Supporting Multiple Screen sizes

DT228/3

Dr. Susan McKeever

Designing for multiple Android devices

When you code up an Android app..

Android Phones have different screen size
And
different screen densities..

And Android tablets too…

How can you get your app to look good/the same on all?

Screen size/ screen density

Screen size

Actual physical screen size, measured as the screen's diagonal.

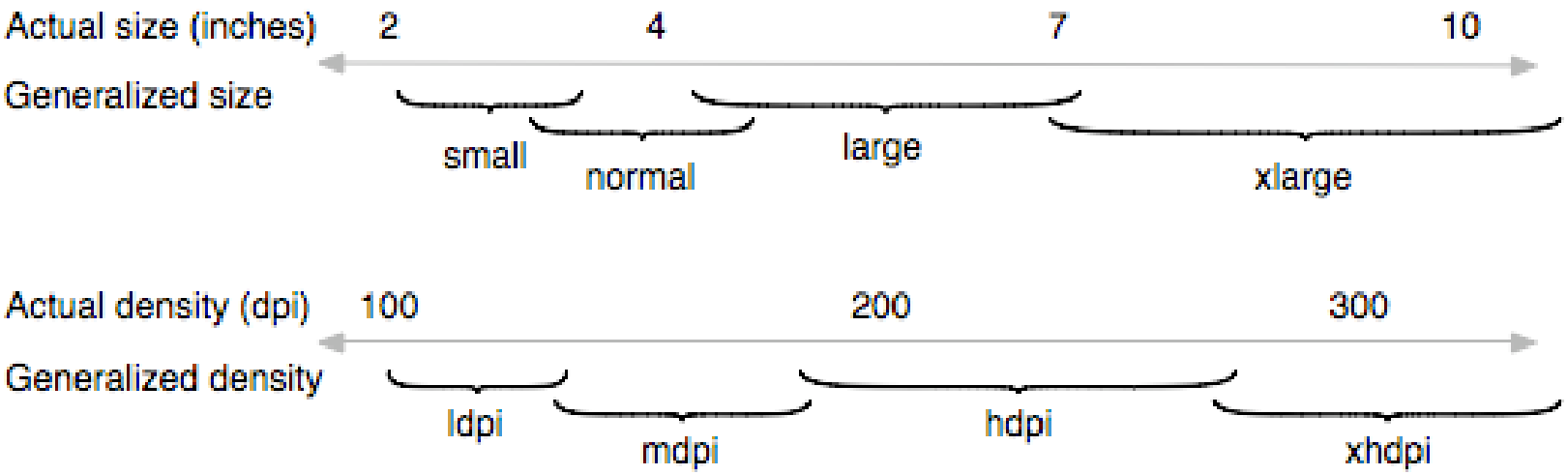
In Android: small, normal, large, and extra large.

Screen density

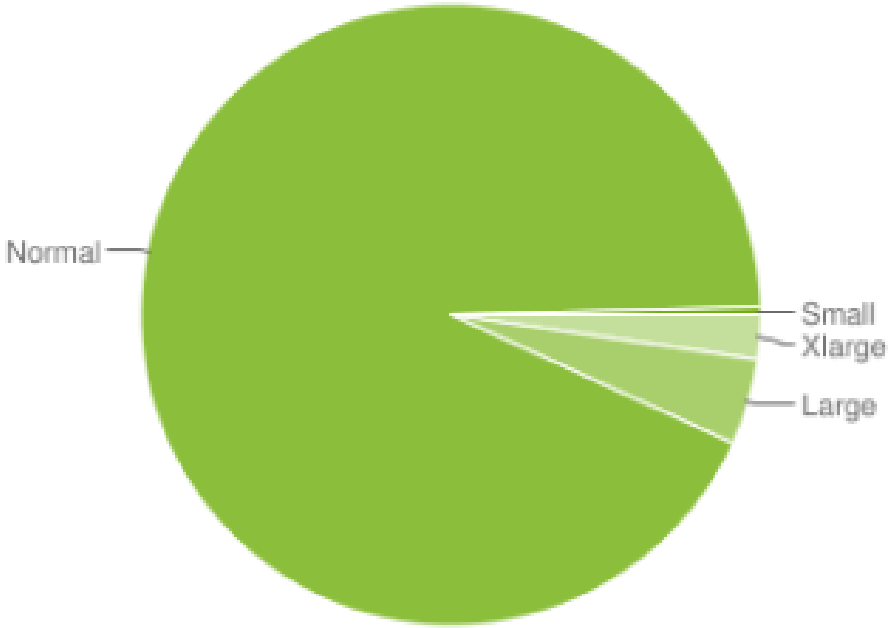
The quantity of pixels within a physical area of the screen; usually referred to as dpi (dots per inch).

