy your app.

Start - stop login

- **Delay a login requirement for as long as possible.**
  - It's best when users can navigate through much of your app and use some of its functionality without logging in. Users often abandon apps that force them to log in before they can do anything useful.
  - If users must log in, display in the login view a brief, friendly explanation that describes the reasons for the requirement and how it benefits users.

Start – stop onboarding

- **Try not to use an onboarding experience (start-up guide).** Onboarding is not a substitute for good app design. If you still feel that onboarding is necessary, follow these guidelines to create a brief, targeted experience that doesn’t get in the user’s way.
  - **Give users only the information they need to get started.** If you give too much information, you make users responsible for remembering details they don't need right away and you may send the message that your app is hard to use. Provide additional help only when needed.
  - **Use animation and interactivity to engage users and help them learn by doing.** Add text sparingly; don’t expect users to read long passages. As much as possible, avoid displaying screenshots of your app because they’re not interactive and users can confuse them with app UI.
  - **Make it easy to dismiss or skip the onboarding experience.** Be sure to remember the choice users make and don’t force them to make it every time they open your app.

Start – stop cont.

- **In general, launch in the device's default orientation.** If your app runs *only* in landscape orientation, you should always launch in landscape and let users rotate the device if necessary.
- **If possible, avoid requiring users to read a disclaimer or agree to an end-user license agreement when they first start your app.** Instead, do this in the app store, or when they install the app.
- **When your app restarts, restore its state so users can continue where they left off.** People shouldn't have to remember the steps they took to reach their previous location in your app.
- **Never quit an app programmatically.** People tend to interpret this as a crash. If something prevents your app from functioning as intended, you need to tell users about the situation and explain what they can do about it.

Layout

- **Size** - **Make it easy for people to interact with content and controls**
- **Make it easy to focus on the main task** - place principal items in the upper half of the screen and—in left-to-right cultures—near the left side of the screen
- **Use visual weight and balance to show users the relative importance of onscreen elements.** Large items catch the eye and tend to appear more important than smaller ones. Larger items are also easier for users to tap, which makes them especially useful in apps—such as Phone and Clock—that users often use in distracting surroundings

Layout cont.

- **Use alignment to ease scanning and communicate groupings or hierarchy.** Alignment tends to make an app look neat and organized and it gives users places to focus while they scroll through screenfuls of information. Indentation and alignment indicate how the groups are related and make it easier for users to find specific items
- **Make sure that users can understand primary content in default/"normal" size.** For example, users shouldn't have to scroll horizontally to read important text, or zoom to see primary images.
- **As much as possible, avoid inconsistent appearances in your UI.** In general, elements that have similar functions should also look similar. People often assume that there must be a reason for the inconsistencies they notice, and they're apt to spend time trying to figure it out.

Navigation

Navigation - People tend to be unaware of the navigation experience in an app unless it doesn't meet their expectations. Your job is to implement navigation in a way that supports the structure and purpose of your app without calling attention to itself.

There are three main styles of navigation:

- Hierarchical
- Flat
- Content- or experience-driven

- In a **hierarchical** app, users navigate by making one choice per screen until they reach their destination. To navigate to another destination, users must retrace some of their steps—or start over from the beginning.
- In an app with a **flat** information structure, users can navigate directly from one primary category to another because all primary categories are accessible from the main screen.
- In an app that uses a **content- or experience-driven** information structure, navigation is also defined by the content or experience. For example, users navigate through a book by moving from one page to the next or by choosing a page in the table of contents; in a game, navigation is often an important part of the experience.
- In some cases, it works well to combine more than one navigation style in an app. For example, the items in one category of a flat information structure might best be displayed in a hierarchy

Navigation cont.

- **Users should always know where they are in your app and how to get to their next destination.**Regardless of the navigation style that suits the structure of your app, the most important thing is that a user's path through the content is logical, predictable, and easy to follow.
- **Use a navigation bar to give users an easy way to traverse a hierarchy of data.** The navigation bar's title can show users their current position in the hierarchy; the back button makes it easy to return to the previous level.
- **Use a tab bar to display several peer categories of content or functionality.** A tab bar is a good way to support a flat information architecture and its persistence lets people switch between categories regardless of their current location.
- **Display a "page control" when each app screen represents an individual instance of the same type of item or page.** A page control is good for showing users how many items or pages are available and which one is currently displayed.

