Android
Human Interface Guidelines

Created by Kyle Stewart

Not affiliated with Google or Apple
This document is adapted from Apple's iPhone Human Interface Guidelines and various Google sources
Some content has been copied from these sources and I would like to thank Apple and Google for their very useful material

Contents

Foundation of Android Design

Framework Structure & Flow

What makes the Android User Experience Unique?

- Multitasking is supported by background processing
- Ongoing and event-driven notifications
- Real-time data views through widgets and live folders
- Any application can pick and choose which intents they want to take
    - Android is not about applications, its about activities that you layer tasks into which can be created by hooking together intents
- Any application can allow the spread of data to other applications
    - Use the hooks, re-use of components from intents is easy
- Android is about a stream of usage instead of a linear flow, at least that's what they say at Google

Framework Basics

Hardware Platform

The representation Android device is a happy marriage of hardware and software. Hard keys aid navigation and give greater functionality to Android. The menu button puts more content on the screen while maximizing screen usage when the menu is not up. The back button allows for the use of the back stack.

Portrait & Landscape

Always develop user interfaces with portrait and landscape in mind. Landscape still exists on the new Android phones. 99% of Android building block layouts will accommodate landscape.

Focus & Menus

No focus in touch mode, only with trackball. The idea of mouse focus does not translate to Android. Make sure you never show focus. The main menu should be comprised of global functions; they relate to an activity as a whole. Order the icons on the menu by importance. If there are more than five icons, use a more menu to hold your less important menu items. Contextual menus (long press) are focused on a specific item. Always put the most relevant action related to the selected item at the top of the long press menu.

Some things to keep in mind

- Design for speed and simplicity
    - Keep hierarchy as flat as possible
    - Minimize onscreen actions

    - Use lazy loading, load data rather than asking a user to wait to see a fully fleshed out page
- Think about activity streams as opposed to linear actions
    - Design your app to have hooks across the framework

Onscreen Behavior

Android has a specific behavior designed into it. Take advantage of this in your application. You should also stick to the standard Android behaviors so you don't confuse the users.

Expression

The details make the product, focus on details. The aesthetics of an app help call attention to the key tasks you have made the core of your application experience. API demos are a great place to start your toolkit.

Human Interface Principles

This section attempts to impart some basic interaction design principles useful in creating a great user interface. These principles are fundamental and can be applied to more than just Android user interface design. Apple suggests that developers spend 60% of their development time doing design work. The following human interface principles will provide a foundation for great design.

Metaphors

Metaphors are the building blocks in the user's mental model of a task; use them to convey concepts and features in your app. Basing your application's objects on real world objects helps users quickly understand your application. As you design your application, be aware of the metaphors that exist in Android and don't redefine them. At the same time, examine the task your app performs and see if there are natural metaphors you can use.

Reflect the User's Mental Model

The user already has a mental model that describes the task your application is enabling. This model arises from a combination of real-world experiences, experience with other software, and with computers in general. For examples, users have a real-world experience writing and mailing letters and have certain expectations, such as the ability to create a new letter, select a recipient, and send a letter. An email application that ignores the user's mental model would be difficult and unpleasant to use. This is because the application imposes an unfamiliar conceptual model on its users instead of building on the knowledge and experience the users already have.

Before you design your application's user interface, try to discover your users' mental model of the task your application helps them to perform. Be aware of the model's inherent metaphors, which represent conceptual components of the task. In the letter-writing example, the metaphors include letters, mail boxes, and envelopes. In the mental model of a task related to photography, the metaphors include photographs, cameras, and albums. Strive to reflect the user's expectations of task components, organization, and workflow in your window layout, menu and toolbar organization, and use of panels.

You should support the user's mental model by striving to incorporate the following characteristics:

Familiarity
The user's mental model is based primarily on experience.

Simplicity
A mental model of a task is typically streamlined and focused on the fundamental components of the task. Although there may be many optional details associated with a given task, the basic components take priority and should not have to compete for the user's attention.

Availability
Avoid hiding components too deeply in submenus or making them accessible only from a contextual menu.

Discoverability
Encourage your users to discover functionality by providing cues about how to use user interface elements. Don't discourage discovery by making actions difficult to reverse or recover from.

Direct Manipulation

Direct manipulation means that people feel they are controlling something tangible, not abstract. The benefit of direct manipulation is that users more readily understand the results of their actions when they can directly manipulate the objects involved. iPhone gives users a heightened sense of direct manipulation using multi-touch. Android can give users most of that experience by using single-touch appropriately. To enhance the sense of direct manipulation in you application, make sure that:

- Objects on the screen remain visible while the user performs actions on them
- The result of a user's action is immediately apparent

Animation

Animation is very important in supporting direct manipulation. It helps the user feel more attached to the device because the interface reacts like real world objects would. Give the user an experience "absent of abruptness" by using the dog ears principle. What happens when a dog stops running? Its ears keep going and bounce back. Give your user interface this alive feel. For example, iPhone fades playing music when switching to another app or taking a call. Another example is the difference between table views on iPhone and list views on Android. When a user gathers scrolling momentum and reaches the end of a list, the scrolling abruptly stops on Android while it continues and bounces back (think dog ears) on the iPhone. The iPhone has real world follow through and Android just hits a wall and stops, yuck! It seems like a small thing but makes a big difference in connecting with users. Seriously, use an iPhone for a while, you will love the animation.

