PRACTICAL STEPS TO ANDROID APP DEVELOPMENT FOR GRAPHIC DESIGNERS

Vesa Antikainen

Bachelor's thesis
March 2014
Degree Programme in Media
Interactive Media

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media
Interactive Media

VESA ANTIKAINEN:
Practical Steps to Android App Development for Graphic Designers

Bachelor's thesis 47 pages, appendices 1 page
March 2014

The purpose of this thesis is to give graphic designers new to Android a clear view of what goes into developing an app.

First all the basics of the Android OS are introduced with building blocks and concepts needed to get started. After introducing the fundamental elements, the thesis provides some recommendations on what programs could be used and how the overall workflow should go. In the final part previously mentioned building blocks and workflow patterns are put to use in the example case Dischive.

In conclusion Android OS was fairly easy platform for graphic designers, helped by the outstanding design documents provided by Google. The industry standard software has not been fully optimized for developing for Android OS, but things are improving, making developing for Android fairly easy.

Key words: android, graphic design, mobile

CONTENTS

1 INTRODUCTION 5
2 ANDROID AS A PLATFORM. 6
2.1 Different versions 6
2.2 KitKat. 7
2.3 Older versions vs. KitKat 7
2.4 System and Custom Themes 7
3 METRICS AND GRIDS 9
3.1 Size and Density Buckets 9
3.2 48dp rhythm 11
4 ANDROID STYLE GUIDE. 12
4.1 Design Principles. 12
4.2 Colors. 13
4.3 Icons. 13
4.4 Typography 15
4.5 Android vs. Other Operating Systems 15
5 USER INTERFACE ELEMENTS 17
5.1 Element states 17
5.2 Icons. 18
5.3 Building Blocks 18
5.3.1 Buttons. 18
5.3.2 Switches and Sliders 20
5.3.3 Progress & Activity Indicators. 21
6 WORKFLOW 23
6.1 Prototyping Software. 23
6.2 Vectors and Artboards. 23
6.3 Naming conventions and asset organizing 24
6.4 File Sizes and Formats. 26
6.4.1 WebP. 26
6.4.2 PNG Optimizers 27
6.5 Automation 28
6.5.1 Actions 29
6.5.2 Batch Processing 30

6.6 Android Asset Studio 31
6.6.1 9-patch 31
6.6.2 Action Bar Style & Holo Colors Generators 34
7 EXAMPLE CASE − DISCHIVE 36
7.1 User Interface 36
7.2 Design Choices 37
7.2.1 Branding 37
7.2.2 Web version vs. Android version 38
7.2.3 Android Components 39
7.3 Problems 40
7.3.1 Marking Shots 40
7.3.2 Selecting Discs 40
7.3.3 Score vs. Throws 41
7.3.4 Hole Information 42
7.4 Future of Dischive App 42
8 DISCUSSION 45
REFERENCES 46
APPENDICES 50

1 INTRODUCTION

Android is the most popular mobile platform in the world. Every day a million new Android devices are registered. It is also the fastest growing mobile OS (operating system). (Android Developers, 2013a.) It was originally developed by Android Inc. but was bought by Google in 2005 (Elgin, 2005).

This thesis aims to give graphic designers a good overview on the fundamentals of Android app development and what kind of elements there are in the OS. Also this thesis provides some advice on what tools and software can and should be used in the designers' workflow and how one might go about doing so.

The example case, Dischive, is an upcoming Android app that gives disc golfers a way to document their scores and have statistics compiled from the data. This section highlights how the previously mentioned workflows and design principles are applied in to real life project. Also some problems that have arisen are inspected with an explanation of how they were solved.

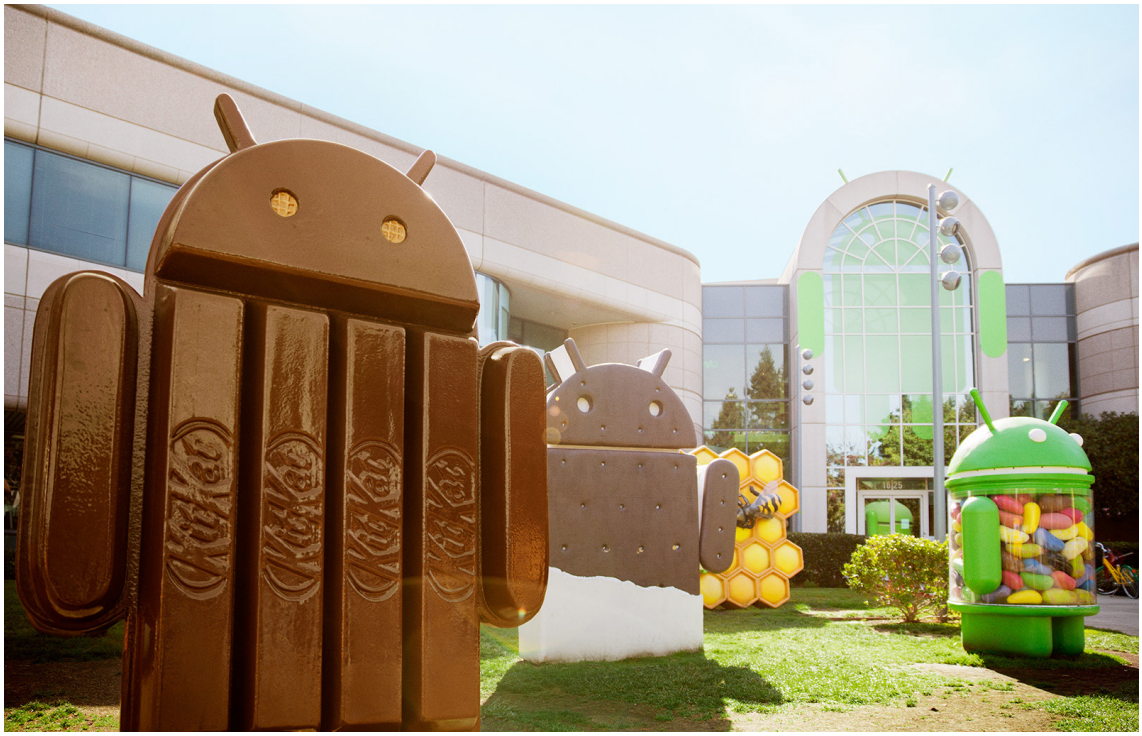
2 ANDROID AS A PLATFORM

2.1 Different versions

There are multiple versions of Android OS on the market. They use a codename for each iteration to help users differentiate significant changes in the software. Google has come to use snacks as codenames in alphabetical order, starting from Cupcake (Android 1.5) and following with the likes of Froyo (2.2–2.2.3), Ice Cream Sandwich (4.0– 4.0.4) and Jelly Bean (4.1–4.3).

All Android OS versions (PCMag.com, 2014):

- Version 1.0/1.1
- Cupcake 1.5
- Donut 1.6
- Eclair 2.0/2.1
- Froyo 2.2
- Gingerbread 2.3
- Honeycomb 3.0
- Ice Cream Sandwich 4.0
- Jelly Bean 4.1/4.2/4.3
- KitKat 4.4



PICTURE 1. Android mascots depicting different OS versions (Android.com, 2013)

2.2 KitKat

The latest one of the versions is 4.4, KitKat. The Android Developers website promises it to be more streamlined and designed to run better on entry level devices (Android Developers 2013b), meaning the OS isn't as demanding on hardware as the previous version.

From a graphic designers' and user experience designers' point of view, KitKat only brings few new things, compared to previous versions of Android. These are mainly the new design guidelines and full screen mode where all system UI (user interface) is hidden and can be revealed with a swipe gesture.

2.3 Older versions vs. KitKat

What usually comes with an older OS version is either a smaller or a lower DPI (dots per inch) screen on the device. Android's way of combatting different screen sizes and DPI's is to have different graphic assets for different devices. This creates an extra step or two in the designer's workflow but is well worth the work as your app will look great on most if not all Android devices. Even though they might have a little bit different look and feel you should be fine, when you follow the Android design guidelines[1] in your design process.

2.4 System and Custom Themes

"Themes are Android's mechanism for applying a consistent style to an app or activity. The style specifies the visual properties of the elements that make up your user interface, such as color, height, padding and font size." (Android Developers, 2013c.)

1 http://developer.android.com/design/index.html

There are two system themes in Android: Holo Dark and Holo Light. Holo Dark is basically the same theme as Holo Light but with dark colors. Paddings, heights and font sizes are the same.

FIGURE 1. Settings in Holo Dark (Android Developers, 2013c)

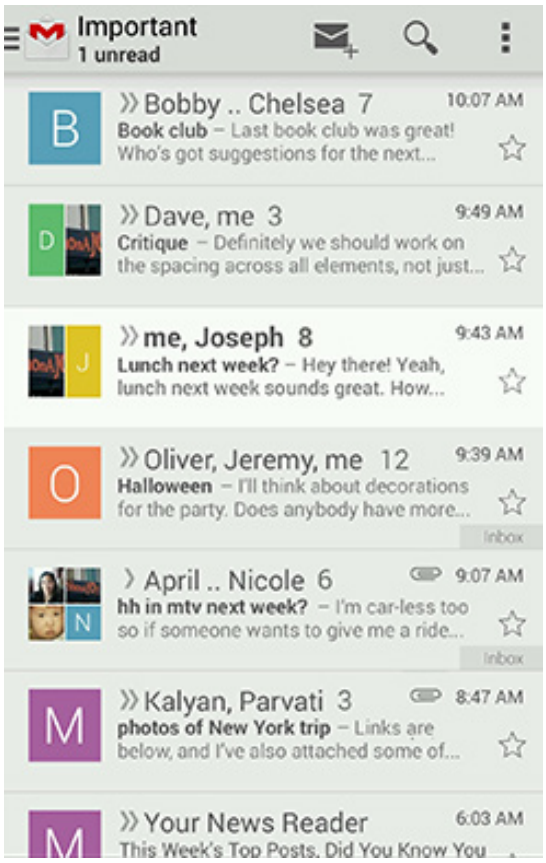


FIGURE 2. Gmail in Holo Light (Android Developers, 2013c)

You can create custom themes to better suit your app's (application) branding and color scheme. This can be done via the Android SDK (Software Development Kit). Most graphic designers don't have good enough coding skills to make custom themes, so usually a developer is needed in the process. When creating a theme it is adviced to use one of the two system themes as a starting point (Android Developers, 2013c). You should also follow the Android design guidelines when ever possible.

3 METRICS AND GRIDS

Every day a million new Android devices are activated (Android Developers, 2013a). Those come in different sizes, aspect ratios and screen densities. Android OS does its best to handle all possible variations. The OS scales graphical assets depending on the situation. The developer is expected to include alternative assets, different size versions of graphics, in their app for the OS to deliver optimized experience for the user.

3.1 Size and Density Buckets

Android documentation categorizes physical screen sizes in four rough categories: small, normal, large and xlarge. Size of the screen is measured as a diagonal of the screen, similarly as in TVs and monitors.

