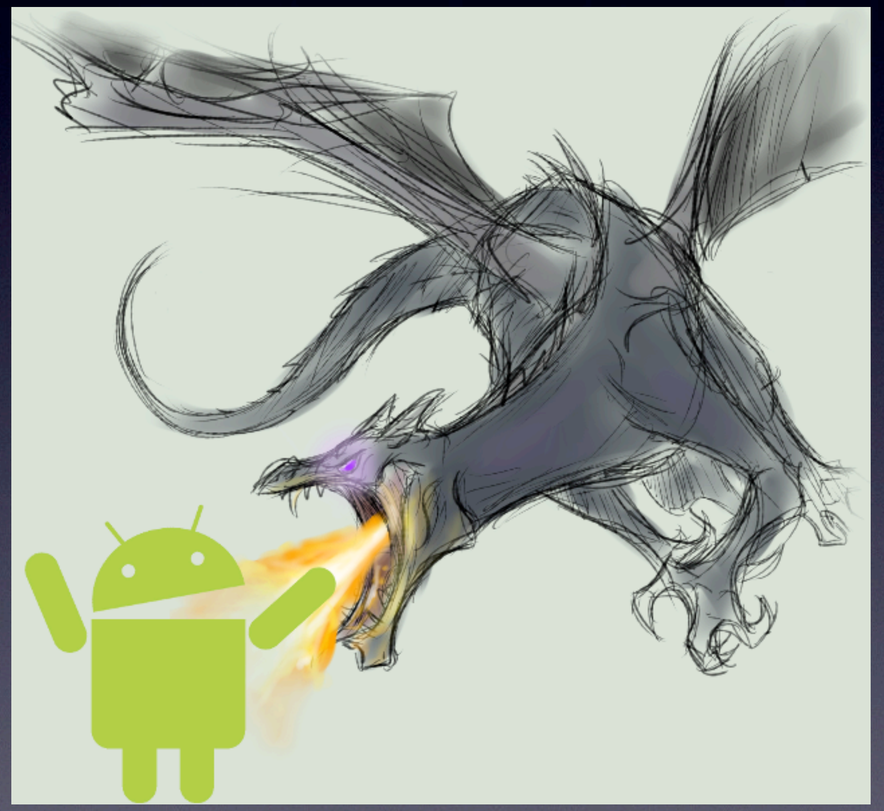
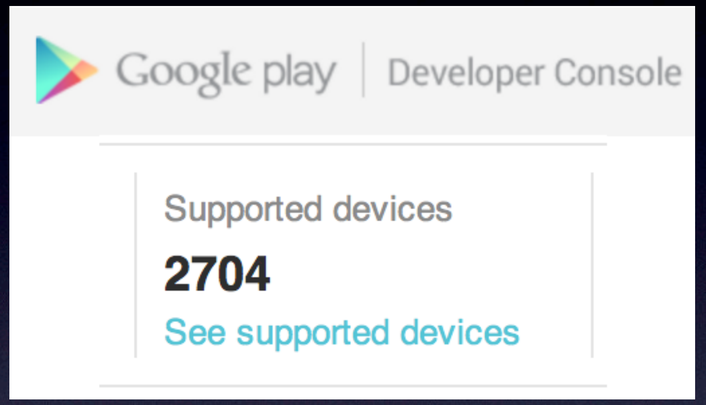
Beating Android Fragmentation