See and Point

An Android application is better than a person at remembering lists of options, commands, data, and so on. Take advantage of this by presenting choices or options in list form, so users can easily scan them and make a choice. Keep text input to a minimum.

User Control

Allow users, not your app, to initiate and control actions. Keep actions simple and straightforward so users can easily understand and remember them. Whenever possible, use standard controls and behaviors that users are already familiar with. They key is to provide users with the capabilities they need while helping them avoid dangerous, irreversible actions. For example, in situations where the users might destroy data accidentally, you should always provide a warning, but allow the user to proceed if they choose.

Feedback and Communication

Users need immediate feedback when they operate controls and status reports during lengthy operations. Your application should respond to every user action with some visible change. For example, list items should highlight briefly when tapped so the user knows there touch event registered. Animation is a great way to provide feedback to users.

Forgiveness

Encourage users to explore your application by building in forgiveness – that is, making actions easily reversible. Warn users when they initiate a task that will cause irreversible loss of data. Anticipate common problems and alert users to potential side effects.

Aesthetic Integrity

Aesthetic integrity means that information is well organized and consistent with principles of good visual design. It is also how well the appearance of your application integrates with its function. Appearance has a strong impact on functionality. An application that appears cluttered or illogical is hard to understand and use. The overall layout of your windows and design of user interface elements should reflect the user's mental model of the task your application performs.

Managing Complexity in Your Application

The best approach to developing easy-to-use software is to keep the design as simple as possible. The more complex your application's task, the more important it is to keep the user interface simple and focused.

Interaction Design on Android

The Google Android OS is a unique platform with many interesting features. Developers should seek to use these features correctly if they want their application to be part of the Android experience. This section will help developers better understand interaction design on the Android platform.

Activities & Tasks

Applications

An Android application typically consists of one or more related, loosely bounded activities for the user to interact with.

Activities

Activities are the main building blocks of Android applications. You can assemble an application from activities you create and from other activities available on the Android OS. Each activity you build should be designed with a single purpose such as taking a photo, finding a contact, or reading an email.

Applications that display a user interface consist of one or more activities.

When using an Android device, as the user moves through the user interface they start activities one after the other, totally oblivious to the underlying behavior – to them the experience should be seamless, activity after activity, task after task. This is like the stream of activities talked about earlier.

Keep in mind that the activities should work together to form a cohesive user interface. If your activity does not follow basic interaction design principles and it ties into a system activity, the user may become confused and frustrated from the lack of consistency.

An activity handles a particular type of content (data) and accepts a set of related user actions.

(Show activity example, mail, contacts, etc)

Activity Stack

As the user moves from activity to activity, across applications, the Android system keeps a linear navigation history of activities the user has visited. This is the *activity stack*, also know as the back stack.

In general, when a user starts a new activity, it is added to the activity stack, so that pressing BACK displays the previous activity on the stack.

However, the user cannot use the BACK key to go back further than the last visit to Home.

Activities are the only things that can be added to the activity stack – views, windows, menus, and dialogs cannot.

Tasks

A *task* is the sequence of activities the user follows to accomplish an objective, regardless of which applications the activities belong to.

Until a new task is explicitly specified, all activities the user starts are considered to be part of the current task.

The activity that starts a task is called the root activity. If is often, but not necessarily, started from the application launcher, Home screen shortcut or "Recent tasks" switcher.

The user can return to a task by choosing the icon for its root activity the same way they started the task.

Interrupting the Task
An important property of a task is that the user can interrupt what they're doing (their task) to perform a different task, then are able to return to where they left off to complete the original task. The idea is that users can run multiple tasks simultaneously and switch between them. (show example)

Menu Design

See Menu Design Guidelines
http://developer.android.com/guide/practices/ui_guidelines/menu_design.html

Icon Design

See Icon Design Guidelines
http://developer.android.com/guide/practices/ui_guidelines/icon_design.html

App Widget Design

See Widget Design Guidelines
http://developer.android.com/guide/practices/ui_guidelines/widget_design.html

Design for Ease of Use

In development…

Design for Consistency

In development…

Touch Mode

In development…

Themes???

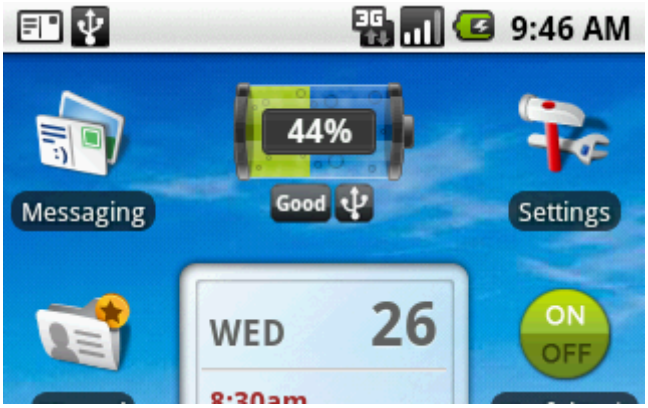
Polish your App

In development…

Designing the User Interface

As much as possible, you should use the standard user interface elements provided in Android and follow their recommended usages. Users are accustomed to the look and behavior of standard views and controls. If you use standard interface elements, users can depend on their prior experience to help them as they learn to use your application.

Status Bar



Usage and Behavior

The status bar contains important information such as the battery charge, current time, network, and signal strength. It also may display notification icons to the user. Although your app can hide the status bar, you should carefully consider the ramifications of this design decision. Users expect to be able to see the information found on the status bar. On the iPhone your app can tell the status bar to become transparent and allow the user to see your app's window behind it. You can even customize the color of the status bar, take note Android!