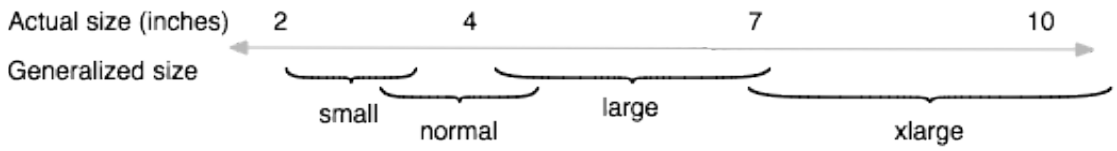


FIGURE 3. Illustration of how Android roughly maps actual screen sizes (Android Developers, 2013g, modified)

It also categorizes devices in two categories describing the nature of the device:

- Handset – smaller than 600dp
- Tablet – larger than or equal 600dp

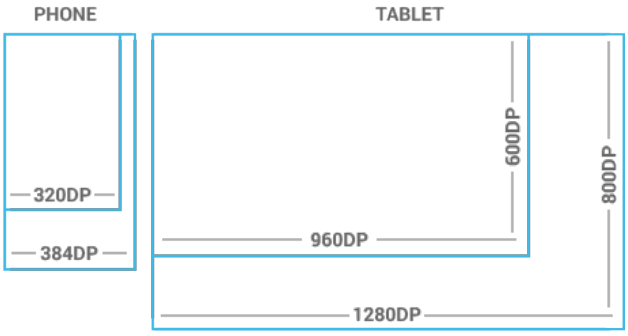


FIGURE 4. Device categories (Android Developers, 2013j)

Screen densities are measured in dots per inch (dpi). The number represents how many pixels there are within the physical screen. In Android world screens are most commonly measured in density-independent pixels (dp). One dp is equal to one physical pixel on a 160dpi screen. Different densities are grouped into generalized density buckets for the sake of simplicity (Android Developers, 2013g.):

- LDPI – Low density (120dpi)
- MDPI – Medium density (160dpi)
- HDPI – High density (240dpi)
- XHDPI – Extra-high density (320dpi)
- XXHDPI – Extra-extra!-high density (480dpi)
- XXXHDPI – Extra-extra-extra!-high density (640dpi)

The last one (XXXHDPI) is practically nonexistent but should be kept in mind if one wants to futureproof their app. The majority of devices are made up of high and extra-high densities while medium still holds at around 20% of the user base, according to data collected from devices running the latest Google Play Store app. (Android Developers, 2013h.)

There is also TVDPI (~213dpi) which fits in the middle of MDPI and HDPI and is intended to be used when the app is displayed via a TV. Normally most apps don't need to provide resources for this density and it is not considered to be a "primary" density group. There also is NODPI which is used for resources that are not ment to be scaled to match the density of the device. (Android Developers, 2013s.)

Developers should provide different graphic assets for different density buckets. The OS chooses which ones to use, depending on the device's specifications. These assets should follow the given scaling ratio between the five density versions, 2:3:4:6:8, starting from medium and ending in xxx-high (FIGURE 5). The LDPI version isn't necessary to provide, as the OS scales the MDPI version in half. For example, launcher icon is 48dp in size so the baseline, medium version, should be 48*48px, HDPI at 72*72px which is 1.5 times the baseline, XHDPI version twice the size of the baseline at 96*96px and so on. (Android Developers, 2013i.)

FIGURE 5. Asset size comparison for different DPIs (Android Developers, 2013i)

3.2 48dp rhythm

The recommended target size for objects on touchscreen is 48dp which translates to about 9mm in physical size. This may vary slightly, but fits comfortably in recommended target sizes of 7mm to 10mm. (Android Developers, 2013j.)

Android documentation urges developers to use this 48dp rhythm in their design. It will result in objects never being too small to hit comfortably regardless of the screen they are displayed on (Android Developers, 2013j). Gaps between touchable UI elements are adviced to be 8dp.

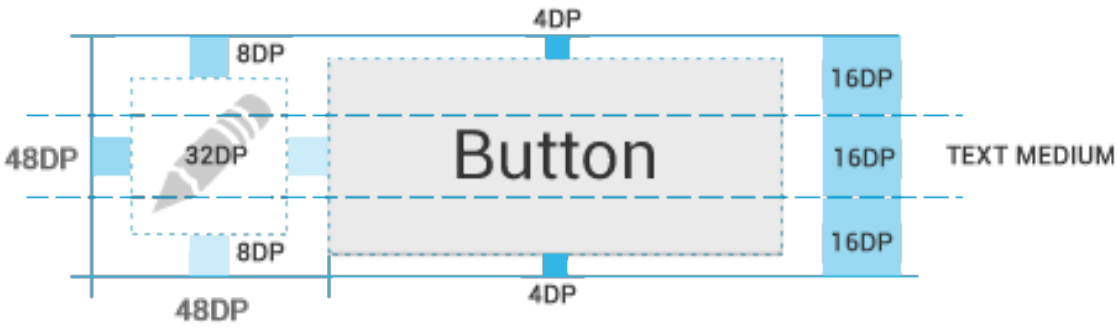


FIGURE 6. Metrics closeup (Android Developers, 2013j)

If the element isn't meant to be interacted with, the 8dp gap isn't absolutely necessary. These guidelines give you a very good starting point for your design. Elements have enough room around each other, and from a usability point of view, everything should be in order.

4 ANDROID STYLE GUIDE

The Android Style Guide is a document created by Google to help developers produce apps that are immediately familiar to users and are easy to use and learn. By following the guidelines a designer will dodge most of the usual usability mistakes and usually create a more refined product.

4.1 Design Principles

The Android style guide defines Android's design principles (Android Developers, 2013k). They fall into three different categories and are as follows.

Enchant Me

- Delight me in surprising ways
- Real objects are more fun than buttons and menus
- Let me make it mine
- Get to know me

Make Me Amazing

- Give me tricks that work everywhere
- It's not my fault
- Sprinkle encouragement
- Do the heavy lifting for me
- Make important things fast

Simplify My Life

- Keep it brief
- Pictures are faster than words
- Decide for me but let me have the final say
- Only show what I need when I need it
- I should always know where I am
- Never lose my stuff
- If it looks the same, it should act the same
- Only interrupt me if it's important

From a UX designer's perspective, all of these are important and you can read more about these in the style guide[1]. As a graphic designer, the most important of these are *Keep it brief*, *Pictures are faster than words*, *Only show what I need when I need*

1 http://developer.android.com/design/get-started/principles.html

it and *If it looks the same, it should act the same* as they directly relate to how things look.

In practice it comes down to things like making sure if something behaves as a spinner it should look like a spinner. If the user can't share the current page of the app they shouldn't be presented with a share button in the UI, even as a disabled option in the action overflow menu in action bar. Adding images of people in contact lists makes it much faster to browse, compared to when there are no images attached to the name. These are only few most obvious examples of how the design principles should be implemented in the graphic design.

4.2 Colors

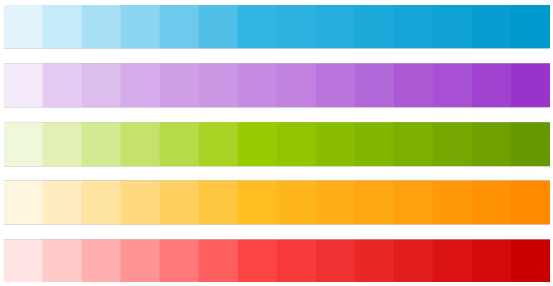


FIGURE 7. Android color spectrum (Android Developers, 2013d)

Android has a palette of predefined colors that appear in the OS. The designer is not restricted to only use these colors. It is encouraged to customize the app to reflect one's own brand colors. Although the documentation states that non-neutral colors should be reserved for emphasizing something, while providing good contrast between visual components and that red and green may be indistinguishable from each other to color-blind users (Android Developers, 2013d).

4.3 Icons

"An icon is a graphic that takes up a small portion of screen real estate and provides a quick, intuitive representation of an action, a status, or an app (Android Developers, 2013e)." Icons should be provided in multiple predetermined sizes so the OS can best choose which version to display for best outcome. The icon sizes can

be viewed in the Icon documentation[2] and are subject to change with different OS versions.

The launcher icon appears on the home screen or all apps view (FIGURE 8). The guidelines state that it should use a disctinct shape with a slight perspective as if viewed from above to be percieved to have some depth. (Android Developers, 2013e)

FIGURE 8. Launcher icon example (Android Developers. 2013i)

The action bar (FIGURE 9) appears on most apps excluding games and single serving apps such as flash light apps. It contains icons that provide access to most used relevant actions. Search, refresh and share icons are among the most usual ones. These icons should be flat, not too detailed and easy to grasp the consept at a glance (Android Developers, 2013e). These icons can be used with different opacities.

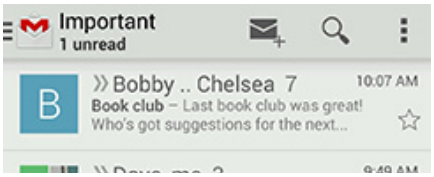


FIGURE 9. Action bar example (Android Developers. 2013c)

Small contextual icons (FIGURE 10) can be used all over apps. For example, the Gmail app uses a star to denote a bookmarked item, much like on the web based Gmail. These small icons should also be fairly simple and subdued like the icons in action bar.

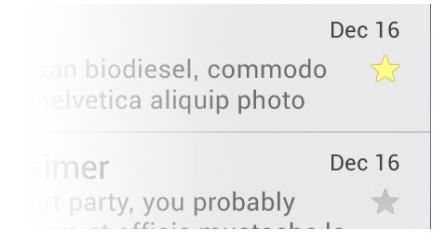


FIGURE 10. Contextual icon example (Android Developers. 2013i)

Notification icons (FIGURE 11) appear alongside with notifications. Not all apps generate notifications though. These icons are smallest of them all and should follow the lines of rest of the icon designs.

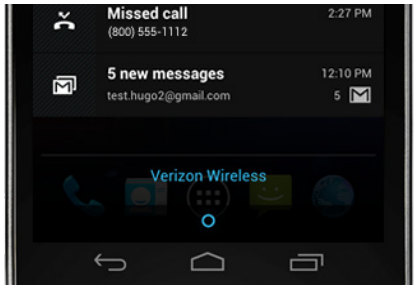


FIGURE 11. Notification icon example (Android Developers. 2013i)

2 http://developer.android.com/design/style/iconography.html

4.4 Typography

The default font for Android is Roboto which was introduced in OS version 4.0, Ice Cream Sandwich (Android Developers, 2013f). Developers can use custom fonts in their apps. Android supports True Type and Open Type fonts (Appcelerator Wiki, 2013). Developers should specify font sizes in sp (scale independent pixels) to ensure their app works well with system-wide scaling factor for text which users can select from their Settings app. “One sp is one pixel on a 160dpi screen if the user’s global text scale is set to 100%.” (Android Developers, 2013f.)

4.5 Android vs. Other Operating Systems

Operating systems have their own set of specifically designed UI elements. iOS uses rounded corners and gradients, at least pre-iOS7, and Windows Phone uses flat shapes with no 3D effects at all. Other conventions familiar from iOS like tab navigation at the bottom of screen is not adviced due to Android having the system buttons at the bottom. Instead, it is preferred to have tabs on top of the screen, under the action bar.