In Android : low (ldpi), medium (mdpi), high (hdpi), and extra high (xhdpi).

Screen size/ screen density
As classified by Android system



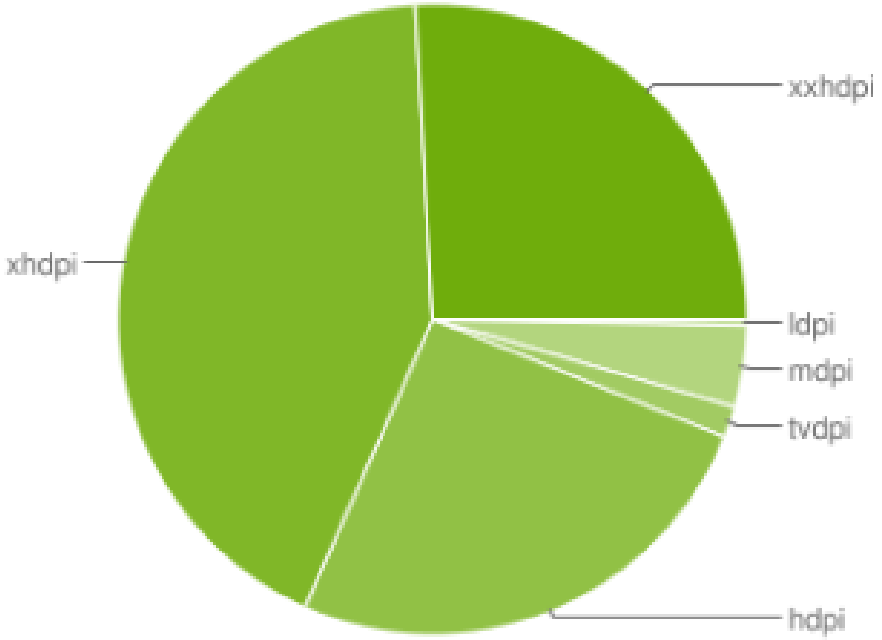
What's types of Android devices are out there?



Screen Sizes - As at Oct 26th 2018 – previous 7days, devices accessing Android app market – see:

http://developer.android.com/resources/dashboard/screens.html

What's types of Android devices are out there?



Screen Densities- As at Oct 26th 2018 – previous 7days, devices accessing Android app market –

What's devices are out there?

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.3%					0.1%	0.4%
Normal		0.7%	0.3%	24.7%	41.9%	25.2%	92.8%
Large		2.0%	1.3%	0.4%	0.3%	0.5%	4.5%
Xlarge		1.5%		0.5%	0.3%		2.3%
Total	0.3%	4.2%	1.6%	25.6%	42.5%	25.8%	

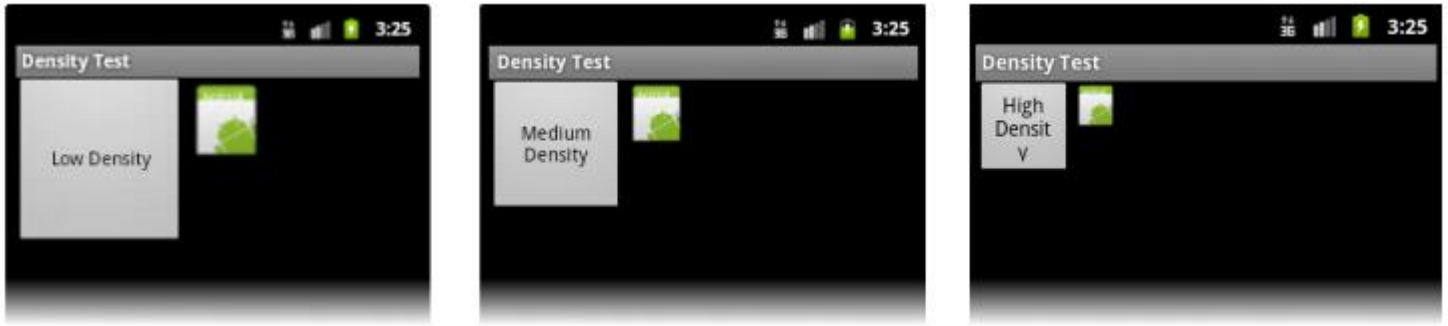
So which size/density one should we aim at?

