References

Supporting Multiple Screens,
http://developer.android.com/guide/practices/screens_support.html
# Screen Size/Density

<table>
<thead>
<tr>
<th></th>
<th>Low density (120), ldpi</th>
<th>Medium density (160), mdpi</th>
<th>High density (240), hdpi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Small screen</strong></td>
<td>QVGA (240x320), 2.6&quot;-3.0&quot; diagonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Normal screen</strong></td>
<td>WQVGA (240x400), 3.2&quot;-3.5&quot; diagonal</td>
<td>HVGA (320x480), 3.0&quot;-3.5&quot; diagonal</td>
<td>WVGA (480x800), 3.3&quot;-4.0&quot; diagonal</td>
</tr>
<tr>
<td></td>
<td>FWQVGA (240x432), 3.5&quot;-3.8&quot; diagonal</td>
<td></td>
<td>FWVGA (480x854), 3.5&quot;-4.0&quot; diagonal</td>
</tr>
<tr>
<td><strong>Large screen</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WVGA (480x800), 4.8&quot;-5.5&quot; diagonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FWVGA (480x854), 5.0&quot;-5.8&quot; diagonal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Test Your Application on Multiple Screens

QVGA (240x320, low density, small screen)
HVGA (320x480, medium density, normal screen)
WVGA800 (480x800, high density, normal screen)
WVGA854 (480x854 high density, normal screen)
Android 2.0
WQVGA400 (240x400, low density, normal screen)
WQVGA432 (240x432, low density, normal screen)
Getting correct size screen

Create an AVD for the target screen size & density

Tell the emulator your monitors density

```
emulator -avd normal -scale 96dpi
```
Difference Emulator Sizes

WVGA854

default emulator size

Corrected for monitor density
Supporting Multiple Screens

Resource Files for different size/density

Specify the supported Screen
Specify the supported Screen

android:smallScreens
android:normalScreens
android:largeScreens
android:anyDensity

<manifest xmlns:android="http://schemas.android.com/apk/res/android">
<br /> <supports-screens
 android:largeScreens="true"
 android:normalScreens="true"
 android:smallScreens="true"
 android:resizable="true"
 android:anyDensity="true" />
</manifest>
## Resource Files for different size/density

<table>
<thead>
<tr>
<th>Screen characteristic</th>
<th>Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>small</td>
</tr>
<tr>
<td></td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>large</td>
</tr>
<tr>
<td>Density</td>
<td>ldpi</td>
</tr>
<tr>
<td></td>
<td>mdpi</td>
</tr>
<tr>
<td></td>
<td>hdpi</td>
</tr>
<tr>
<td></td>
<td>nodpi</td>
</tr>
<tr>
<td>Aspect ratio</td>
<td>long</td>
</tr>
<tr>
<td></td>
<td>notlong</td>
</tr>
<tr>
<td>Platform version</td>
<td>v&lt;api-level&gt;</td>
</tr>
</tbody>
</table>
res directory

res/layout/my_layout.xml   // layout for normal screen size
res/layout-small/my_layout.xml   // layout for small screen size
res/layout-large/my_layout.xml   // layout for large screen size
res/layout-large-land/my_layout.xml // layout for large screen size in landscape mode

res/drawable-ldpi/my_icon.png  // icon image for low density
res/drawable-mdpi/dpi/my_icon.png // icon for medium density
res/drawable-hdpi/my_icon.png   // icon image for high density

res/drawable-nodpi/composite.xml  // density independent resource
Runtime support

Pre-scaling of resources

Load size- or density-specific resources

If not available scale resource to needed size
Runtime support

Auto-scaling of pixel dimensions and coordinates

If application does not support different screen densities

Android auto-scales
  absolute pixel coordinates,
  pixel dimension values, and
  pixel math
Runtime support

Compatibility-mode display on larger screen-sizes

If current screen's size is larger than your application supports

Application is shown in supported size surrounded by black
Density independence

Pre-scaling of drawable resources

Auto-scaling of device-independent pixel (dip) values

Auto-scaling of absolute pixel values used in the application when android:anyDensity="false"
Best practices for Screen Independence

Prefer wrap_content, fill_parent and the dip unit to px in XML layout files
Avoid AbsoluteLayout
Do not use hard coded pixel values in your code
Use density and/or resolution specific resources
Prefer `wrap_content` etc

```
wrap_content, fill_parent & dip
layout_width="100dip"
```

Font sizes

```
sp (scale-independent pixel)
  Allows the user to scale the text
```

```
dip
  User can't scale the text
```
AbsoluteLayout

deprecated in Android 1.5

Use FrameLayout
set layout_margin of children
Don't hard-coded pixel values in your code

But you may need to know the actual pixel scroll gesture -
  triggered when move by at least 16 pixels

HVGA
  16 pixels / 160 dpi = 1/10th of an inch

high (240) density display
  16 pixels / 240 dpi = 1/15th of an inch
getContext().getResources().getDisplayMetrics().density

scaling factor for the Density Independent Pixel

160dpi screen - 1.0
120dpi screen - 0.75
240dpi screen - 1.5

private static final float GESTURE_THRESHOLD_DIP = 16.0f;

// Convert the dips to pixels
final float scale = getContext().getResources().getDisplayMetrics().density;
mGestureThreshold = (int) (GESTURE_THRESHOLD_DIP * scale + 0.5f);
Use pre-scaled configuration value

android.view.ViewConfiguration

ViewConfiguration.get(aContext).getScaledTouchSlop()

g getScaledXXX()
  returns values in pixels that will display properly
Pre-scaling, auto-scaling

resources
pre-scaled before displayed
value of android:anyDensity does not affect pre-scaling

android:anyDensity=true
resources are pre-scaled
original sizes are reported
If you modify the resource odd things may happen

nodpi resources are not scaled

BitmapFactory.Options
gives control over pre-scaling
Pre-scaling, auto-scaling

auto-scaling
Done when creating bitmaps in memory and drawing on them

Drawing on the larger bitmap is different than scaling after drawing
Pre-scaling versus auto-scaling

Effects of loading low (120 dpi) medium (160 dpi) & high (240 dpi) bitmaps on a medium screen