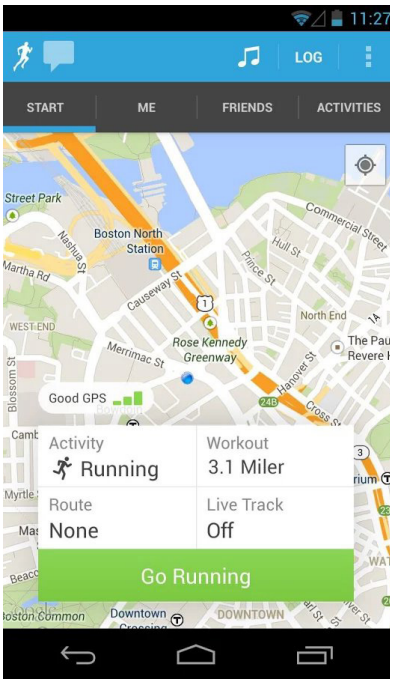


FIGURE 12. Runkeeper app on Android with tabs on top (Google Play, 2014)

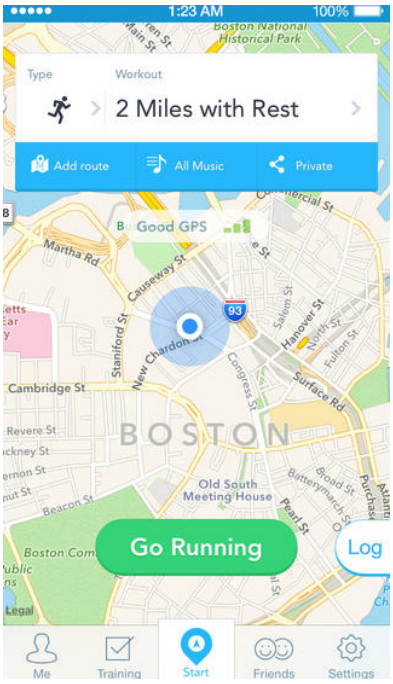


FIGURE 13. Runkeeper app on iOS, tabs at the bottom (iTunes, 2014)

Android's own conventions are well presented in their own documentation[3] and are subject to change with each major release. (Android Developers, 2014.)

FIGURE 14. Sampling of icons from Android (top), iOS (center) and Windows Phone (bottom) (Android Developers, 2014)

3 http://developer.android.com/design/index.html

5 USER INTERFACE ELEMENTS

Android has a plethora of ready-to-use building blocks for developers to use. There is no need to reinvent the wheel. These can be customized but it is adviced that developers keep these like they are unless changes are very necessary and only tweak the colors to fit their brand.

The ready-made elements range from tabs and grid lists to spinners, buttons and switches. The theme should handle most of the styling of these elements and often ther is no need to customize these at all. If the designer decides to make custom UI elements, they should make sure to follow the Android guidelines closely.

5.1 Element states

Elements in Android have different states:

- Normal
- Pressed
- Focused
- Disabled
- Disabled & Focused.

Normal is the default state, Pressed appears when the user touches the element, Focused is displayed when element is selected, Disabled renders as normal but with 30% opacity and Disabled & Focused is 30% opacity of Focused. Many of the UI elements have the states built in and require no further action from the developer. (Android Developers, 2013l.) If and when the designer wants to customize the elements, they might need to create different assets for each or some of the states.

5.2 Icons

Icons are used throughout the OS and they all need to be of a certain size to fit where they are designed to be used. Icons have two measurements that should be taken into account when creating them, full asset and optical square. Full asset is the physical size of the image. Optical square is the portion of the full asset the graphic is allowed to occupy (FIGURE 15).

FIGURE 15. Full asset and optical square of icons (Android Developers. 2013i)

TABLE 1. Icon sizes

Icon	Full asset	Optical Square
Launcher	48*48dp	48*48dp
Action Bar	32*32dp	24*24dp
Small / Contextual	16*16dp	12*12dp
Notification	24*24dp	22*22dp

For Google Play store the Launcher Icon must be 512*512px. Icons in the Action Bar will be used in different opacities, to represent different states like Normal and Disabled. Notification icons must be white. (Android Developers, 2013i.)

5.3 Building Blocks

5.3.1 Buttons

Buttons can have images and text in them. Image Only buttons should have an icon that can be easily understood, for example a magnifying glass in search button. For buttons with only an image, a background isn't necessary. Text Only buttons are best used when there is no easy way to represent with an image. A background isn't

necessary for text buttons either unless the designer really wants to draw attention to them, like Accept, Buy now and Sign up buttons. (Android Developers, 2013n.)

When the designer wants to customize buttons, they need to create three different assets for it: Normal, Pressed and Focused (Android Developers, 2013m). All those of course need to also be in different dpi versions.

While Android doesn't support vector assets, it is still possible to create stretchable buttons. This comes in the form of 9-patch. "A NinePatchDrawable graphic is a stretchable bitmap image, which Android will automatically resize to accommodate the contents of the View in which you have placed it as the background." (Android Developers, 2013m) It is a .png file with an extra border that is 1 pixel wide. The borders should be completely white or transparent on the parts that are not meant to be stretched. Black parts of the top and left border define stretchable area. You can optionally define the padding area with black portions in the bottom and right borders.

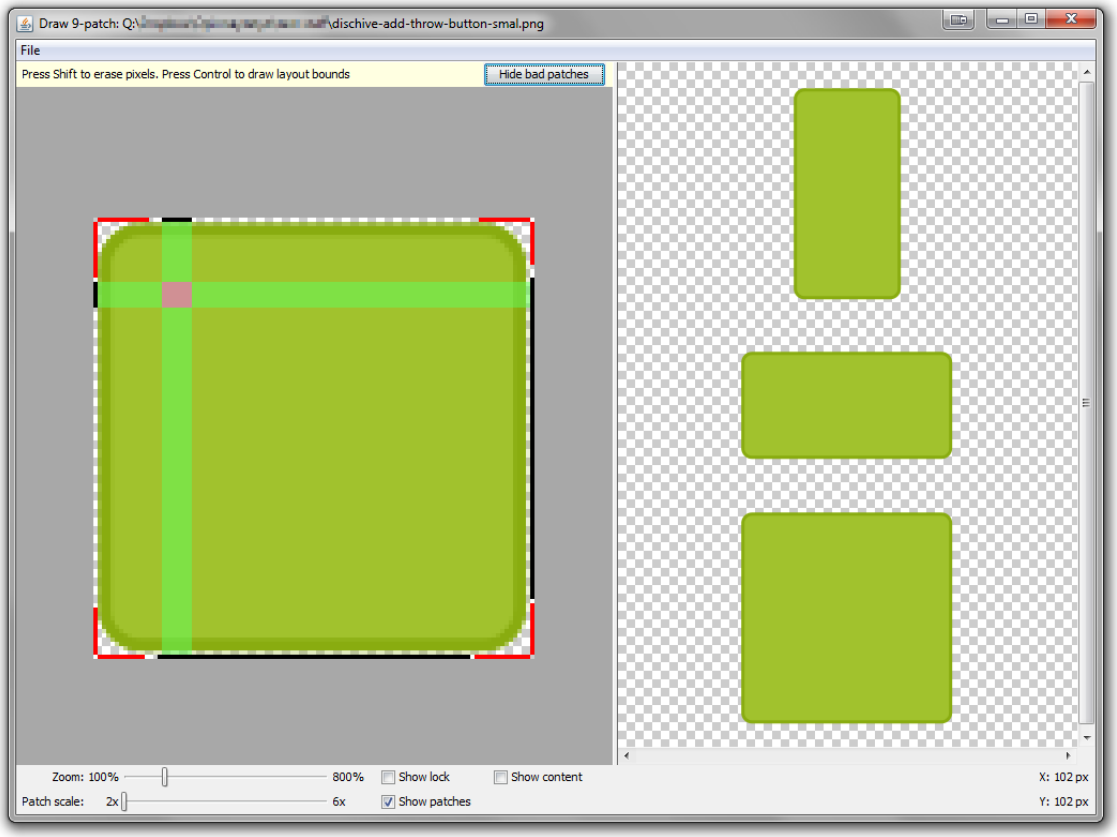


FIGURE 16. 9-patch button demonstrated in the Android SDK Draw 9-patch software

FIGURE 16 demonstrates how a 9-patched graphic works. The left pane shows the original asset with stretch regions (top and left black lines with green areas), paddings (right and bottom black lines) and optical bounds (red lines). On the right pane, there are three previews of how the 9-patch will behave once it is stretched.

Due to this mechanic a continuous gradient from top to bottom or side to side isn't adviced. It can look clumsy when stretched and banding can occur if the gradient is very subtle.

5.3.2 Switches and Sliders

There are three types of switches in Android, Check Boxes, Radio Buttons and On/Off Switches. These behave very much like on normal web forms. Check boxes allow you to choose multiple options from a set, while radio buttons are suitable for picking only one option. On/off switches are best used when the designer needs to toggle one option on or off. (Android Developers, 2013p.) These switches can be of any size but it is best to keep them close to the default sizes.

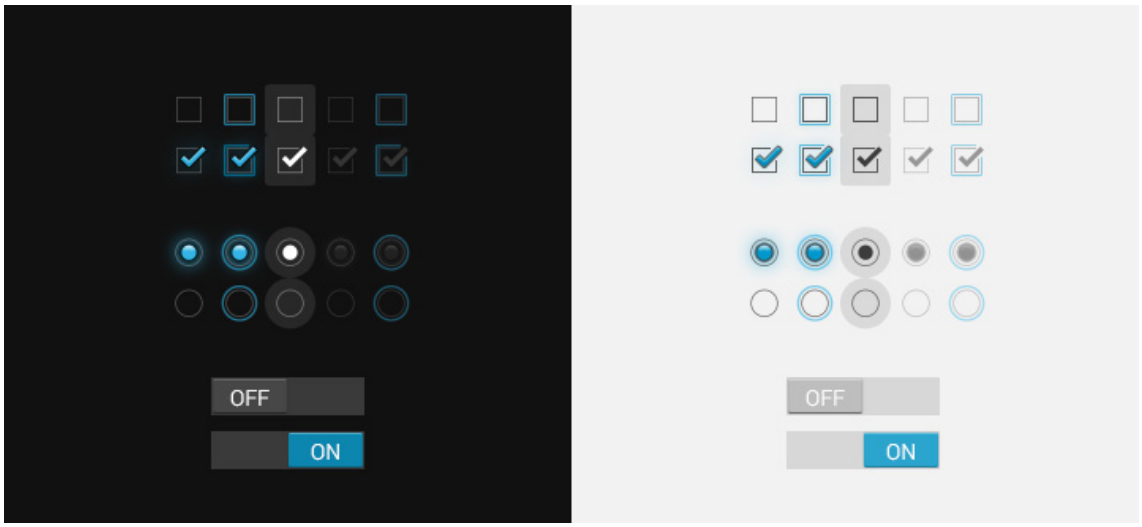


FIGURE 17. Adroid Holo switches in different states, dark and light theme

Sliders are interactive bars that "make it possible to select a value from a continuous or discrete range of values by moving the slider thumb." (Android Developers, 2013q) Most often these can be seen in form of volume sliders and seek bars in

video players. According to Android documentation, sliders don't have disabled & focused state.