Recommendations

- Don't hide the status bar unless you have a good reason to do so (e.g. your app is a game).
- Take advantage of the notifications system that allows you to put icons in the status bar.

Examples

In development…

Notifications

Usage and Behavior

In development…

Recommendations

In development…

Examples

In development…

Dialogs

In development…

Toast

In development…

Progress Bar

In development…

Options Menu

Usage and Behavior

In development…

Recommendations

In development…

Examples

In development…

Context Menu

Usage and Behavior

In development…

Recommendations

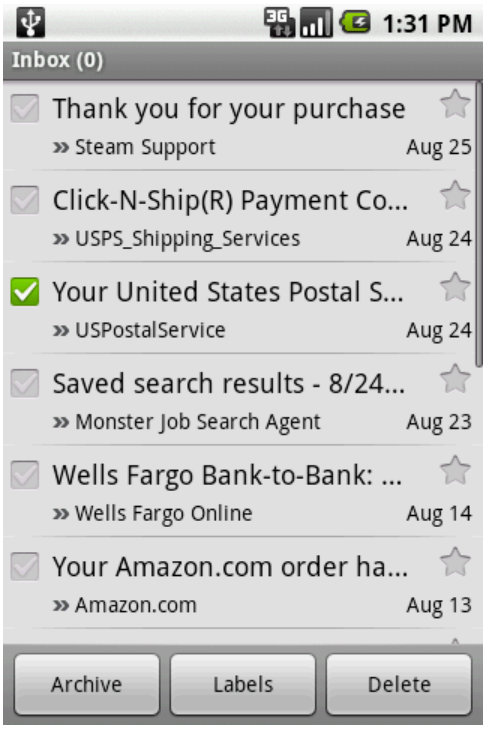
In development…

Examples

In development…

Toolbar

In development…


1:31 PM
Inbox (0)
Thank you for your purchase
» Steam Support
Aug 25
Click-N-Ship(R) Payment Co...
» USPS_Shipping_Services
Aug 24
Your United States Postal S...
» USPostalService
Aug 24
Saved search results - 8/24...
» Monster Job Search Agent
Aug 23
Wells Fargo Bank-to-Bank: ...
» Wells Fargo Online
Aug 14
Your Amazon.com order ha...
» Amazon.com
Aug 13
Archive
Labels
Delete

Tab Widget


2:31 PM
TAB 1
TAB 2
TAB 3
this is a tab


2:32 PM
TAB 2
TAB 3
this is another tab

Usage and Behavior

A Tab Widget offers the ability to easily draw an interface that uses tabs to navigate between different views. The tabs are available anywhere in your application. They can be used as an alternative to the options menu for navigation through your app. Each tab should provide different perspectives on the same set of data, or different subtasks related to the overall function of the application. iPhone's alternative to the Tab Widget is the Tab Bar which looks nicer and is located on the bottom of the screen where Android should have put theirs.

Recommendations

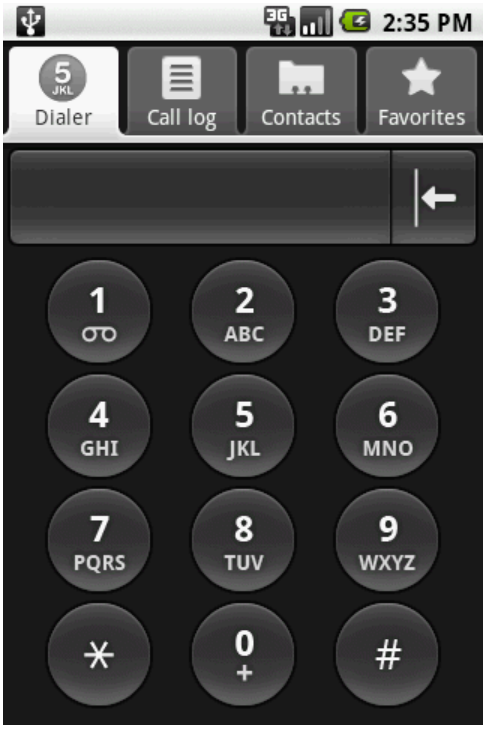
- The Tab Bar widget isn't the same as an iPhone Toolbar. It is not meant to be persistent when combined with list views.
- Use easy to understand grayscale images in your tabs. Refer to the Android Contacts app for examples.
- Use text along with images to make the tabs functionality more understandable. Make sure the text is small so it will fit in the tab.
- Four is the optimal number of tabs, more than four makes them hard to read and tap. If the app is made to be used only in landscape mode, more tabs can be added.
- If you use tabs, they should *almost always* be available to the user. Removing them in some areas of your application will make the interface inconsistent and the user may get lost.