Here be Dragons



Why it's Scary



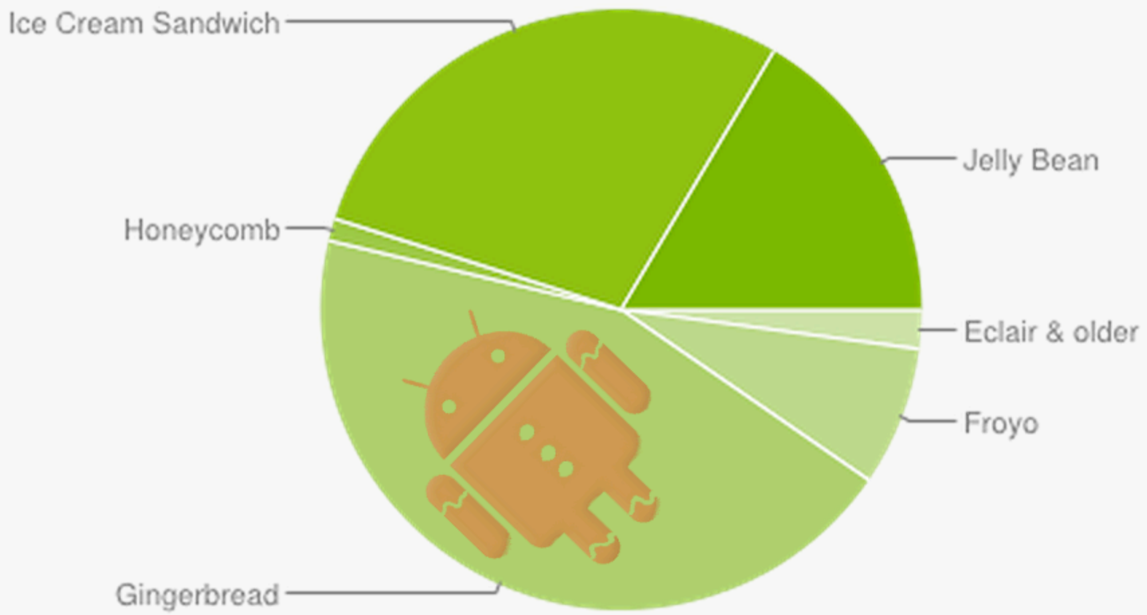
That's a big number.

Know Thy Enemy

- Android OS Versions
- Device Display Variations
- Hardware Configurations

The Landscape

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	1.9%
2.2	Froyo	8	7.5%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	43.9%
3.1	Honeycomb	12	0.3%
3.2		13	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	28.6%
4.1	Jelly Bean	16	14.9%
4.2		17	1.6%



Data collected during a 14-day period ending on March 4, 2013

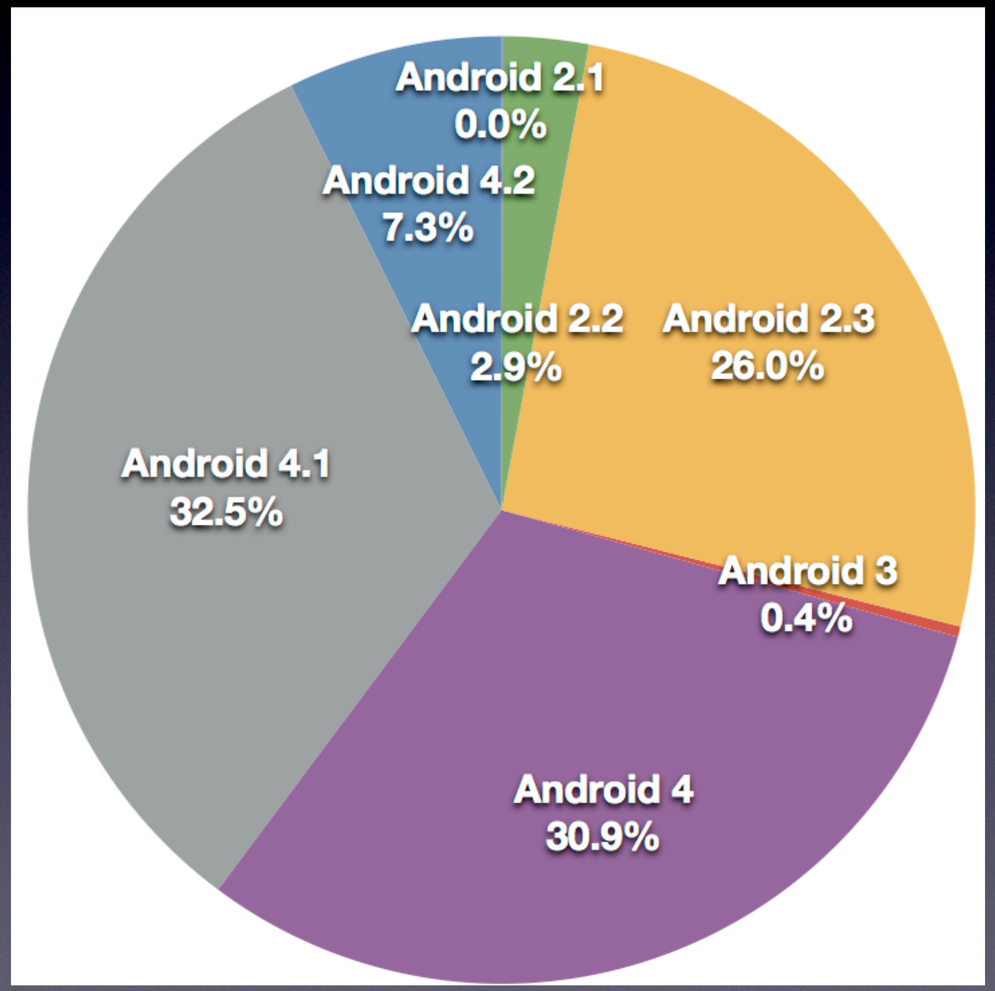
The Landscape

- Over 40% of devices in the wild are running Gingerbread! (Android version 2.3.x, API level 8)
- "Modern" Android OS versions (Honeycomb and higher) are < 50% of the market.
- Thus, in order to reach all potential users in your market, you must deal with OS fragmentation.