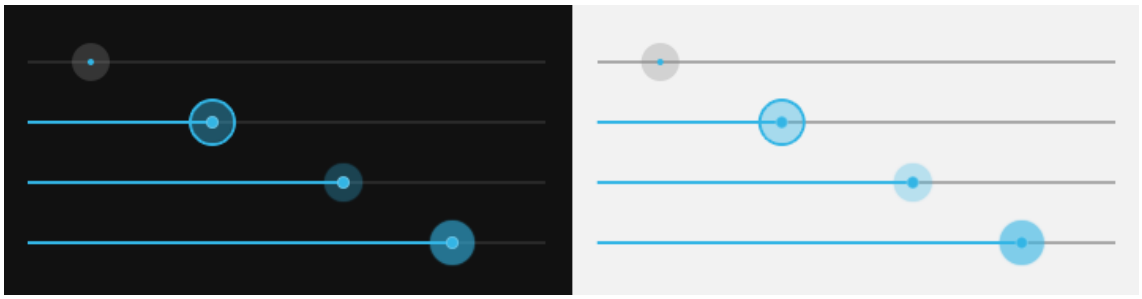


FIGURE 18. Android Holo sliders in different states, dark and light theme

5.3.3 Progress & Activity Indicators

"Progress bars are for situations where the percentage completed can be determined. They give users a quick sense of how much longer an operation will take." (Android Developers, 2013r.)

FIGURE 19. Android Holo progress bars, dark and light theme

Activity indicators tell a user that something is happening, but not how long it will take. There are two types of them, Activity Bar and Activity Circle. One example how activity bars can be used is to show that download is starting and the bar transforms into a progress bar when the download starts. Activity circle is a circular animation to be displayed when the app is loading something like a new email but can't show the content straight away. One shouldn't use "Loading" text with the activity circle as the animation already indicates that sufficiently. (Android Developers, 2013r.)

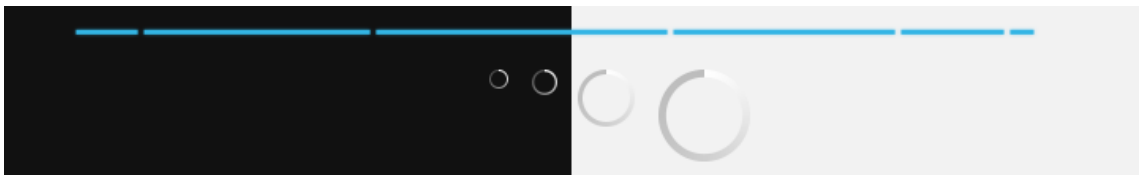


FIGURE 20. Android Holo activity indicators, dark and light theme

6 WORKFLOW

Cambridge Dictionaries Online (2014) defines workflow as “the way that a particular type of work is organized, or the order of the stages in a particular work process“. Workflow of each individual can vary immensly. Although there are some general principles a designer can incorporate to maximize productivity.

6.1 Prototyping Software

Prototyping with pen and paper is still as valid a method as any. Others prefer dedicated prototyping softwares. Some of the softwares offer native Android UI elements to play with. They might not be included in the initial install though, in which case one should look for them on the developer’s website.

6.2 Vectors and Artboards

While Android doesn’t support vector assets it is still a smart move to use vectors when creating one’s assets, either by using a vector editing program like Adobe Illustrator or using vector shapes in raster graphics editors like Adobe Photoshop. By working with vectors, it is possible to upscale assets if the need arises and make downscaling easier as paths can easily be tweaked to match pixel boundaries (Android Developers, 2013i). If one is not using vectors and needs to make a larger asset than what they are working with, a total remake of the asset is most likely needed for a sharp appearance in higher dpi screens.

It is also advisable to start with a large artboard when working with graphic assets. Android documentation suggests starting with an artboard with dimensions that are multiples of the target size. This will make the resize process easier and cleaner. (Android Developers, 2013i.) And of course it is easier to do small modifications to the graphics more easily than with small artboards where you can only zoom in to some extent.

Some like to work with all assets in one artboard and slice the document to get all assets independently. While this is an acceptable way of working, it is easier to have a separate artboard for each icon, at least in Illustrator CS5 and up. You can set the size for each artboard independently and export each artboard as a single image. It is easier to handle full asset / optical square sizes of each icon this way.

6.3 Naming Conventions and Asset Organizing

Android doesn't force the designer to use any predetermined file naming conventions. They offer guidelines and some best practices to help you keep the files organized and easy to understand for other people who work in a same team.

In an Android project all graphical assets are in a "res" folder, short for resources. Inside the res folder the files are divided by their use. Most likely a graphic designer is interested in folders with drawables. There can be many uses for different folders like drawable-ja (graphics optimized to be used with Japanese) or drawable-small-land-stylus (graphics optimized to be used in landscape mode on a low dpi screen with a stylus) (Android Developers, 2013s). Often this amount of specificity isn't needed and you get by with just the dpi specific folders:

- drawable-mdpi
- drawable-hdpi
- drawable-xhdpi
- drawable-xxhdpi

A graphic designer can have the assets on their computer in a folder with any name they desire, but an app project demands for them to be named in certain way. The folder names are easy to understand like they are so there is no real reason to name them any other way. This makes the co-workers' lives easier, as they don't have to decode the folder names to understand which is which.

Naming the image files is up to the designer, but it is wise to follow conventions set by the default images of Android OS. TABLE 2 explains few but these are not all of available types of assets.

TABLE 2. Different drawables names

Asset Type	Prefix	Example
Action bar	ab_	ab_bottom_solid.9.png
Button	btn_	btn_check_off.png
Dialog	dialog_	dialog_full.9.png
Divider	divider_	divider_horizontal.9.png
Icon	ic_	ic_star.png
Menu	menu_	menu_hardkey_panel.9.png
Tabs	tab_	tab_selected.9.png

Naming different icons is more liberal but few good practices have emerged. TABLE 3 demonstrates suggested icon naming conventions.

TABLE 3. Icon naming convention suggestion (Android Developers, 2013e)

Asset Type	Prefix	Example
Icons	ic_	ic_star.png
Launcher icons	ic_launcher	ic_launcher_calendar.png
Menu icons and Action Bar icons	ic_menu	ic_menu_archive.png
Status bar icons	ic_stat_notify	ic_stat_notify_msg.png
Tab icons	ic_tab	ic_tab_recent.png
Dialog icons	ic_dialog	ic_dialog_info.png

Buttons have several states and those need different assets. A fully functioning button requires at least three different assets: Normal, Pressed and Focused. Normal state doesn't need any suffix as it is the default state of the button. *Disabled* and *Disabled and Focused* images are optional. Those states can be rendered with the other three states with different opacity.

TABLE 4. Button states

State	Suffix	Example
Normal	*no suffix*	btn_check_off.png
Pressed	_pressed	btn_check_off_pressed.png
Focused	_focused	btn_check_off_focused.png
Disabled	_disabled	btn_check_off_disabled.png
Disabled and Focused	_disabled_focused	btn_check_off_disabled_focused.png

6.4 File Sizes and Formats

The file size limit of an Android app (APK) is 50MB, which can be expanded by two 2GB extension files (Android Developers, 2013t). It is advisable to keep your app as small as possible, even though there are no limitations on how big a file one can download over cellular network. Therefore it is to one's best interest to keep image assets as small as possible, without compromising the quality.

The norm today is to use JPG for images without transparency and little to no solid areas of one color like photos, and PNG for images with transparency, icons and UI elements. Many developers use additional software to reduce file size of images even further than the software it was created in.

6.4.1 WebP

WebP is an image format designed by Google. It provides lossy and lossless compression and animation support. Transparency is supported with both compressions. The file size of lossless WebP images are promised to be 26% smaller than the equivalent PNG file. For lossy image the file size reduction is up to 34% when compared to JPG images of same quality. (Google Developers, 2013.)

Android supports WebP from Ice Cream Sandwich (4.0) and up. Chrome, Opera 11.10 also have native support and Internet Explorer supports WebP via the Google

Chrome Frame plug-in. (Google Developers, 2013.) While WebP isn't a foolproof solution for smaller file sizes globally it is possible to use it for apps when the app is aimed at Android version 4.0 or newer. In 2013 Google started to serve WebP images in Google+ app for Android where photos and images comprise the majority of bytes, resulting in 50% savings just by using a different file format (Google I/O 2013 - WebP, 2013). This change affected only the content, not the UI, but demonstrates the power of a different file format can have.

Windows doesn't natively support WebP but there is a codec[1] that allows one to view thumbnails in Windows Explorer and programs that use Windows Imaging Component (WIC). Other software support is entirely up to the developers but many imaging software do support WebP. On a Mac there is no graphic-designer-proof way to view WebP files natively yet. The most convenient way is to use Google Chrome or most other browsers using Weppy plugin[2].

There are tools like IMG2WEBP[3] to convert images to WebP format but those are hardly incorporable to the workflow as they can convert only one image at a time. There is a file format plugin[4] for Photoshop created by Toby Thain. To use this one needs to download a proper version from the Telegraphics website and extract the file format to `[Photoshop folder]\Required\Plug-Ins\File Format` folder. With this, one can open and save WebP images in Photoshop. The only drawback is that at the moment, this plugin doesn't support transparency.

6.4.2 PNG Optimizers

Different optimizers are a great way to bring down the filesize of PNG images. By using these programs, the size of PNGs can be reduced by a tremendous amount. In FIGURE 21, three PNGs with size of 444 by 416 pixels are compared. The first is the

1 https://developers.google.com/speed/webp/docs/webp_codec

2 http://seiryu.home.comcast.net/~seiryu/weppy.html

3 http://img2webp.net/

4 http://telegraphics.com.au/sw/product/WebPFormat

original out of Photoshop's Save for Web dialog at 155KB. The second is TinyPNG[5] optimized version with a size of 45,3KB. For the sake of testing the third one is run through TinyPNG seven times and it is 25,8KB in size. At this point jagged edges and very small changes in the noise pattern in the middle can be seen.

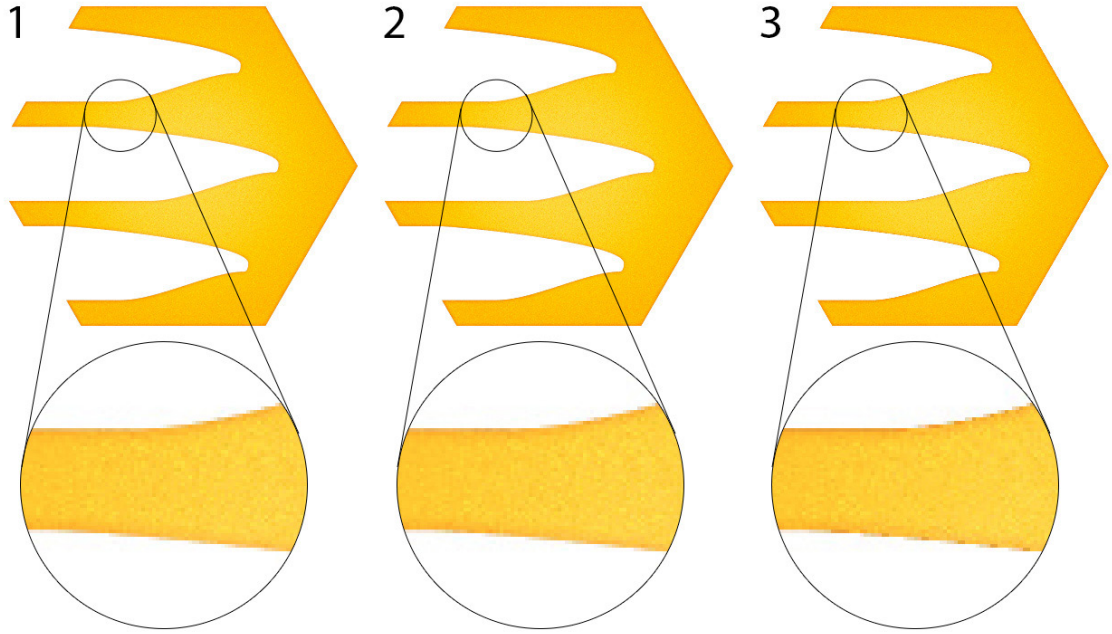


FIGURE 21. PNG optimization comparison with 400% zoom

There are too many PNG optimizers out there to list, but here are few features to keep in mind. Some of the optimizers are lossless that will perfectly preserve how the image looks but will achieve less file size reduction than lossy ones. With lossy compression one needs to judge how many times the image is compressed and see how it affects the outcome. One essential feature of the optimizer is that it can compress multiple images or specified folders in one go. And if it's an online software it should be able to provide optimized images as a downloadable archive instead of individual files one at a time.