- **In general, it's best to give users one path to each screen.**
  If there's one screen that users need to see in more than one context, consider using a temporary view (dialog, alert-box etc.)

Modal views

- **A modal view implies a task that must either be completed or cancelled before they can interact with rest of the app.**
- Ideally, people can interact with the app in nonlinear ways, so it's best when you can minimize the number of modal experiences in your app. In general, consider creating a modal context only when:

  - It's critical to get the user's attention
  - A self-contained task must be completed—or explicitly abandoned—to avoid leaving the user's data in an ambiguous state.

- **Keep modal tasks simple, short, and narrowly focused.** You don't want your users to experience a modal view as a mini app within your app. If a modal task must contain subtasks in separate views, be sure to give users a single, clear path through the hierarchy, and avoid circularities.
- **Always provide an obvious and safe way to exit a modal task.** People should always be able to predict the fate of their work when they dismiss a modal view.
- If the task requires a hierarchy of modal views, make sure your users understand what happens **if they tap a Done button in a view that's below the top level**. Examine the task to decide whether a Done button in a lower-level view should complete only the part of the task in that view, or the entire task. Because of this potential for confusion, **avoid adding a Done button to a subordinate view as much as possible**.
- **Reserve alerts for delivering essential—and ideally actionable—information.** An alert interrupts the user's experience and requires a tap to dismiss, so it's important for users to feel that the alert's message warrants the intrusion.
- **Respect users' preferences for receiving notifications.** In Settings, users indicate how they want to receive notifications from your app. Be sure to abide by these preferences so that users aren't tempted to turn off all notifications from your app.

Typography

- **Text Should Always Be Legible** - Above all, text must be legible. If users can’t read the words in your app, it doesn’t matter how beautiful the typography is.
- **When appropriate, adjust the layout when the user chooses a different text size.** For example, you might want to change a one-column layout of body text to a two-column layout when the user chooses a small text size.
- **In general, use a single font throughout your app.** Mixing several different fonts can make your app seem fragmented and sloppy. Instead, use one font and just a few styles and sizes.

Icons and graphics – UI icons

- UI icons are (small) icons used in buttons etc.
- It's a good idea to use the built-in icons as much as possible because users already know what they mean
- Note that you can use text instead of icons to represent items in a navigation bar or toolbar.
- To help you decide whether to use text or icons in the navigation bar or toolbar in your app, consider how many icons are visible onscreen at one time. Too many icons on a screen can make an app seem difficult to decode. Also, note that this decision might be different for a smartphone app than for a pad app because a pad tends to have more room for text in bars

Icon and graphics - photos

- **Display photos and graphics in their original aspect ratio, and don't scale them greater than 100%.** You don't want the artwork or graphics in your app to look skewed or too large. Let users choose whether they want to zoom images in or out.
- **Don't use images that replicate native products in your designs.** These symbols are copyrighted, and product designs can change frequently.

Using standard native UI elements

As much as possible, it’s a good idea to use the standard UI elements. When you use standard elements instead of creating custom ones, both you and your users benefit:

- For native apps: Standard UI elements automatically receive updates if the OS introduces a redesigned appearance—custom elements don’t get updated. Standard UI elements also tend to offer various ways to customize their appearance or behavior.
- People are comfortable with the standard UI elements, so they instantly understand how to use them in your app.
- To take advantage of the benefits of using standard UI elements, it’s crucial that you:
    - **Don’t mix UI element styles from different versions of the OS.** You don’t want to confuse users by displaying UI elements that look like they belong in a different version that’s currently running on the device.
    - **In general, avoid creating a custom UI element that performs a standard action.** First, ask yourself why you’re creating a custom UI element that behaves exactly like a standard one. If you just want a custom look, consider changing the look of a standard element. If you need completely custom behavior, it’s best to design a custom element that doesn’t look too similar to the standard ones
    - **Don’t use define standard elements to mean something else**. Be sure you understand the documented, semantic meaning of these buttons and icons; don’t rely on your interpretation of their appearance.
    - If you can’t find a system provided button or icon that has the appropriate meaning for a function in your app, you can create your own.