Will change over time as device usage/popularity changes..
Which direction do you think?

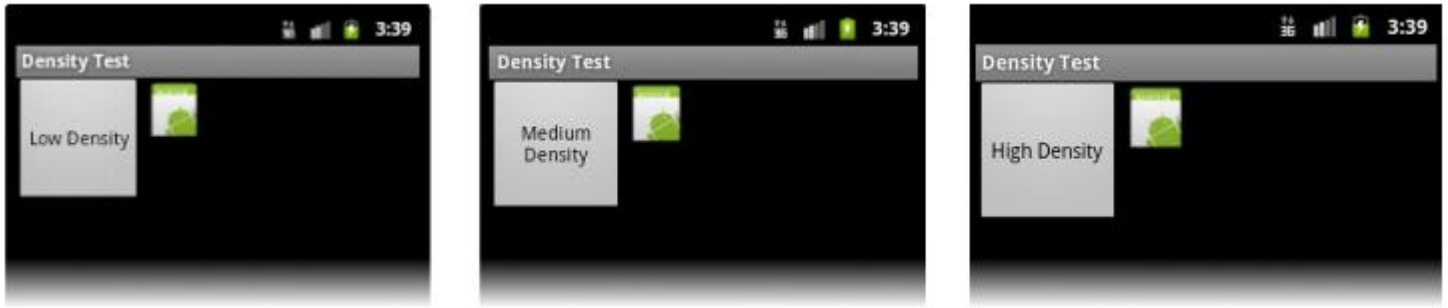
Versions?

2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%

Different screen densities..



low, medium, and high density screens… showing same app with no consideration of densities.



low, medium, and high density screens… but **WITH good** support for densities..

Tactics for coping with different screen sizes and densities

- (1) Use Screen compatibility mode
  - (But different behaviour depending on Android version)
- (2) Specify in your app which screen sizes/densities are supported
  - (overrides Screen compatability mode)
- (3) Provide multiple screen layouts
  - to support multiple screen sizes
- (4) Provide multiple image densities
  - to support multiple screen densities

Tactics for coping with different screen sizes and densities

- (5) Use Best practices
  - See various practices
- (6) Use fragments (Android 3 onwards)

Generally: Not an exact science.. ! May be a combination of some or all of the above…

Each Android version brings changes.

Biggest drawback of Android

Tactics for catering for muliple devices – (1) screen compatibility mode

Get Android to does the work of resizing your screen if you run in screen compatibility mode (but can cause pixelation/blurring) – behaviour depends on Android API version

Version 1.6 – 3.2

The system draws the application's User interface in a "postage stamp" window.

Black border that fills the remaining area of the screen.

To DISABLE screen compatability mode: SET android:minSdkVersion or android:targetSdkVersion to "4" or higher, or set android:resizeable to "true".

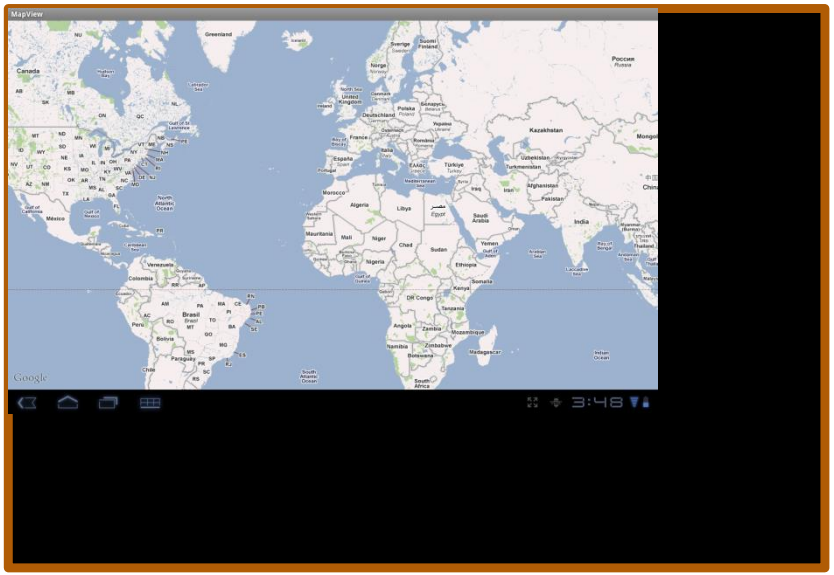
Version 2 (Android 3.2 and greater)

The system draws the application's layout the same as it would on a normal size handset (approximately emulating a 320dp x 480dp screen), then scales it up to fill the screen.