6.5 Automation

In modern imaging software like Photoshop there are a few features to help you with repetitive tasks like resizing and saving an image. These will make the work

5 https://tinypng.com/

easier and faster if one takes the time in the beginning to create actions that will automate majority, if not all, of the steps.

6.5.1 Actions

Actions are Photoshops way of automating different steps one would otherwise do manually repeatedly. The steps can be recorded and then redone for other images just by clicking a button instead of repeating each step manually.

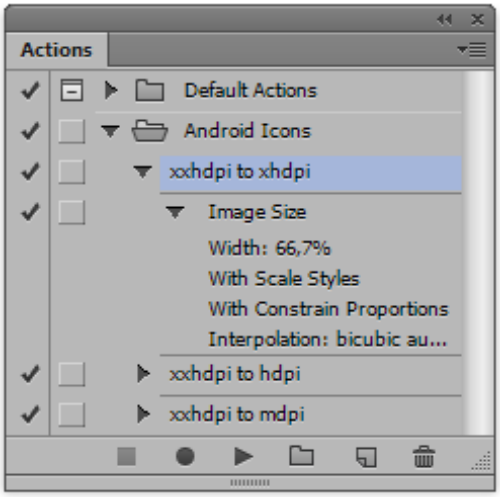


FIGURE 22. Photoshop Actions panel

To create an action in Photoshop for converting an xxhdpi image to xhdpi the following steps will be needed with any document open:

1. Open the actions panel: `Window → Actions`
2. Create a set for your actions from either the Actions panel menu or bottom of the panel. This is optional but it is good practice to keep your actions organized.
3. Create new action from the panel menu or bottom of the panel.
4. Recording starts and everything you do will be recorded in the action. Resize the image: `Image → Image Size`... and specify the width to 66,666% and hit OK.
5. Stop recording.

Now ther is an action that generates an image ⅔ size of the source. After this, just create similar actions for *xxhdpi to hdpi* (50%) and *xxhdpi to mdpi* (33,333%). This example assumes the starting point is xxhdpi version and always scale image from that version instead of the already resized ones. If you create the original in xxxhdpi or larger the percentages will vary.

One should keep in mind whether to scale layer styles with the image resize or not. One could also include the saving the images in actions, but Photoshop insists on

having absolute paths to the folder instead of relative paths. So it is not possible to have the action use xxhdpi image as a source, do the steps, go up one folder and select drawable-mdpi folder and save a version there for example. Instead we create an action for each target dpi and use batch processing to alleviate the problem.

6.5.2 Batch Processing

Batch processing in Photoshop is a way to run actions on multiple images in a row. One can specify a folder filled with files and what action to run. One can additionally choose to override the file naming scheme and where the files are saved in.

Adobe Illustrator allows batch processing much the same way as Photoshop does, but with one huge caveat. Instead of saving the files automatically, it prompts the save dialog and you need to specify where to save each file and with what name. This completely defeats the purpose of batch processing the files.

To start a batch process, you need to navigate to `File → Automate → Batch...` in Photoshop. In the dialog window (FIGURE 23) you need to select the action you want to use, in this case the *xxhdpi to xhdpi*. The source are the files that will be affected by the action. There are a few options but folder is often most convenient.

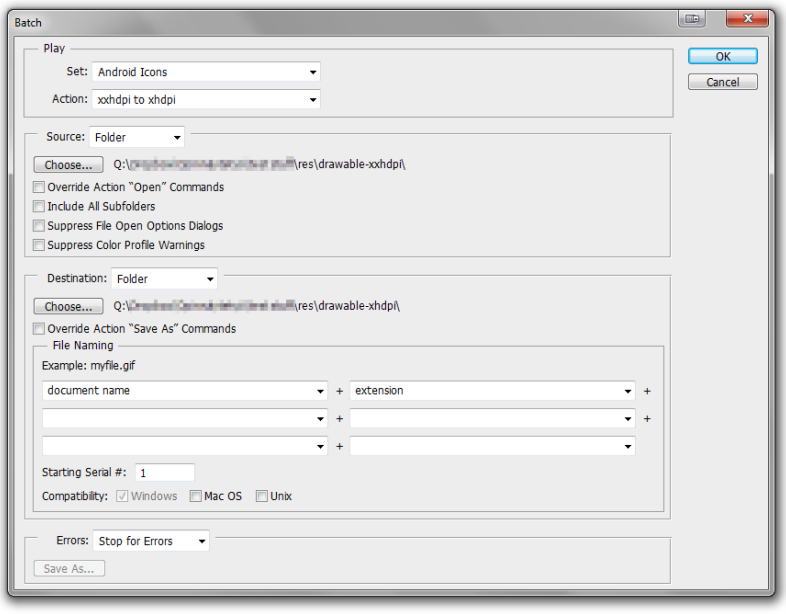


FIGURE 23. Photoshop Batch dialog window

Destination will be the folder the files are saved in after the action is performed. If *none* is selected the files will stay open in Photoshop instead of being saved and closed. After setting all the parameters hit OK and Photoshop will start processing the images. Make sure to afterwards check the images that they actually are the right size and if something has been resized poorly, like borders etc, correct manually or try another method. Sometimes it is best to redraw whole icon for certain dpi.

6.6 Android Asset Studio

Android Asset Studio[6] is a collection of helpful tools for Android developers including icon, 9-patch and style generators. Some of the icon generators are only good for quick comparison on how the icons look on different sizes and while the generated icons can be saved as image files it is ultimately faster to use Photoshop actions to generate the icons and automatically put them in correct folders.

6.6.1 9-patch

Android SDK has a Draw 9-Patch program which can be used to generate 9-patch images fairly easily. It lets you specify the stretch regions, content paddings and optical bounds with visual editor and has a good preview on the right side (FIGURE 24). The only downside is that you will need to do this for every dpi version of your 9-patch images. Depending on the case one could possibly get by with just one asset but if it has rounded corners the corner radius will appear to change from dpi to another.

9-patch images are easier made with a generator from the Asset Studio than in Photoshop. You get a similar visual editor for the stretch, padding and optical bound regions as you do with the SDK version and you can download the images as one ZIP file. Scaling preview is missing though. After that you need to manually sort the files in to the corrent folders for each density bucket you're supporting.

---

6 http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

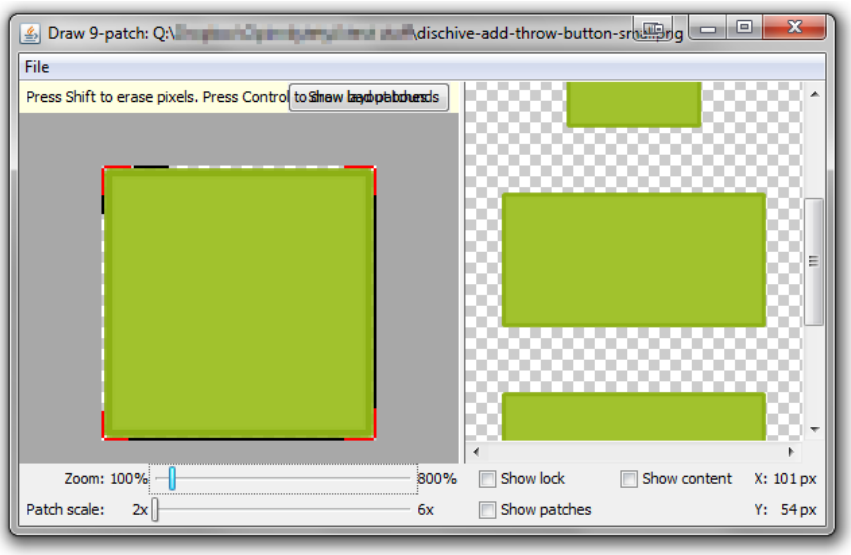


FIGURE 24. Android SDK Draw 9-patch software

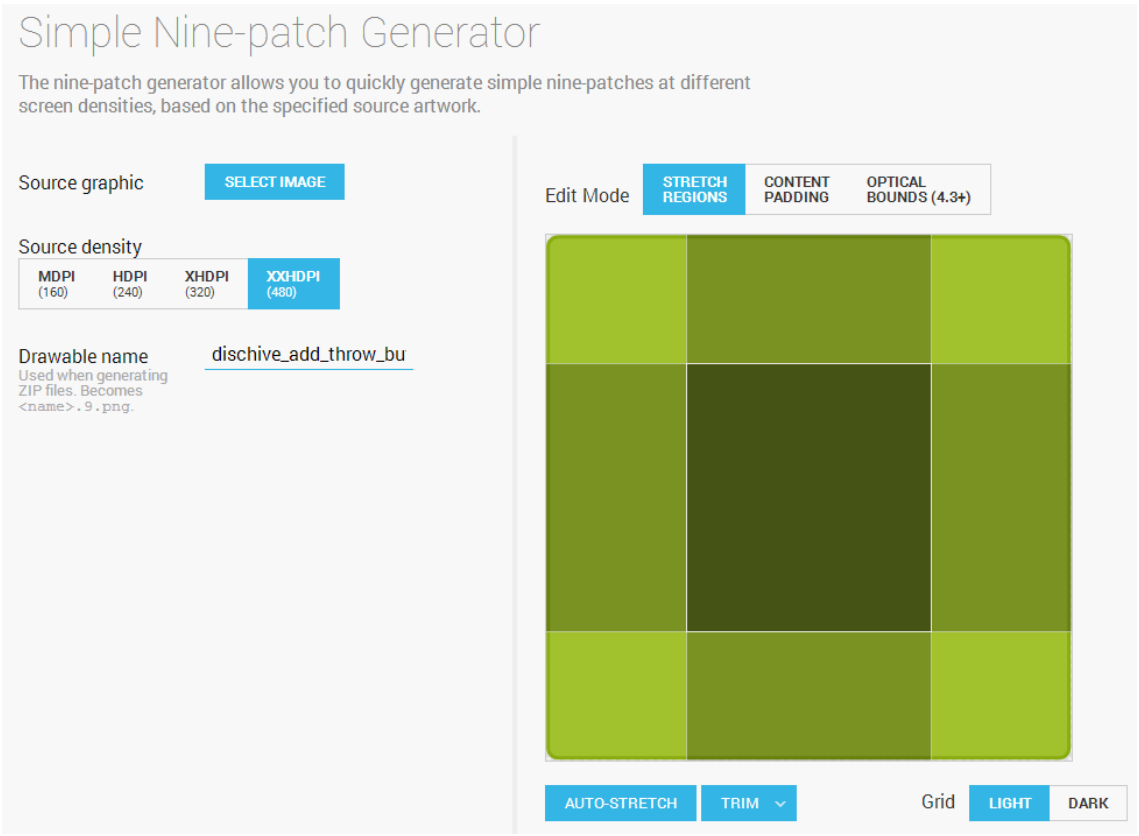


FIGURE 25. Nine-patch Generator from Asset Studio web page

The SDK 9-patch software has one advantage over the Asset Studio one, multiple stretch regions. This means that more than one region can be defined that stretch on a side, enabling things like speech bubbles with the tail centered instead of in a corner. FIGURES 26, 27 and 28 demonstrate the effects of different stretchable regions on a speech bubble graphic.