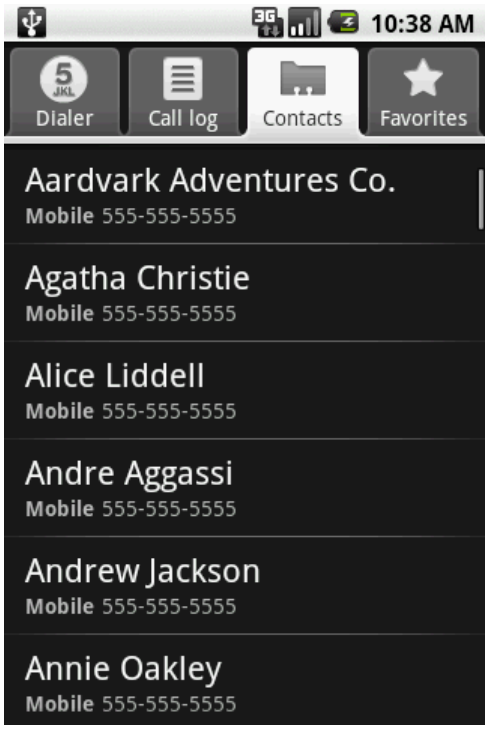
Examples

Code: HelloTabWidget

http://binarysheep.com/AndroidCode/HelloTabWidget/

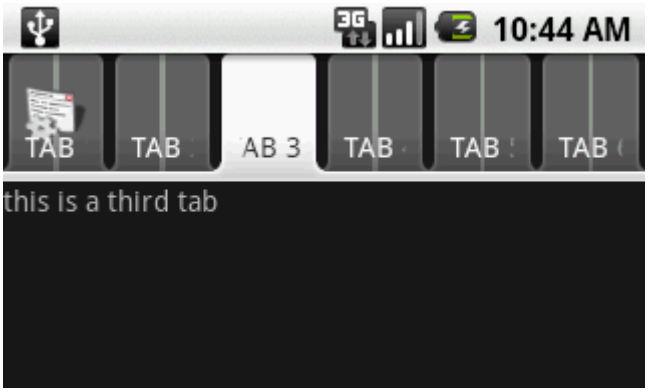
Application: Contacts





The Contacts app makes very good use of the Tab Widget. Each tab has a simple grayscale image and text so the user quickly understands their functionality. Notice that the tabs align well with the user's mental model by using the metaphors of a dialer, call log, contacts, and favorites.

Sample: Too Many Tabs

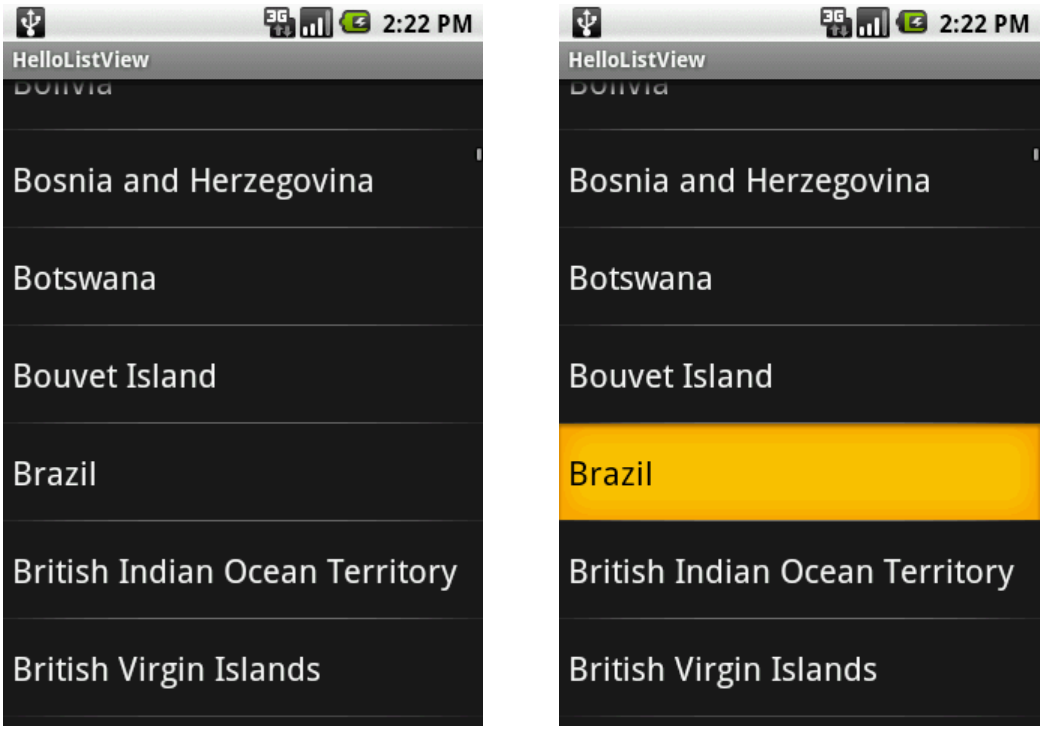


There are too many tabs unless if this app is meant to be used in landscape mode only. Users may have a hard time tapping the tabs or even understanding which tab does what.

Sample: Tabs with List Views

???

List View



Usage and Behavior

Mainstay of Android programming. Can be used for drill down interfaces. Hook up to ???

A list view presents data in a single-column list of multiple rows. Each row can contain some combination of text, images, and controls. Rows can be divided into groups (is this true)? List views are very useful because they can organize large amounts of data. Feedback is provided when a user selects an item. The row containing the item highlights briefly to show the selection has been made, and then the defined action attached to that row occurs. You might also notice that the top of the list view shown above uses *fading edges*. Android employs this technique to indicate that the container can be scrolled. List views are versatile and can be used for many different user interfaces.

- **Selecting Options**
  - The list view can display a list of options from which the user can choose. A checkmark image can show the currently selected option in the list.
- **Navigating Hierarchical Information**
  - A list view can also be used to display a hierarchy of information in which each row (that is, list item) can contain its own subset of information. Apple refers to this as a drill down interface and uses it often on the iPhone. Sadly Android does not have navigation bars

like iPhone that can be attached to the top of the list view to make hierarchical drill downs easier to navigate.