Interactivity - gestures

- **Avoid associating different actions with the standard gestures.** Unless your app is a game, redefining the meaning of a standard gesture may confuse people and make your app harder to use.
- **Avoid creating custom gestures that invoke the same actions as the standard gestures.** People are used to the behavior of the standard gestures, and they don't appreciate being expected to learn different ways to do the same thing.
- **Use complex gestures as shortcuts to expedite a task, not as the only way to perform it.** As much as possible, always give users a simple, straightforward way to perform an action, even if it means an extra tap or two.
- **In general, avoid defining new gestures unless your app is a game.** In games and other immersive apps, custom gestures can be a fun part of the experience. But in apps that help people do things that are important to them, it's best to use standard gestures because people don't have to make an effort to discover them or remember them.

Interactivity - feedback

Feedback helps users know what an app is doing, discover what they can do next, and understand the results of their actions.

- **As much as possible, integrate status and other relevant feedback information into your UI.** It's best when users can get this type of information without taking action or being distracted from their content.
- **Avoid unnecessary alerts**. An alert is a powerful feedback mechanism, but it should be used only to deliver important information. If users see too many alerts that don't contain essential information, they quickly learn to ignore all alerts.

Interactivity – inputting info

Inputting Information should be easy. When an app slows people down by asking for a lot of user input before anything useful happens, people can feel discouraged from using it.

- **Make it easy for users to make choices.** For example, you can use a picker or a table view instead of a text field, because most people find it easier to select an item from a list than to type words.
- **Get information from the OS, when appropriate.** People store lots of information on their devices. When it makes sense, don't force people to give you information that you can easily find for yourself, such as their contacts or calendar information
- **Balance a request for input by giving users something useful in return.** A sense of give and take helps people feel they're making progress as they move through your app.

Orientation

People generally expect to use their devices in any orientation, so it’s best when your app responds appropriately.

- **Maintain focus on the primary content in all orientations.** This is your highest priority. People use your app to view and interact with the content they care about. Changing the focus when the device rotates can disorient people and make them feel they’ve lost control over the app.
- **In general, run in all orientations.** People expect to use your app in different orientations, and it’s best when you can fulfill that expectation. If it’s essential that your app run in only one orientation, you should:
  - **Launch your app in the supported orientation, regardless of the current device orientation.** For example, if a game or media-viewing app runs in landscape only, it’s appropriate to launch the app in landscape, even if the device is currently in portrait. This way, if people start the app while the device is in portrait, they know to rotate the device to landscape to view the content.
  - **Avoid displaying a UI element that tells people to rotate the device.** Running in the supported orientation clearly tells people to rotate the device, if required, without adding unnecessary clutter to the UI.
  - **Support both variants of an orientation.** For example, if an app runs only in landscape, people should be able to use it whether they’re holding the device with the Home button on the right or on the left. And if people rotate the device 180 degrees while using the app, it’s best if the app responds by rotating its content 180 degrees.

Orientation cont.

- **If your app interprets changes in device orientation as user input, handle rotation in app-specific ways.** In a case like this, you should allow people to switch between the variants until they start the main task of the app. As soon as people begin the main task, begin responding to device movement in app-specific ways.
- **Anticipate users' needs when you respond to a change in device orientation.** Users often rotate their devices to landscape orientation because they want to "see more." If you respond by merely scaling up the content, you fail to meet users' expectations. Instead, respond by rewrapping lines of text and—if necessary—adjusting the layout of the UI so that more content fits on the screen.

Orientation cont.