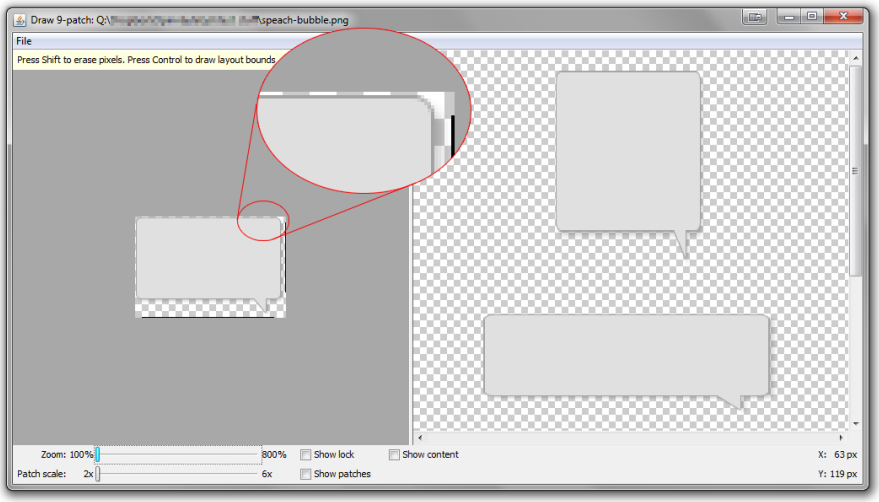


FIGURE 26. No stretchable regions. Bubble and tail get deformed.

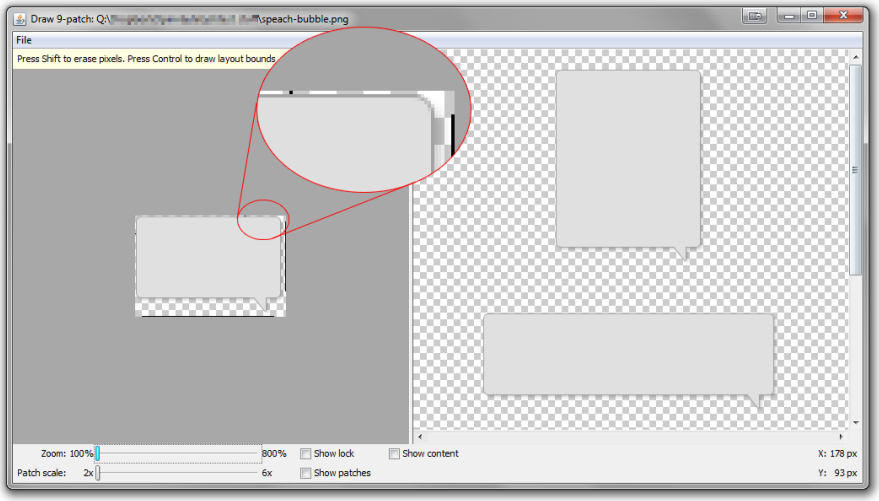


FIGURE 27. One horizontal stretchable region. Bubble tail stays on right side.

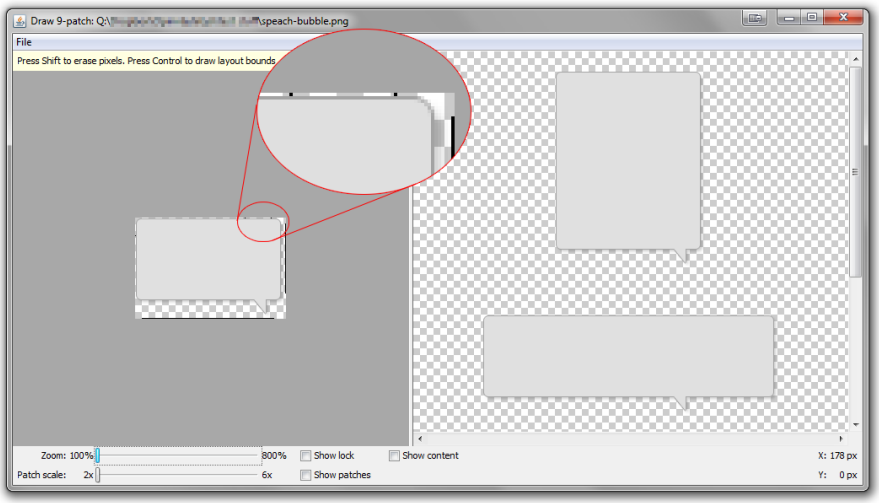


FIGURE 28. Two stretchable regions. Bubble tail position varies but doesn't get deformed.

6.6.2 Action Bar Style & Holo Colors Generators

If the app uses the default Holo theme as basis for the theme the Action Bar Style Generator[7] makes the graphic designer's life easier. You can define different colors to match your branding and the generator creates necessary files for the theme. It also has an excellent preview of how the theme will look like. After the color values have been specified, the assets can be saved as a ZIP file and send it to the coders.

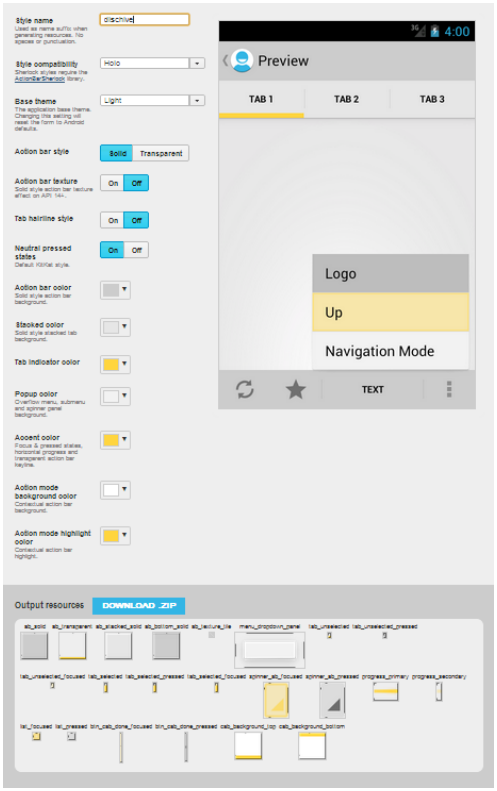


FIGURE 29. Action Bar Style Generator

It is also possible to customize Holo styled elements to fit your brand with Android Holo Colors Generator[8]. The brand color can be specified and which elements are needed, and then downloaded as an archive with all dpi versions.

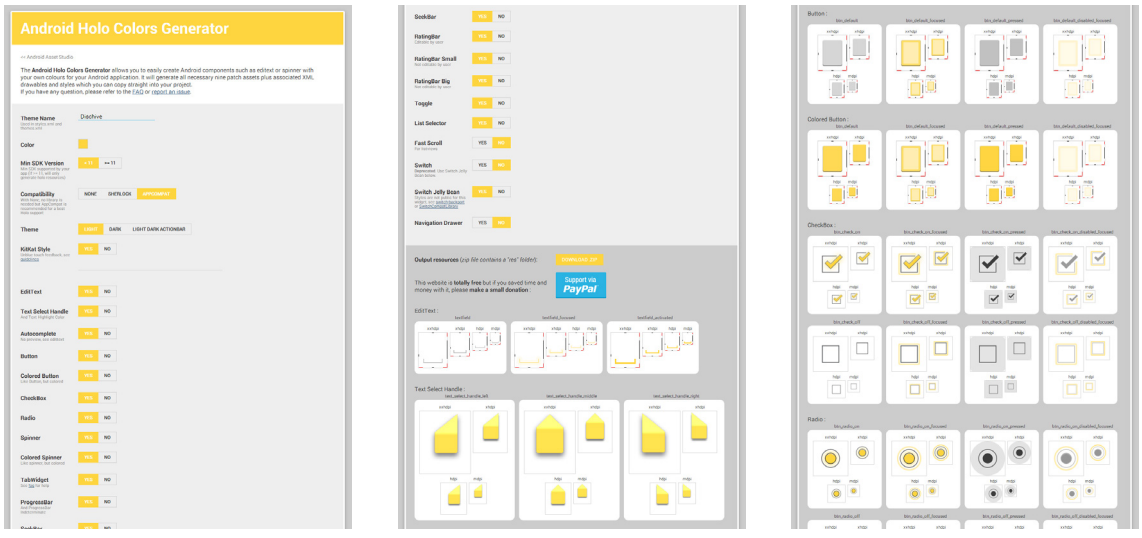


FIGURE 30. Android Holo Colors Generator

7 http://jgilfelt.github.io/android-actionbarstylegenerator/

8 http://android-holo-colors.com

7 EXAMPLE CASE – DISCHIVE

Dischive is a service for disc golfers for saving their scores and following their progress through statistics. Scores are saved via a smartphone app and sent to a database. On the website, users can view their scores and stats as well as interact with other users and form teams and competitions. The app features will include such as score keeping, friends' scores on a current round, course maps and disc bag management.

It has been in development for about a year with first alpha tests done in summer of 2013, with a web app version for proof of concept. After a successful test period the Android app development started with defining the core features and designing an easy to use interface around those features.

Users were classed in two basic categories, normal users and advanced users. Normal users want to only track how many shots they've made in each hole and maybe view them on the website later. Advanced users desire more control over their statistics and will input their discs per throw, even position where the shot was made and where it landed if given the option. They also want much more detailed statistics on the web portion of the service.

7.1 User Interface

The goal for the app was to look minimalistic and be easy to use while still having many advanced features for users that desire them. First we did some benchmarking (Appendix 1.) with existing disc golf apps and construed different throw input methods and biggest pitfalls to avoid. Afterwards, a prototype version was developed as a pure web app, to test the proof of concept and to receive initial user feedback from small test group on multiple devices. The web app (FIGURE 31) gave us a good understanding of which direction to take the Android version and what functionality was essential and revealed what was completely missing.