- **Looking up Indexed Information**
  - A list view can be used to display information ordered according to a scheme (e.g. alphabetized).
- **Viewing Conceptually Grouped Information**
  - Can we group information (e.g. work, home, school)

Recommendations

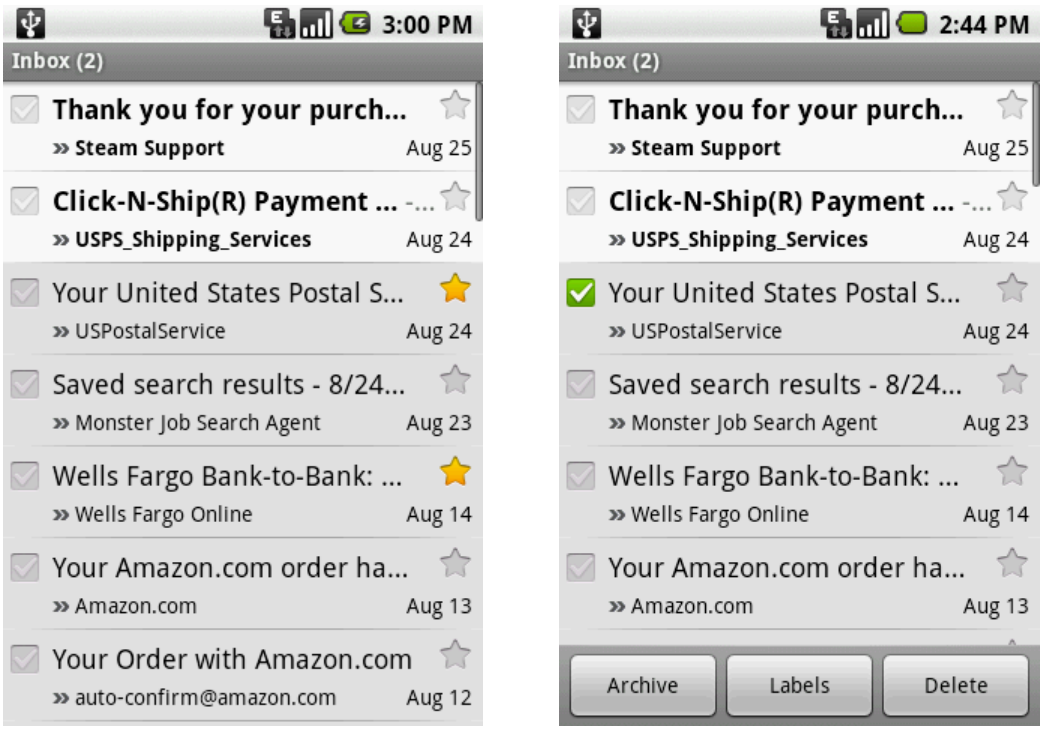
- Use a checkmark image to show an item as selected instead of leaving the row highlighted.
- Keep in mind that the list view widget shows more rows in portrait mode.
- There is no current item selection in [touch mode](#).
- If each row needs to display multiple information items, consider using a three-column layout (see examples below)

Examples

Code: HelloListView

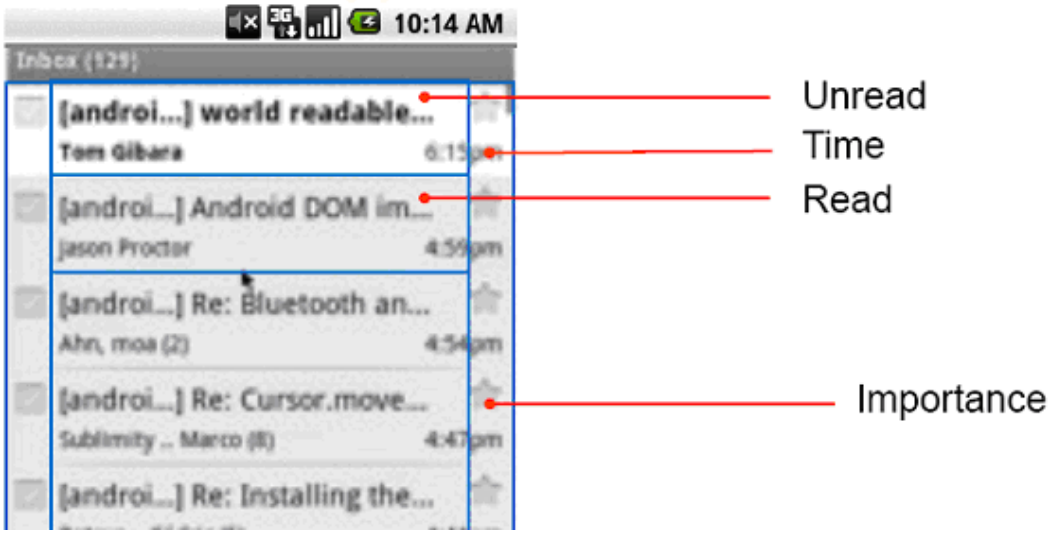
http://binarysheep.com/AndroidCode/HelloListView/