- **Consider changing how you display auxiliary information or functionality.** While keeping the most important content always in focus, you can respond to rotation by changing the way you provide secondary content. E.g. a game could display supplemental information in the additional space.
- **Avoid unnecessary changes in layout.** For example, if your app displays images in a grid while in landscape, it's not necessary to display the same information in a list while in portrait (although you might adjust the dimensions of the grid).
- **As much as possible, avoid reformatting information and rewrapping text on rotation.** Strive to maintain a similar format in all orientations. If people are reading text in an app, it's especially important to help them keep their place when they rotate the device.
  If some reformatting is unavoidable, use animation to help people track the changes. For example, if you add or remove a column of text in different orientations, you can hide the movement of columns and simply fade in the new arrangement. To help you design appropriate rotation behavior, think about how you'd expect your content to behave if you were physically interacting with it in the real world.

Terminology and Wording

Every word you display in an app is part of a conversation you have with users. Use this conversation as an opportunity to provide clarity and to help people feel comfortable in your app.

- **Use terminology that you're sure your users understand.** Use what you know about your users to determine whether the words and phrases you plan to use are appropriate. For example, technical jargon is rarely helpful in an app aimed at unsophisticated users, but in an app designed for technically savvy users, it might be appreciated.
- **Use a tone that's informal and friendly, but not too familiar.** You want to avoid being stilted or too formal, but you don't want to risk sounding falsely jovial or patronizing. Remember that users are likely to read the text in your UI many times, and what might seem clever at first can become irritating when repeated.
- **Think like a newspaper editor, and watch out for redundant or unnecessary words.** When your UI text is short and direct, users can absorb it quickly and easily. Identify the most important information, express it concisely, and display it prominently so that people don't have to read too many words to find what they're looking for or to figure out what to do next.
- **Give controls short labels or use well-understood icons.** People should be able to tell at a glance what a control does.

Terminology and Wording cont.

- **Take care to be accurate when describing dates.** It's often appropriate to use friendly terms such as*today* and *tomorrow* when you display date information in your UI. But it can be confusing if you don't account for the user's current locale. For example, consider an event that starts just before midnight. To users in the same time zone, the event starts today, but to users in an earlier time zone, the same event may have started yesterday.
- **Correct all spelling, grammatical, and punctuation errors.** Although such errors don't bother everyone, in some people they can create a negative impression of your app's quality.
- **Keep all-capital words to a minimum.** Occasional all-capital words help draw people's attention, but when an entire passage is capitalized, it's difficult to read and it can be interpreted as shouting.
- **Make the most of the opportunity to communicate with potential users by writing a great app store description.** Also Consider describing specific bug fixes. If a new version of your app contains bug fixes that customers have been waiting for, it can be a good idea to mention this in your description.

Color

Color helps indicate interactivity, impart vitality, and provide visual continuity.

- **If you create multiple custom colors, make sure they work well together.** For example, if pastels are essential to your app's style, you should create a family of coordinating pastels that can be used throughout the app.
- **Pay attention to color contrasts in different contexts.** For example, if there's not enough contrast between the navigation bar background and the bar-button titles, the buttons will be hard for users to see. One way to discover areas that need higher contrast is to desaturate the UI and look at it in grayscale. If you have trouble telling the difference between interactive and noninteractive elements or backgrounds, you probably need to increase the contrast between these elements
- **Be aware of color blindness.** Most color blind people have difficulty distinguishing red from green. Test your app to make sure that there are no places where you use red and green as the only way to distinguish between two states or values

Color cont.

- **Consider choosing a key color to indicate interactivity and state.** If you define a key color to indicate interactivity and state, make sure that the other colors in your app don't compete with it.
- **Avoid using the same color in both interactive and noninteractive elements.** Color is one of the ways that a UI element indicates its interactivity. If interactive and noninteractive elements have the same color, it's harder for users to know where to tap.
- **Color communicates, but not always in the way you intend.** Everyone sees color differently, and many cultures differ in how they assign meanings to colors. Spend time to research how your use of color might be perceived in other countries and cultures. As much as possible, you want to be sure that the colors in your app send the appropriate message.
- **Don't let color distract users.** Unless color is essential to your app's purpose, it usually works well to use color as a subtle enhancement