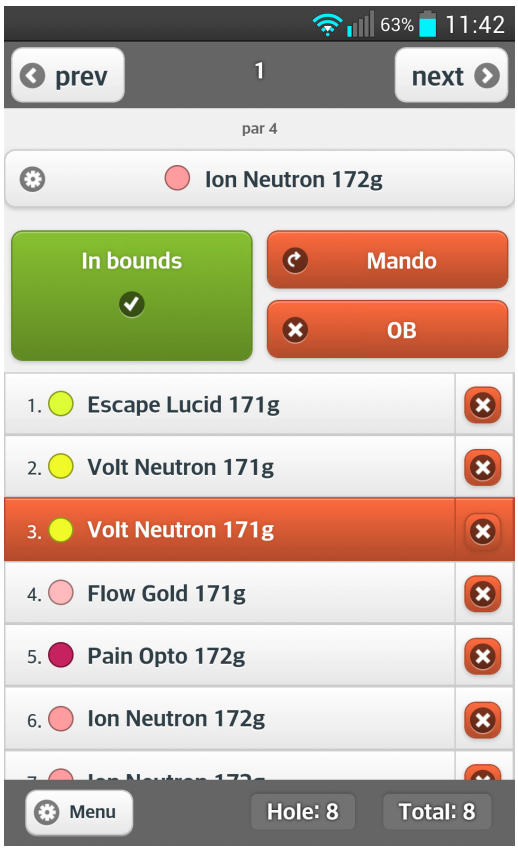


FIGURE 31. Dischive web app UI

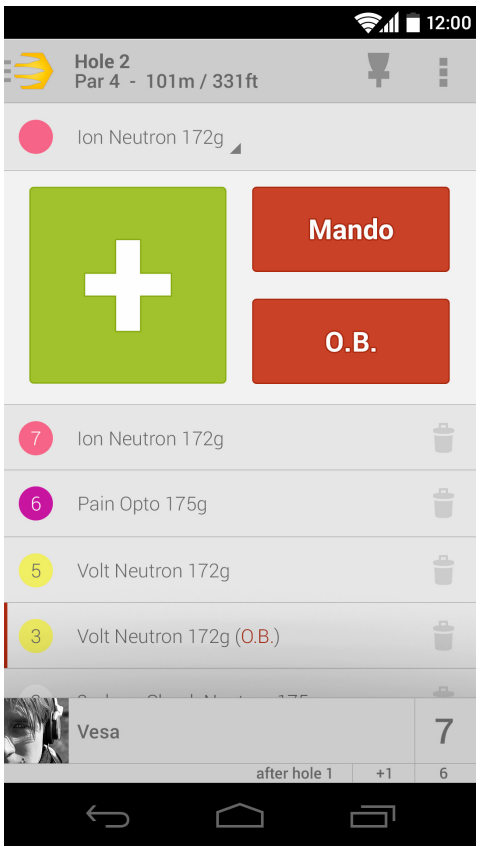


FIGURE 32. Dischive Android app UI

The first prototype was made with default Android Holo Light theme. It served the purpose very well but later on more customization was needed. This came as color changes, custom icons and elements. The first version of the Android app would have all the features the web version had but in a more polished way and the back end would be improved and compatible with future features.

7.2 Design Choices

7.2.1 Branding

Initially after settling for a name a visual identity was created around it. In the beginning the web portion of the service was supposed to be called Dischive and the app Discbee. Later this was discarded as it would potentially only confuse users, leaving everything under that Dischive brand. After the first alpha tests, it was

decided that the logo needs some additional work as it was more of a football with strange colors than a representative of a hive and didn't reflect disc golf at all.

FIGURE 33. Old Dischive and Discbee logos

FIGURE 34. New Dischive logo

The new logo incorporates the octagon shape of a hive cell with golf disc cut outs. It is also has a bit more movement to it than the old one. Also the brand colors were adjusted towards a lighter scheme.

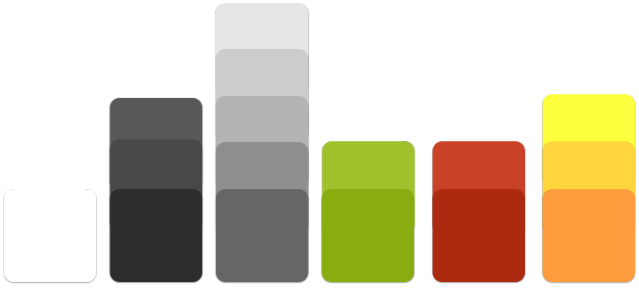


FIGURE 35. Dischive colors

7.2.2 Web version vs. Android version

The first functional prototype UI was most affected by jquery mobile default themes. These enabled fast iteration and development. It was possible to achieve a functional and reasonably eye-pleasing design very fast with minimal effort. While it was given that it won't be used in the native apps, we'd still have a functional product for people using platforms we don't have a native app for yet, so it won't be wasted effort to make it look even moderately nice. Some of the planned features just were impossible or too hard to implement on the web app.

Jquery mobile offered support for gestures on the web app. These were used for switching between holes while marking throws. These were a little buggy at best and buttons for switching the hole needed to be available. On Android, the buttons were scrapped and swiping is the only method to switch between holes. This allows for a more minimal look and frees up space for other functions.

7.2.3 Android Components

One of the main priorities of the Android app was to make the user experience better and more fluid. For the Android version, a decision was made to follow Android style guides very closely to provide an easy and consistent experience for the users. This would also benefit the product as the components will get updated as new versions of Android are realeased.

On the web version, the course selection was only one list of all the courses with the ability to filter it by text. Text input on mobile devices is fairly awkward even in the best conditions. Something had to be done to make the experience better. Most players play regularly only on a handful of courses thus it made sense to give the user the ability to mark their favorite courses, and those would be offered first when starting a round. Also the app can filter nearby courses based on the user's location. These options (All, Favorite and Nearby courses) are presented using a scrollable tab component to give the user a hint that there are more than one category of courses.

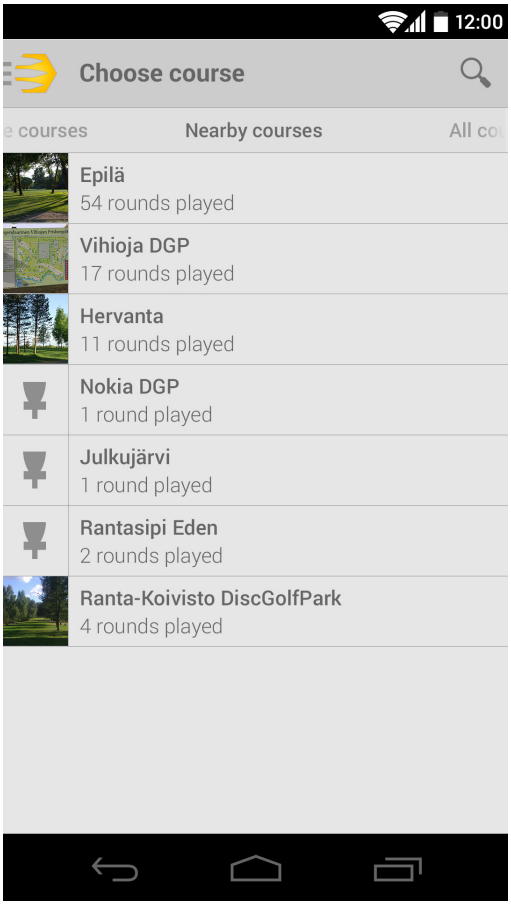


FIGURE 36. Dischive course select screen

7.3 Problems

In our user testing a few problems came to light that either bothered the users or made the app unnecessarily hard to use.

7.3.1 Marking Shots

In the usual score keeping apps, you could only mark the amount of shots you made on a certain hole, maybe how many of those were putts. You couldn't get any data about any penalties that may occur. To please the advanced users group it was needed to add buttons for the common penalties like Out of Bounds and Missed Mandatory. User setting will be implemented to swap the places of the buttons to better suit user preferences in which hand they used to operate the app.

One obvious flaw in design was the list of throws on smaller screens. After four shots, you couldn't see the shots you just recorded as they appeared out of screen. You could still scroll to the score but it wasn't convenient enough. In the web app the buttons also scrolled out of view and you needed to scroll back to input more throws. The shot list needed to be scrollable on its own and have an option for users to have the list appear in reverse order, last shot on top.

7.3.2 Selecting Discs

In the beginning the disc select list was just an alphabetical list of user's discs. It was fairly hard to find the disc you were looking for if the user had more than five discs in the list. Afterwards, an "in Bag" feature was added, which allowed users to select which discs they had in their bag and the disc selection only showed those. Still it was hard to tell the discs apart with just a list of disc names. A simple and functional solution was to add the disc color in front of the disc name. This way, users could easily find the disc they were looking for, most times even without reading the name and just going by the color of the disc.

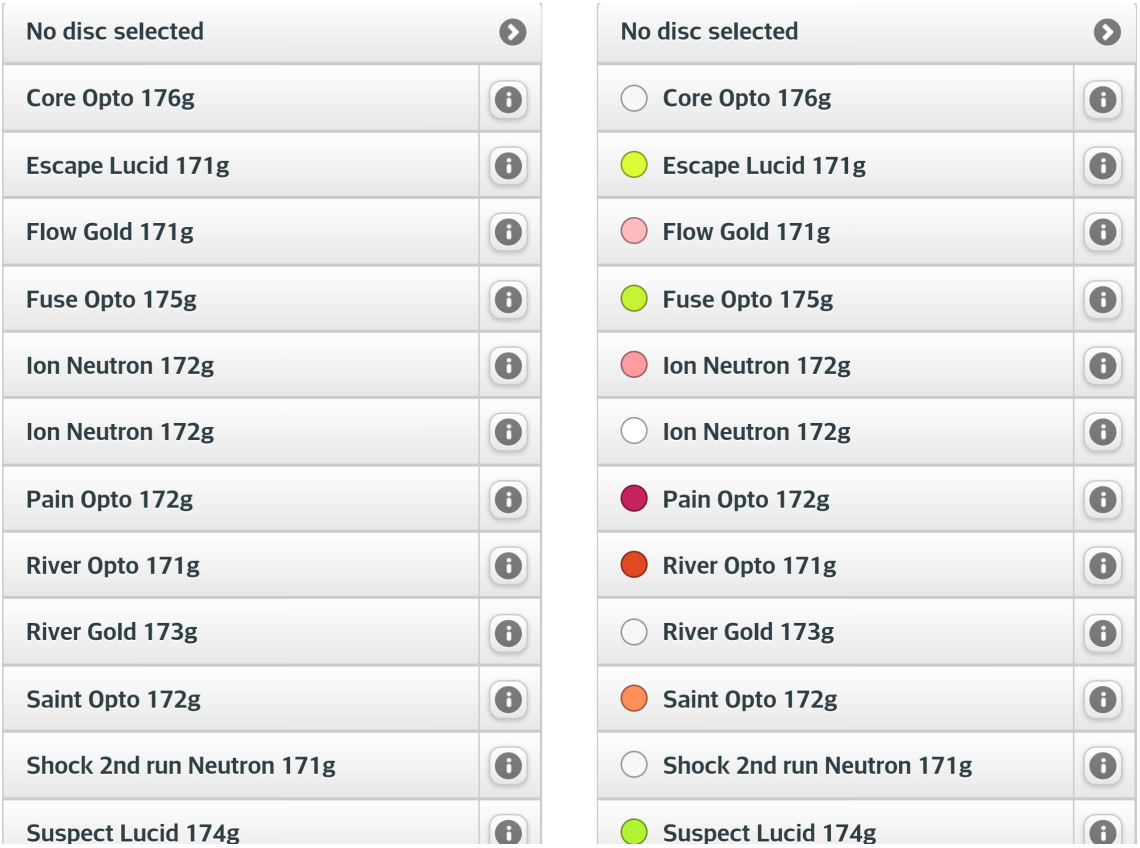


FIGURE 37. Disc list in the web app with and without disc colors

In the Android app, the disc color is put in use more when applicable. For example, it makes the End of the Round screen more visual and gives an idea of which discs were used.