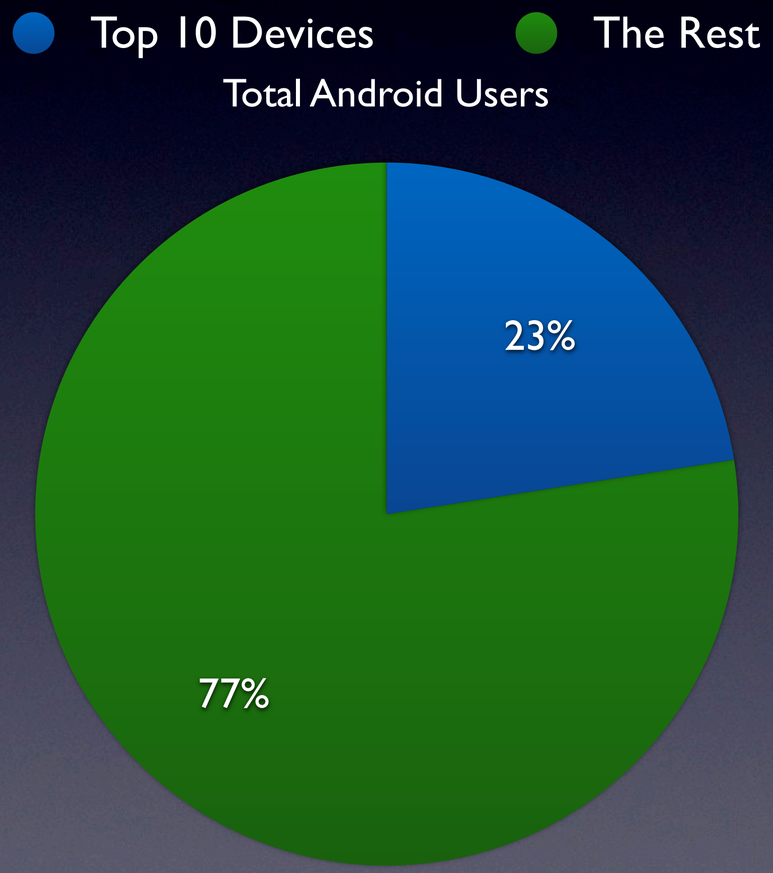
Basic Defense

- Set `targetSdk` to the current highest API level.
- Use helper classes to encapsulate OS version checking logic. Reflection will not work. But you can check `Build.VERSION.SdkInt`
- Optionally set `minSdk` to the lower bound of Android API level your app supports.
- It is possible to have multiple APKs in Google Play that support different Android API levels.