Can cause blurring and pixelation

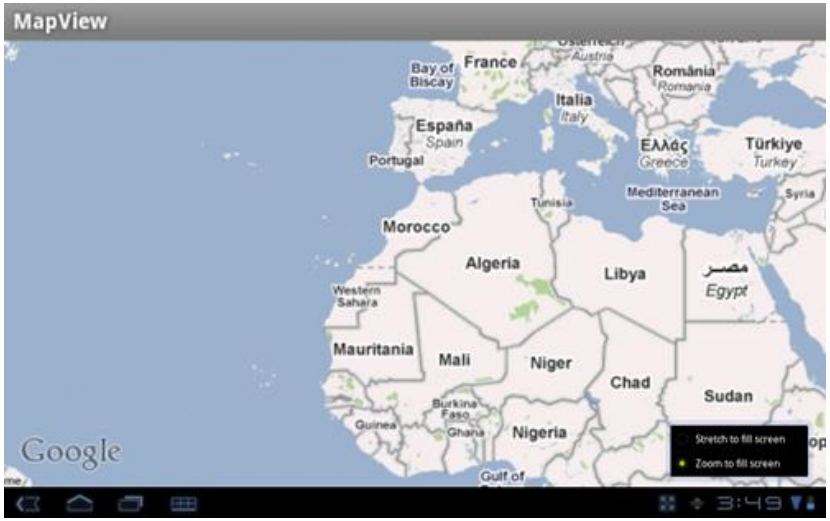
Tactics for catering for muliple devices – (1) screen compatability mode

Version 1.6 – 3.2



Layout is limited to one size and rest of screen is black

Version 2 (Android 3.2 and greater)



Layout is zoomed to fill screen

Tactics for catering for multiple devices – (2) Specify Screens supported

Specify what screen sizes your application support (if you don't, screen compatability mode will take over)

Manifest file

<supports-screens> tag

```
<supports-screens android:resizeable=["true"| "false"]
        android:smallScreens=["true" | "false"]
        android:normalScreens=["true" | "false"]
        android:largeScreens=["true" | "false"]
        android:xlargeScreens=["true" | "false"]
        android:anyDensity=["true" | "false"]
        android:requiresSmallestWidthDp="integer"
        android:compatibleWidthLimitDp="integer"
        android:largestWidthLimitDp="integer"/>
```

Tactics for catering for multiple devices –
(3) Provide multiple layout files

Provide Different Layouts (.XML files) for different screen sizes

- e.g. On a larger screen, might want to adjust the position and size of some elements to take advantage of the additional screen space,
- On a smaller screen, you might need to adjust sizes so that everything can fit on the screen.

```
res/layout/my_layout.xml             // layout for normal screen size ("default")
res/layout-small/my_layout.xml       // layout for small screen size
res/layout-large/my_layout.xml       // layout for large screen size
res/layout-xlarge/my_layout.xml      // layout for extra large screen size
res/layout-xlarge-land/my_layout.xml // layout for extra large in landscape
```

NOTE ! Beginning with Android 3.2 (API level 13), the above size groups are deprecated and you should instead use the sw<N>dp

e.g.. res/layout-sw600dp/main_activity.xml # For tablets - layout, small width = 600 dp

(3) Multiple layouts (Android 3.2 onwards)

SmallestWidth (screen) SW<n>DP
Size of screen based on smallest dimension

SW600dp = smallest screen width of 600 dots per inch..

Available screen width w<n>cp

W1200dp = available screen width = 1200 dpi. Minimum width needed for a layout

Available screen height h<n>dp
Minimum screen height in dp units at which the resources should be used—defined by the <N> value.

```
res/layout/main_activity.xml          # For handsets (smaller
than 600dp available width)
res/layout-w600dp/main_activity.xml   # Multi-pane (any screen
with 600dp available width or more)
```

Tactics for catering for multiple devices – (3) Provide multiple layout files

E.g. An application that is built to run on mobile phone and on tablet:
has two layouts created –
1 for Mobile handsets and 1 for tablets

Stored in:

```
res/layout/main_activity.xml            # For handsets
res/layout-sw600dp/main_activity.xml    # For tablets
```

Tactics for catering for multiple devices
(4) Provide multiple density images versions

To Ensure images look their best, include alternative versions at different resolutions for different screen densities.