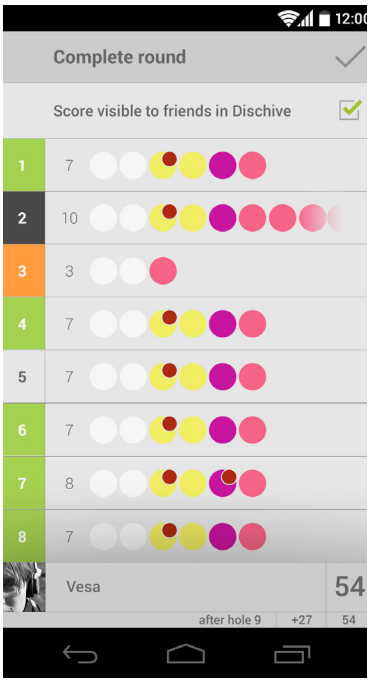


FIGURE 38. Dischive End of Round screen

7.3.3 Score vs. Throws

In professional disc golf scores are recorded by the total number of throws. In more casual settings, players might use an over/under (+/-) system to annotate performance compared to par of each hole, i.e. on a par 4 hole -1 if player holed out with 3 throws.

Due to technical and time restrictions, the web app only had the number of throws visible. Some players found it to be offputting and demanded to have a “score after

hole X" over/under option. Other players preferred to not "know" the score and just record throws and calculate the score after the round themselves. Not knowing the score made playing easier for them as they got less annoyed when doing poorly and less anxious when they were doing well. In the final app, this will be a user defined option and the round end screen will show both the amount of throws and over/under score.

7.3.4 Hole Information

Most score keeping apps only allow one tee and one basket placement for each hole. In reality, each hole may have two of each, sometimes more, with all the combinations. Initially, it was decided that Dischive would support different course layout configurations, so users could choose which variation they would play, instead of being forced to decide it before the round. Also this would allow the statistics to represent the correct holes. For example when hole 5 on a course has a par 3 tee and par 4 tee, in Dischive both would still be hole 5 for the same course but just different variations while rest of the holes are the same regardless of the hole 5. In other apps users would have two different rounds for each case, the par 3 and par 4, which you couldn't change after starting the round.

Due to different configurations, it was important to show hole information in the score keeping screen and allow to change the configuration on the fly if needed without it being too obtrusive. The Nice to Know info, like hole distance were added in the app.

7.4 Future of Dischive App

After the public launch of Dischive Android app the development continues and new features will be introduced. These won't be in the first release, since they need a considerable amount of testing.

The aim is to make the score-keeping as easy as possible. Users often have their go-to discs for different holes and the app could learn this. After a user has been using the app for a while the app could suggest a disc on each hole based on what the user has thrown before. This would cut down the time needed to mark the tee throw on most local courses the user frequents.

A more robust change to score-keeping might come with NFC (near field communication) stickers. This won't be a feature for the professional players as you can't use discs that have been altered in any way in competition. The player could have an NFC sticker under the disc which when held close to the mobile device could tell the app which disc the user is throwing. This would remove the need to even take the device out of bag or pocket and speed up the score keeping process considerably. Later the disc manufacturers could also start including NFC chips molded straight into their discs. Initial tests for this method have been made but some major obstacles are still preventing this becoming a reality.

Support for tablets is an important feature for Dischive app. This would enable the app to use multiple panes in one view. Tournament officials could use the app to record scores of multiple contestants and spectators could follow the scores in real time with their own tablet or phone.

Dischive aims to be a social app for groups of people who enjoy disc golf together. This could come in a form of group rounds. Users could open the app at course and bump their devices together, using features like Android Beam, to mark that they are throwing the round together. The users could see each others' scores during the round and in the

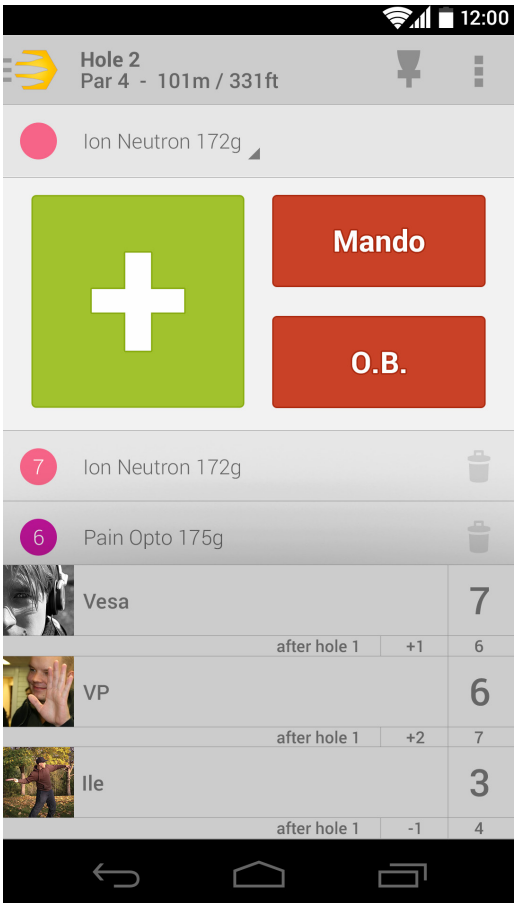


FIGURE 39. Dischive score keeping screen with friends' scores

web portion it could be visible who the user was with and how they did if the user wishes to do so. One of the main principles of Dischive is to not show the users scores publicly unless the user shares them to certain group or everyone. In other services where every score is public the results can get distorted as users are too embarassed to share their worst scores.

One final huge feature would be a course editor with vector shapes. Course maps are very important for users new to the course. They rely on it to know which basket to throw in and where the next hole is etc. There are services online that offer course maps but nothing can beat the convenience of having the course map with you in the app. Also courses tend to change over time and many services don't update the maps or don't know that the course has changed. Giving the users ability to modify the courses and maps would provide a great way to keep the courses up to date. Later the users perhaps could mark tees and basket placements with gps coordinates via mobile devices if the coordinates prove to be reliable enough. This would open up possibilities to show approximately how long each hole and their variations are, what kind of distance is left to throw after teeing and directions to the next hole.

8 CONCLUSION

Making graphics for an Android app isn't technically very challening. After you get familiar with the building blocks it is fairly straight forward. You need to know which convetions are native to Android and avoid their counter parts from other operating systems. The developer documents on designing for Android are an outstanding resource for referencing how certain things should be done on top of being the most up to date information on the OS.

While generating graphics for Android doesn't demand a tremendous amount of skill, it still can be irritating to jump through all the hoops needed to produce all the different assets. Industry standard software doesn't natively support all the needed file types, making it necessary to use multiple softwares for generating 9-patch versions of certain graphics for example. Also it takes time for a new file type to reach high enough popularity for software developers to add support in their software, not to mention the time it takes to work out all the issues related to that file type in the software.

Big part of the Dischive app development was spent figuring out how to make the app easy to use. This provided good foundation for the later design choises about the look and feel of the app. First alpha tests helped greatly in streamlining the app and figuring out which features were must and which could be either incorporated later or left out entirely. In the end Dischive didn't need that many custom graphic assets and Android Asset Studio made the designers portion of the development very fast.

REFERENCES

Android Developers. 2013a. Read 12.12.2013.
http://developer.android.com/about/index.html

Android Developers. 2013b. Read 12.12.2013.
http://developer.android.com/about/versions/kitkat.html

Android Developers. 2013c. Read 12.12.2013.
http://developer.android.com/design/style/themes.html

Android Developers. 2013d. Read 12.12.2013.
http://developer.android.com/design/style/color.html

Android Developers. 2013e. Read 12.12.2013.
http://developer.android.com/design/style/iconography.html

Android Developers. 2013f. Read 14.12.2013.
http://developer.android.com/design/style/typography.html

Android Developers. 2013g. Read 14.12.2013.
http://developer.android.com/guide/practices/screens_support.html

Android Developers. 2013h. Read 14.12.2013.
http://developer.android.com/about/dashboards/index.html

Android Developers. 2013i. Read 14.12.2013.
http://developer.android.com/design/style/iconography.html

Android Developers. 2013j. Read 14.12.2013.
http://developer.android.com/design/style/metrics-grids.html

Android Developers. 2013k. Read 14.12.2013.
http://developer.android.com/design/get-started/principles.html

Android Developers. 2013l. Read 16.12.2013.
http://developer.android.com/design/style/touch-feedback.html

Android Developers. 2013m. Read 16.12.2013.
http://developer.android.com/guide/topics/ui/controls/button.html

Android Developers. 2013n. Read 16.12.2013.
http://developer.android.com/design/building-blocks/buttons.html

Android Developers. 2013o. Read 16.12.2013.
http://developer.android.com/guide/topics/graphics/2d-graphics.html

Android Developers. 2013p. Read 17.12.2013.
http://developer.android.com/design/building-blocks/switches.html

Android Developers. 2013q. Read 17.12.2013.
http://developer.android.com/design/building-blocks/seek-bars.html

Android Developers. 2013r. Read 17.12.2013.
http://developer.android.com/design/building-blocks/progress.html

Android Developers. 2013s. Read 17.12.2013.
http://developer.android.com/guide/topics/resources/providing-resources.html

Android Developers. 2013t. Read 31.12.2013.
http://developer.android.com/distribute/googleplay/publish/preparing.html

Android Developers. 2014. Read 15.1.2014.
http://developer.android.com/design/patterns/pure-android.html

Appcelerator Wiki. Custom Fonts. 2013. Read 14.12.2013.
https://wiki.appcelerator.org/display/guides/Custom+Fonts

Cambridge Dictionaries Online. 2014. Read 7.1.2014
http://dictionary.cambridge.org/dictionary/british/workflow

Elgin, Ben. Google Buys Android for Its Mobile Arsenal. Bloomberg Businessweek. 16.8.2005. Read 12.12.2013. http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal

Google Developers. 2013. Read 31.12.2013.
https://developers.google.com/speed/webp/

Google I/O 2013 - WebP: Deploying Faster, Smaller, and More Beautiful. Google Developers 2013. Watched 31.12.2013. http://youtu.be/pS8udLMOOaE

PCMag.com, 2014. Read 22.2.2014.
http://www.pcmag.com/encyclopedia/term/63822/android-versions

IMAGES

Android.com. 2013. [Website]. Visited 12.12.2013.
http://www.android.com/versions/kit-kat-4-4/

Android Developers. 2013d. Visited 12.12.2013.
http://developer.android.com/design/style/color.html

Android Developers. 2013c. Visited 22.2.2014.
http://developer.android.com/design/style/themes.html

Android Developers 2013g. Visited 14.12.2013.
http://developer.android.com/guide/practices/screens_support.html

Android Developers 2013j. Visited 14.12.2013.
http://developer.android.com/design/style/metrics-grids.html

Android Developers. 2013i. Visited 16.12.2013.
http://developer.android.com/design/style/iconography.html

Android Developers. 2014. Visited 15.1.2014.
http://developer.android.com/design/patterns/pure-android.html

Google Play. Runkeeper. 2014. Visited 22.2.2014.
https://play.google.com/store/apps/details?id=com.fitnesskeeper.runkeeper.pro

iTunes. Runkeeper. 2014. Visited 22.2.2014.
https://itunes.apple.com/us/app/id300235330

APPENDICES

Appendix 1. Benchmark of score keeping apps