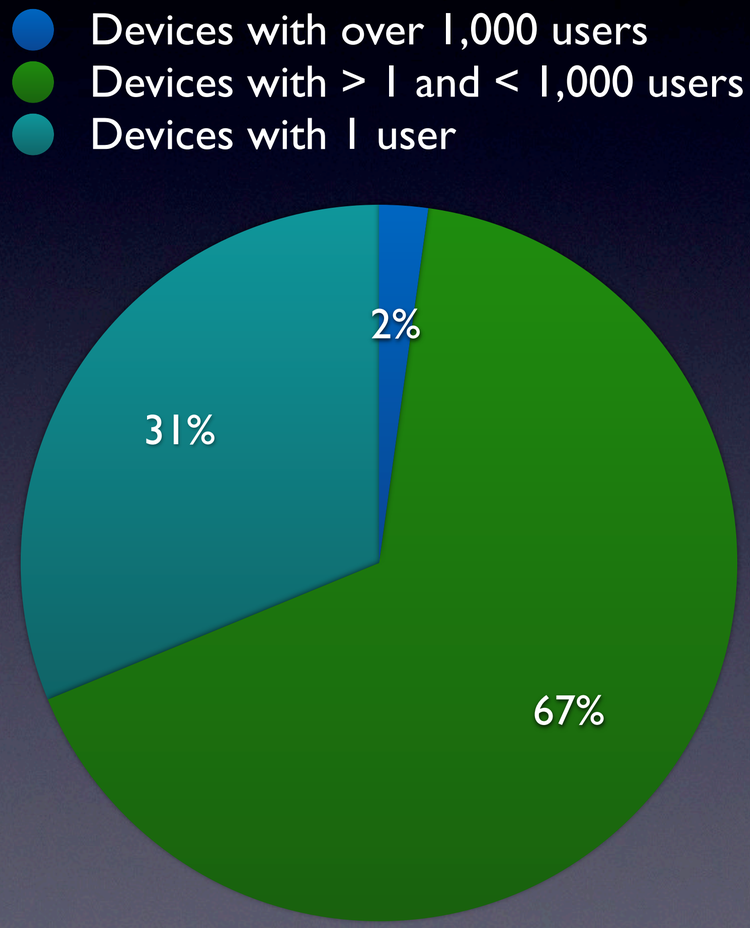
Rdio's Android Landscape



Rdio's Long Tail



Rdio Device Distribution



Rdio's Long-tail (is long)

- 2866 unique devices per month (as determined by Build.Product, Build.Model)
- Top 10 devices represent 23% of all users
- 255 devices represent 85% of all users
- 894 devices had exactly (one) user

How Did We Get This Data?

- Not from Google Play.
- Even a simple means of identifying the device types and Android versions running your app/service will prove invaluable.
- Embedding the `Build.Model`, `Build.Product`, `Build.VERSION.SdkInt` into the User-Agent header, for example.
- A service that pings an API endpoint every 24 hours. (The example app has an implementation of this method).

Be Cautious

- The product and model strings can be too granular sometimes.
- Many devices with the same street name have multiple model and product strings.
- The Samsung Galaxy S III has at least 5 unique model strings.

Pick Your Battles

- Use data to drive your development focus.
- Make smart decisions on where you cut off support.
  - It is _possible_ to support all the way down to Android 1.6, but is it worth it?
- Focus on device-specific issues that a lot of your users have.

Device Display Variations

Device Display Variations

- Display dimensions
- Display density (dpi)
- Display quality

- Don’t try to collect them all.
- Use SDK tools to help during design

Tools

- Android support library: Adds support for certain features (such as Fragments) all the way back to Donut (1.6).
- ActionBar Sherlock: An open source library that adds support for ActionBar to all versions of Android. http://actionbarsherlock.com
- android-ui-utils: https://code.google.com/p/android-ui-utils/

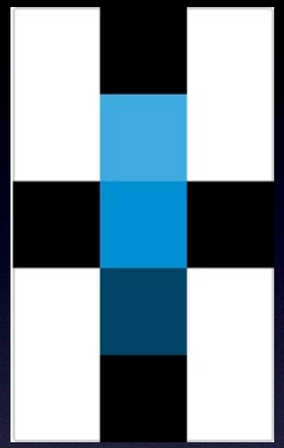
Use Fragments!

- Activities are heavy, Fragments are light.
- Allow for more flexible UI design and runtime configuration.