Use various drawable directories

```
res/drawable-mdpi/sampleImage.png  // bitmap for medium density
res/drawable-hdpi/sampleImage.png  // bitmap for high density
res/drawable-xhdpi/sampleImage.png // bitmap for extra high density
```

Q: If just using one set of images where do they get stored?

Note: launcher icons go in the mipmaps directory

Tactics for catering for multiple devices

(5) Use Best practices

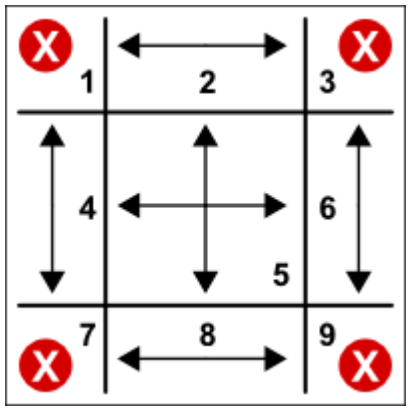
- ✓ "Wrap content"/ "fill parent" when specify XML for layouts
- ✓ No hardcoded pixel values
- ✓ Absolute layout X
  Relative layout – good!
- ✓ Always use **NinePatch** images when designing bitmaps that will scaled on different size screens – planned scaling for images – see overleaf;
- ✓ Test test test.. on multiple screens

Tactics for catering for multiple devices

(5) Use Best practices

NinePatch images - where image bitmap is divided into 9 (or more) patches – where one or more of the patches won't need to scale

"X" portions as not need to expand.. Scaling aplied to other parts



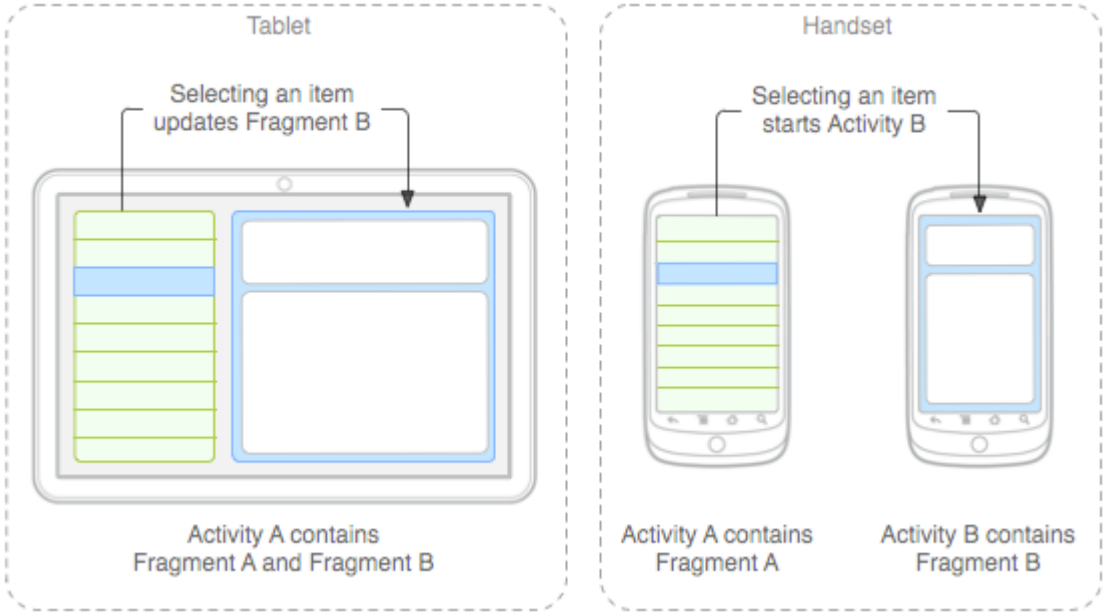
Draw9patch supplied as part of the SDK

Tactics for catering for multiple devices

(6) Fragments

As covered Android 3 onwards…

Reusable bits (i.e. fragments) of user interface.. Good for different screen sizes



Fragments

Implement

statically (hard coded <fragment > tag or

dynamically (actually adding/ removing fragments at run time

Summary

Different screen sizes and densities

Range is expanding all the time

Need to decide which screens your app will support

Then determine which practices to use

Android support continually developing in newer versions