Application: Email

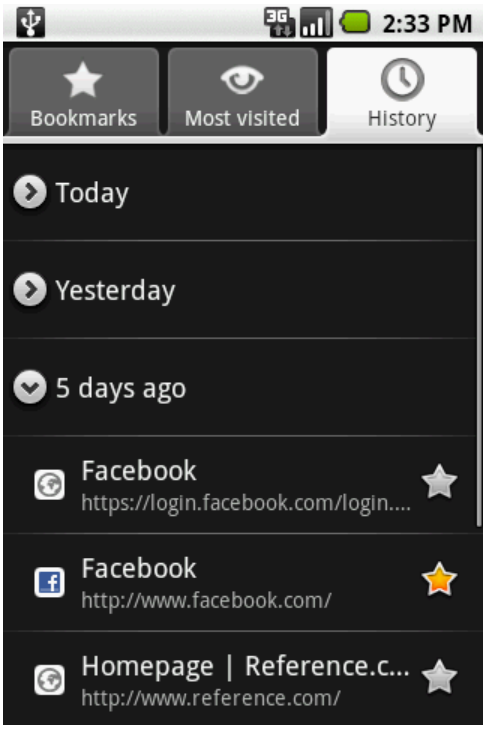


Google does a good job making the most out of the list view widget in the Gmail app. A three-column layout is used to stuff usability into the app without cluttering the design. The first column triggers a multitasking mode which pops up the three-button toolbar shown above. This toolbar is only shown when one or

more items in the list view is checked. It allows users to quickly perform an action on multiple items in their email box. The user may also find that pushing the menu button gives them even more options related to the selected emails. These features are very nice for those who discover it by clicking a checkbox. The second column shows the title and sender of the email. You will notice that read and unread messages are colored differently. Color is a great way to show more information without using any screen real-estate. The third column allows the user to give a higher significance an email item. It also displays the time the email message was received.



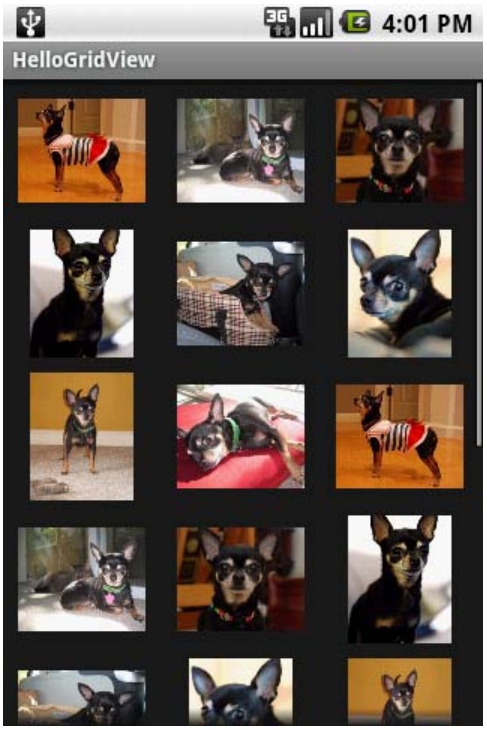
Application: Browser History



The history tab in the browser will most likely have a large amount of history. Google used an expandable list behavior to compress the data and help the user navigate it more easily. You may also notice that they used the same three-column layout found in the Gmail application.

Expandable List

Grid View



Usage and Behavior

A Grid View displays items in a two-dimensional, scrolling grid. The items are acquired from a ListAdapter.

Recommendations

In development…

Examples

Code: HelloGridView
http://binarysheep.com/AndroidCode/HelloGridView/

Application: Gallery
In development…

Gallery

In development…

Text View

In development…

Image View

In development…

Spinner

Usage & Behavior

The Spinner is the optimal widget to use to provide a dropdown list behavior. When the user wants to set the value in the Spinner, they tap anywhere on the spinner widget (not just the down arrow on the right). The value list is displayed in list view style. Using a spinner temporarily takes the user away from the rest of

the UI. To help the user realize they haven't left the UI, the Spinner value list animates onto the screen and floats on top of the UI. The user can then pick their desired value by tapping anywhere on the value's row. The Spinner list then animates back into the UI and the selected value is shown on the Spinner.

Recommendations

- If its not obvious from the list values, add a title to remind the user what they are doing
- Less than 16 values is optimal so the user only has to thumb scroll three or less times