Think in Points, not Pixels

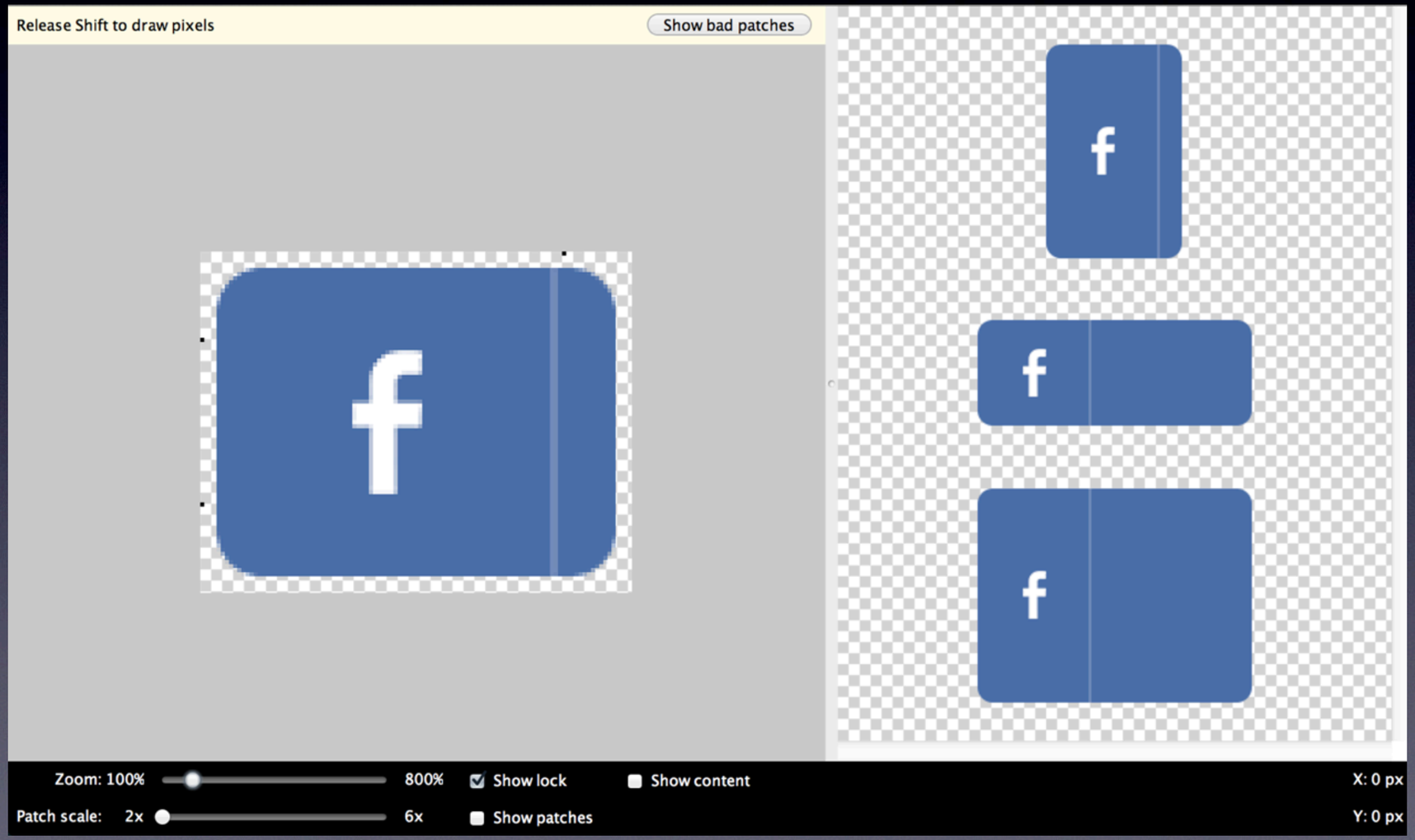
- You can't make a pixel perfect implementation, so don't try.
- A point is a logical representation of a pixel, based on the actual size of a pixel on a 160dpi display.
- When designing start with the baseline (160dpi), and extrapolate upwards. At this density 1pt == 1px (roughly).

Think in Points, not Pixels



- 9-patch is your friend. If your designers aren't using 9-patch, make them. There is an excellent 9-patch tool that ships with the Android SDK (draw9patch).
- 9-patch images are specially formatted PNGs that define what areas can and cannot be stretched.

draw9patch


Release Shift to draw pixels
Show bad patches
Zoom: 100%
800%
Show lock
Show content
X: 0 px
Patch scale:
2x
6x
Show patches
Y: 0 px

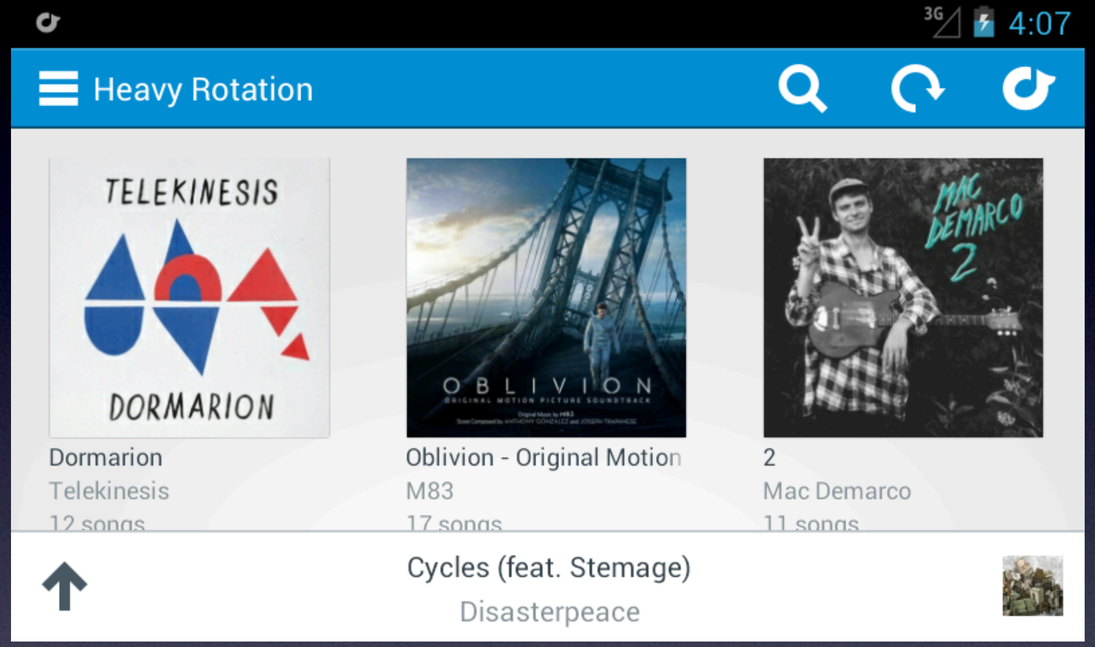
Design for Android

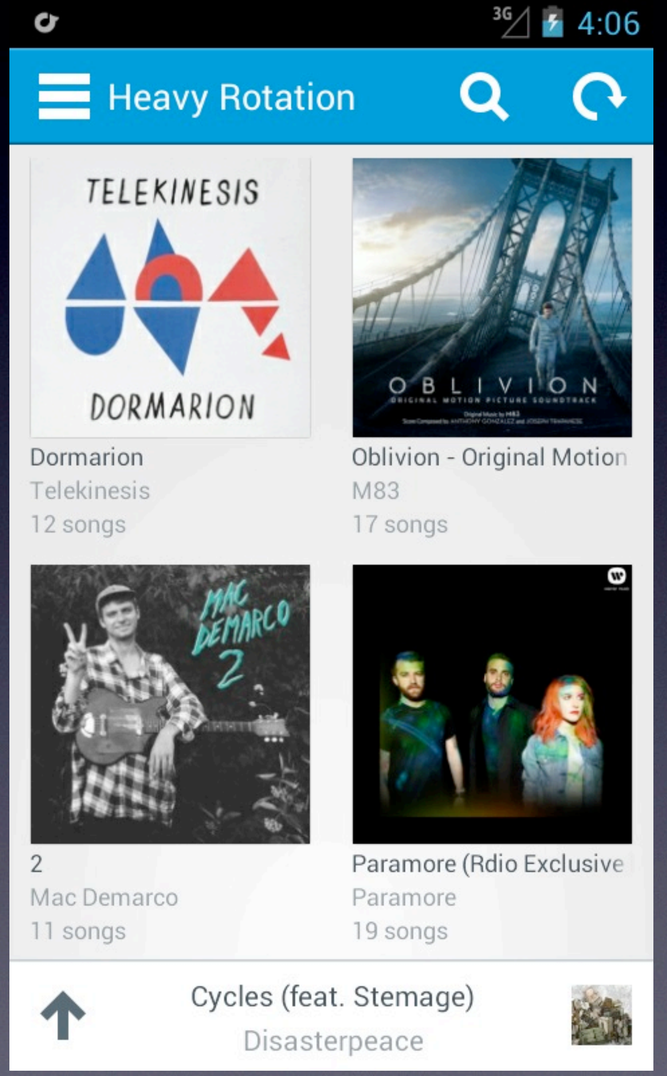
- Don’t just take a design made for iOS and shoehorn it into Android.
- Use Android UI idioms: ActionBar, long-press, support landscape orientation whenever possible, etc.
- RTFM! Google provides some excellent resources on how to design and develop apps to support different display types.

Level-up Your Layout!