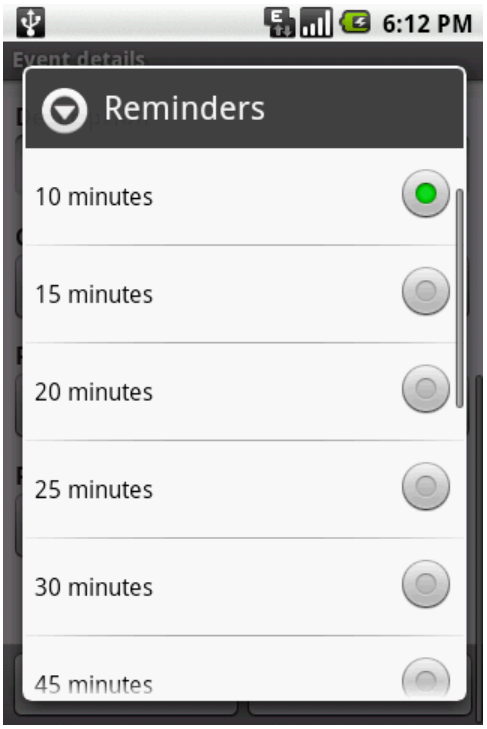
Examples

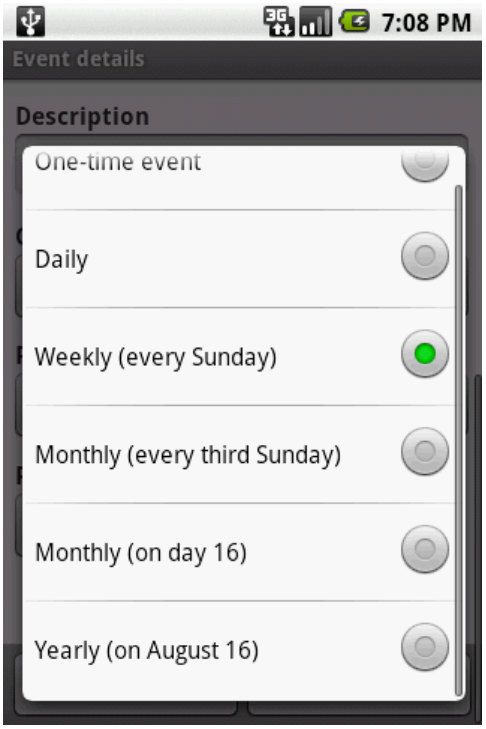
Code: HelloSpinner

http://binarysheep.com/AndroidCode/HelloSpinner/

Application: Calendar

When adding a new event to the calendar the user clicks on the Reminders Spinner and is given a list of possible values. For some reason the next Spinner in the Calendar UI does not have a title. In Android the value list of a Spinner does not expand to fill the title space when no title is given. It's a good idea to be consistent with titles; either all of your Spinners have them or none do.





Auto Complete

Usage and Behavior

In development…

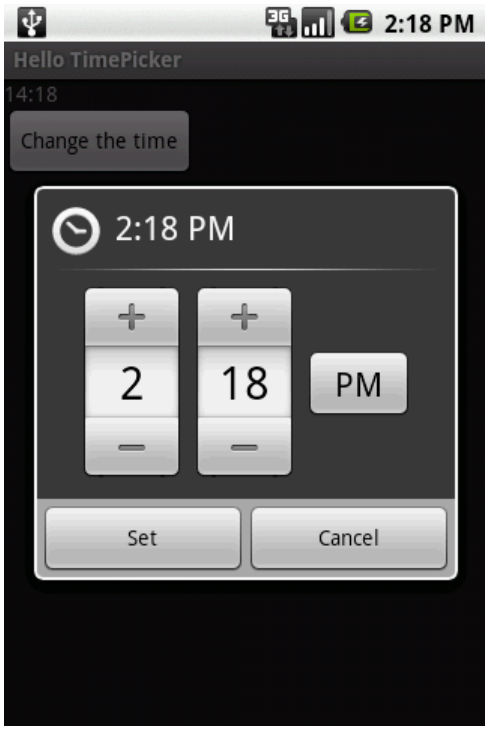
Recommendations

In development…

Examples

In development…

Date & Time Picker



Usage and Behavior

In development…

Recommendations

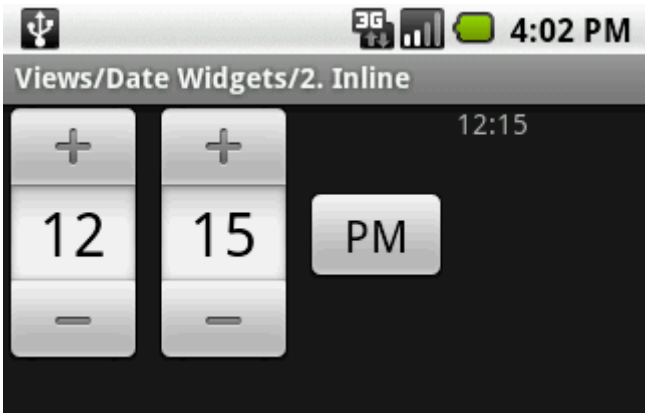
In development…

Examples

Code: ???
In development…

Application: ???

Example: ???



The time picker widget can also be used inline on the view.

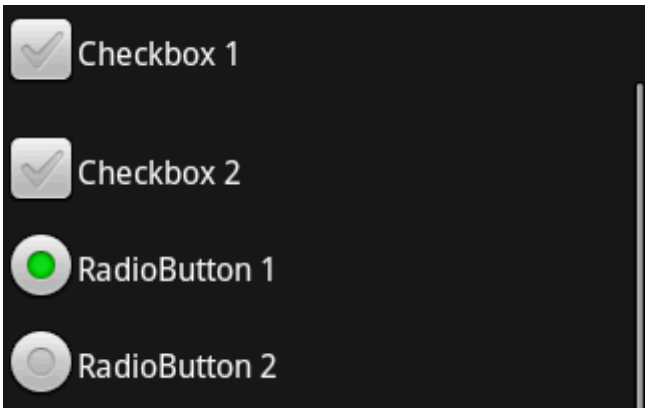
Button & Image Button

In development…

Edit Text

In development…

Check Box & Radio Button & Toggle Button



Usage and Behavior

A check box widget presents the user two mutually exclusive choices or states, such as yes/no or on/off.
A switch control shows only one of the two possible choices at a time; users slide the control to reveal the hidden choice or state.