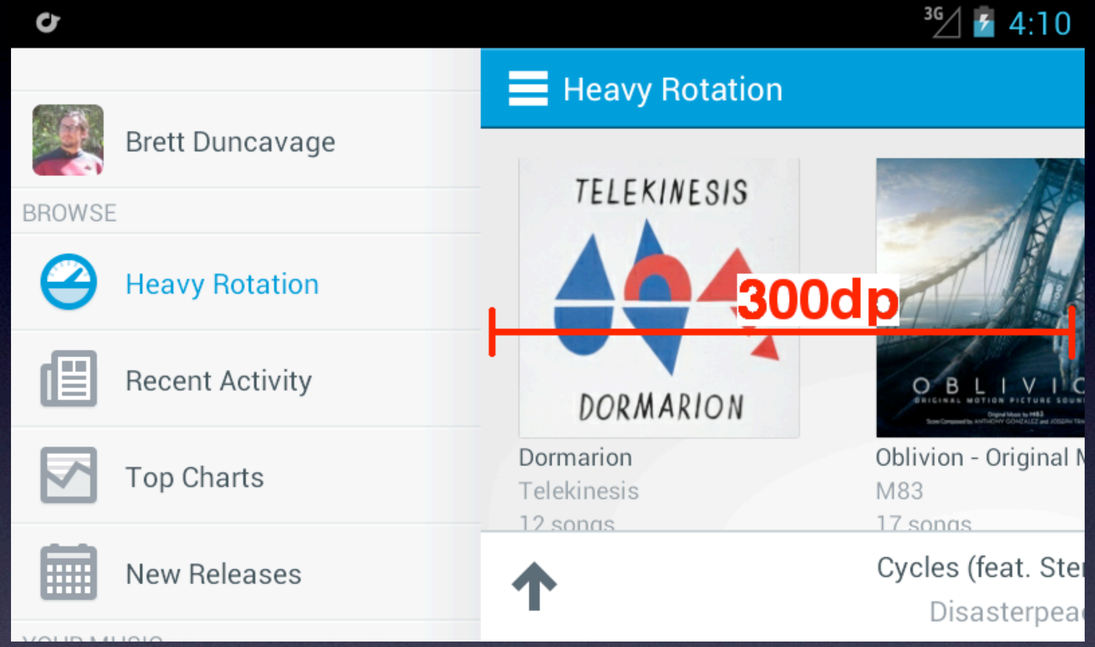
- Two easy, and often overlooked, resources are dimens.xml, and integers.xml.
- Use Android's resource directory structure to include multiple versions (as with any resource). Layout-land, Layout-large, etc.
- For example, define number of columns in a grid view in integers.xml so different values will be used depending on screen size and orientation (in lieu of using auto_fit).

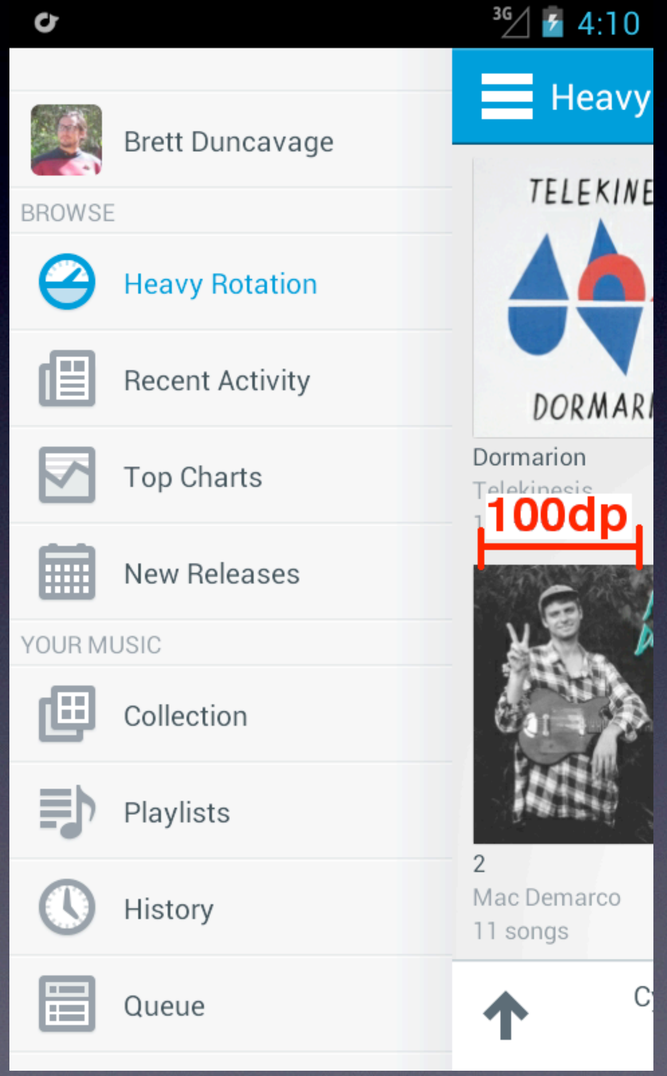
integers.xml example



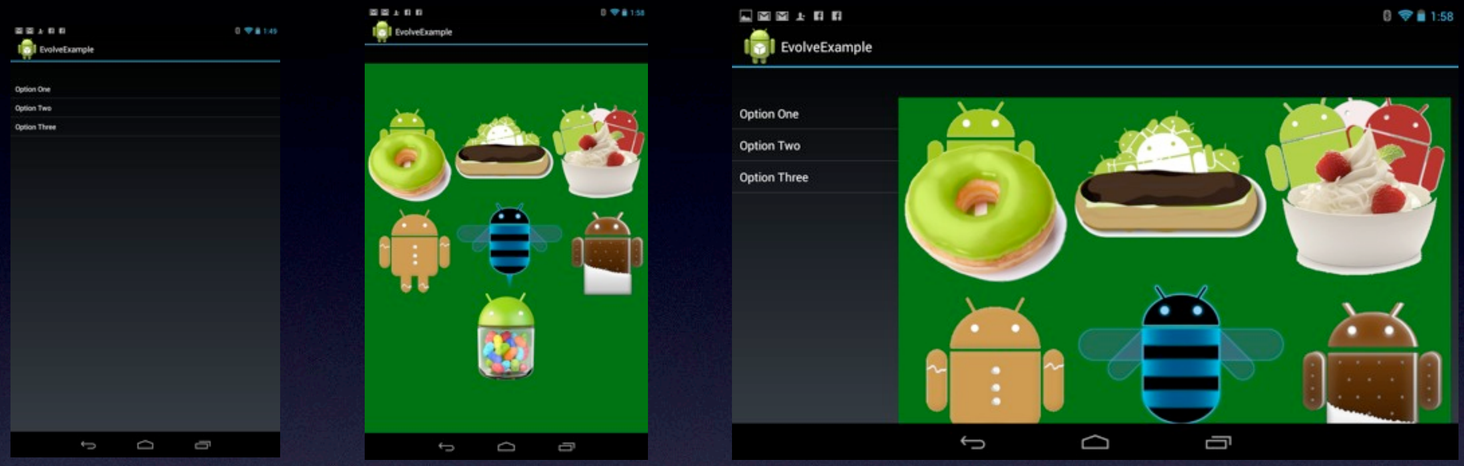


dimens.xml example





Level-up Your Layout!



- Use layout aliases: Define your different layouts (single_pane.xml, dual_pane.xml, etc) in Resource/Layout, and provide mappings in the size-qualified values directories (Resource/Values-large, etc).

- This saves you from having duplicate layouts residing in different Layout directories (i.e. Layout-large/ and Layout-sw600dp/).

Don’t Forget About Focused State

- When designing buttons, or any other clickable/tappable widget, don’t forget to supply a focused state.
- Without the focused state, the pointing device becomes useless as the user will have no visual feedback to indicate which control has focus and will be activated on click.
- `<item android:state_focused="true" android:drawable="@drawable/button_focused" />`

Hardware Fragmentation

- May not be an issue depending on your application's domain.
- Use your manifest to inform Google Play of your hardware requirements (requires camera, etc).
- Use available APIs to determine fallback action at runtime if optional peripheral is not available. Such as `SensorManager`.

External Storage

- Not always external or removable
- Never use a hardcoded path! /sdcard is wrong, wrong, wrong.
- Environment.ExternalDirectory can return a path to internal storage.
- Store vital application data on internal storage (sqlite database, etc).

Media Framework

- OEMs use many different vendors for their media hardware needs.
- OEMs don’t always or can’t use the default Android MediaPlayer implementation.
- This means the MediaPlayer can behave in insane ways.

How Rdio Mitigates Device Issues

- Trial and Error (and sometimes luck).
- Beta users group.
- Send one-off test builds to affected users.
- Remote device laboratories such as DeviceAnywhere and Perfecto Mobile.

Conclusion

- You have no choice, you will encounter fragmentation.
- Be pragmatic, choose your battles, and use usage data to backup your decisions on where to focus resources.
- Design for Android! Android is not iOS, Android users are not iOS users, make an app that Android users will like.

Thanks!

- There are 4 of us from Rdio at Evolve (Anthony, Eric, Patrick, and myself), if you see us, say hi!
- Brett Duncavage, Rdio Android Lead
- @xforward, http://brett.duncavage.org
- Source for example available here: https://github.com/bduncavage/evolve2013

Biblography

- Android OS Version Distribution: http://developer.android.com/about/dashboards/index.html
- Android Designing for Multiple Screens: http://developer.android.com/training/multiscreen/index.html
- ActionBar Sherlock: http://actionbarsherlock.com
- android-ui-utils: https://code.google.com/p/android-ui-utils/