Recommendations

In development…

Examples

In development…

Map View

In development…

Web View

In development…

Preferences

In development…

A Development Process

STEP 0 – Read the Human Interface Guide

STEP 1 – Decide What to Build

- Create an **Application Definition Statement**
  - *Define a solution, not a collection of features*
  - Summarize the intended purpose of your application
  - Identify your primary user type
  - Use it to guide development and filter every feature
  - Example: iPhoto's Application Definition Statement
    - Desktop: Easy to use digital photo editing, organizing, and sharing for casual and amateur photographers
    - iPhone: Easy to use digital photo sharing for iPhone users
- **Know your users**
  - Who is the user you are building this software for?
    - Pick a very narrow user group (or a single user)
    - Create a *persona* with details of the target user, keep this persona in view as you work (e.g. Who is this person, what is their typical day like, how do they perform tasks)
  - Develop your app with people and their capabilities – not computers – in mind
  - Communication with your user base is key
    - *Involve users in every phase of the design process*
    - Conduct user observation sessions
  - The best way to have a successful app is to build an app for yourself
- **Refine your basic feature set**
  - Develop a basic description of each feature
  - Select the bare minimum of features
    - Filter with your Application Definition Statement
    - Select the fewest features that are most frequently used by the majority of users and are appropriate for a mobile context

STEP 2 – Visit the App Store / Android Market

- Analyze similar products in related markets to see what audiences they target

    - Are these products competitors or do they interleave nicely with your app
    - Do competing apps have high or low switching costs?
- Build on already established usage patterns
    - Mobile users have very limited attention and learning spans
    - Create apps that are quick to learn and build on existing learned usage
- Study well designed apps for inspiration

STEP 3 – Explore Possible Solutions

- Define and analyze the user's mental model
    - Discover the mental or conceptual model people associate with the task your app will help them perform
    - How do people perform similar tasks without a computer?
    - What concepts, objects, and gestures do your users associate with this task?
- **Apply Human Interface Design Principles**
- Develop an expanded features list
    - Give each feature a concise description
- ***Be minimal – Great design is about solutions, not features***
    - Design for 80% of your users and allow customization for the rest
    - Use your application definition statement and user persona to filter your features
    - Do user testing to get feedback on which features are missing or extraneous
    - The best apps are the ones whose features are tightly integrated with the solutions they provide
- Don't get stuck on your first design
    - Your first design is never the best, especially when you are new on the platform

STEP 4 – Sketch

- Sketch 10 alternative designs for your app
    - The last couple of designs are hard to come up with but the most creative ideas will usually come from these
- Take these sketches to your user group or friends for feedback
    - This will help narrow down your 10 designs to a few designs
- ***Quality through quantity***
    - By creating many designs you will get to the better design much faster

STEP 5 – Prototype in Omnigraffle

- You end up with a **pixel perfect** interface layout
  - The developers will know exactly what to create
- Paper prototype – One piece of paper for each of your screens
  - Users can now use your application on paper
  - Do **user observation tests** with your prototypes
- Principle – Fail early to succeed sooner
- (Note: Omnigraffle is Mac only software, alternatives are Fireworks, Photoshop, Visio, and similar)

STEP 6 – Do It All Again

- Its ok to throw it away and start again
- Reiterate the design so you don't have to rewrite code
- Spend at least 60% of your total development time on design, apple does!
- Principle – Remember that nothing is precious

STEP 7 – Okay, You Can Code Finally

- Use the MVC design paradigm
- Use Top Down design
  - Design the interface first and the back end data later
  - Use dummy data stubs when implementing the interface
- Code for performance
  - Use code profiling tools to gather data about where your application is performing poorly
  - Avoid waiting until the end of your development cycle to do performance tuning
    - Include specific goals in your product requirements

STEP 8 – Beta Test Your Application

- Test before you submit
  - Your application is only on the new apps page once
  - If your users rate you low because of small bugs, it is hard to recover
- Who should be your beta testers?
  - Friends
  - Amazon Mechanical Turk (https://www.mturk.com)
- Principle – Test before you submit

STEP 9 – Release!

- Be ready to do support and bug fixes
- Polish for application differentiation
- Iterate based on user feedback (should already be doing this)

Ingredients for Great Apps

All developers need experience using the Android OS platform.

Delightful

- Inviting – makes you want to use it
  - Add a fun component to your app
- Intuitive – Does what you expect it to do
  - You shouldn't have to think about how to use it
- Engaging – Keeps you occupied and involved
- Exciting – Provides a thrill
- Enabling – Lets you do things you never could before
  - Or extends your ability to do things you already do
  - Example: Facebook on your phone allows you to use it anytime

Innovative

- Revolutionary
- Inspirational
- Fresh

Designed

- Great Design…
  - Is supported from the top
  - Comes from small teams
    - Inter-disciplinary
    - More organic
    - Easier to manage
    - Better communication
    - Make decisions more quickly
    - Iterate quickly
  - Comes from a constant focus on design
    - Spend most of the development time in design phase
    - Continue designing through the entire development cycle
  - Is about solutions, not features
  - Comes from saying "No" to 1,000 things

- Choose the top features and amplify them

Integrated

- Take advantage of existing Android frameworks
- Use frameworks as intended
  - Avoid trying to get around frameworks

Optimized

- Do performance analysis, testing, and benchmarking
- Focus on performance throughout the entire development cycle
- Build quality – make sure each revision of your app is better

Connected

- Work well with other Android applications
  - Mail, browser, phone, maps, etc
- Use on-board personal information
  - Photos, Contacts, etc
- Go beyond the device
  - Connect to remote content and services
  - Work well offline too!

Localized

- U.S. isn't the only market for Android
- Gain a major competitive advantage by localization
- Don't use online translators, find a translation service or work with users and community