About This Article

This article provides information about how to make Android application for multiple screen resolutions. Android application developers can use this article as reference to implement the application for multiple screens available from different vendors.

Scope:

This article is intended for Android developers wishing to develop mobile applications. It assumes basic knowledge of Android and Java programming languages.

To find out more about Android, please refer to the Knowledge Base under Samsung Developers.
http://developer.samsung.com/android

Introduction

Android devices come in a variety of screen sizes and resolutions. Android dynamically scales and resizes application graphics to fit these different from factors, however this scaling isn’t perfect and may hurt user experience. A smartphone app that is run on a Tablet, for example, “blows up” the graphics in order to fill the larger screen, but if resolution of those graphics does not take different screen sizes into account, the result will not be appealing.

Developers can apply the following methods to maximize the user experience on a variety of devices. This document explains multi-screen handing on Android platform along with the overview of system supported utilities.

Default Things on App Installation

By default Android runs applications in "compatibility mode", scaling everything based on the actual screen size. For a 20px square PNG file on a device with high-density screen, it may scale PNG file up to 30px during image rendering making the image blurry.

Android blocks applications from running on devices with too small of a screen, hence QVGA devices will not downloads certain apps that require a larger screen. To avoid this, compile the application with the higher resolution graphics so that Android can fill the screen size properly, without relying on "compatibility mode".

Handing Screen Density

You can develop your application to maintains the physical size of the interface by using fill_parent and wrap_contentvalues for android:layout_width and android:layout_height. This parameter does not specify size but adapts to the space available.

```
        ...
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        ...
```

Bitmap scaling can result in blurry images. To prevent this, provide higher-resolution bitmaps for high-density screens and the system will use those instead of resizing the bitmap designed for medium-density screens.

Following are ways the Android system helps to achieve density independence:

1. Android considers dp [density-independent pixels] units as appropriate for the current screen density. These map 1:1 to pixels for a 160dpi screen (e.g., a standard HVGA Android device) and

scales from there.

The conversion rule follows here is px=dp*(dpi/160). Now apply this as conversion rule to a 240dpi device (e.g., a phone-sized WVGA device), the pixel ratio is 3:2 or 1.5. It assumes that 60dip = at 160dpi of screen, so what is the pixel cover at the 240dpi screen:

px = 60*(240/160);

px = 90;

As per the formula it’s 90 px at 240dpi screen.

So at 60dpi at 160dpi screen is equivalent to 90px at 240dpi screen. The advantage to the user is that the actual dimensions stay the same, there is no visible difference between 60dip at 160dpi and 60dip at 240dpi.

2. Android also accounts for scaled pixels (sp), which are scaled based on the user's preference of font size (FONT_SCALE value in System. Settings).
3. Android scales the drawable resources to the suitable size, if alternative resources are not available, and it solely depends on the current screen density.

Density Based Set:

To set labels for different layouts, dimensions or different screen density resources, use the following resource directories:

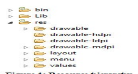


Figure 1: Resource hierarchy

For example:

- res/drawable-hdpi: contains bitmap resources used for high-density
- res/drawable-mdpi: contains bitmap resources used for medium-density
- res/drawable-ldpi: contains bitmap resources used for low-density

For layouts:

- res/layout/my_layout.xml: contains layout for normal screen size ("default")
- res/layout/my_layout.xml: contains layout for normal screen size ("default")
- res/layout-small/my_layout.xml: contains layout for small screen size
- res/layout-large/my_layout.xml: contains layout for large screen size

Supporting Multiple Screen Sizes

To set up support for multiple device sizes in Android, add the <support-screens> element into theAndroidManifest.xml file. This field specifies which screen size support and which do not. The developers who don't want to handle automatic "compatibility mode" can implement this attribute into the manifest file. Include <support-screens> attributes into the AndroidManifest.xml as shown the following code sample:

```
        <?xml version="1.0" encoding="utf-8"?>
        <manifest xmlns:android="http://schemas.android.com/apk/res/android"
                package="com.expense.app"
                android:versionCode="1"
                android:versionName="1.0"
                android:installLocation="auto">

                <supports-screens
                        android:smallScreens="true"
                                android:normalScreens="true"
                                android:largeScreens= "true"
                                android:anyDensity="true"
                />

                <application
                        android:icon="@drawable/ic_launcher"
                        android:label="@string/app_name" >

                        <activity android:name=".Helloworld"

                                           android:theme =
                                        "@android:style/Theme.NoTitleBar.Fullscreen"
                                        android:screenOrientation="portrait">
                                <intent-filter>
                                        <action
android:name="android.intent.action.MAIN" />

                                        <category
android:name="android.intent.category.LAUNCHER" />
                                </intent-filter>
                        </activity>

                </application>


        </manifest>
```

Finding Screen Layout Size

The Configuration object holds information related to screen size. It has a public field named screenLayout, which is a bitmask indicator which helps to find out the screen type. The following code sample shows how to find the screen size:

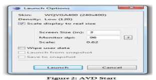
```
        if((getResources().getConfiguration().screenLayout &
                        Configuration.SCREENLAYOUT_SIZE_LARGE) ==
                        Configuration.SCREENLAYOUT_SIZE_LARGE){
                        //do something for Screen layout large
                }else{
```

```
                          //do something
                 }
```
You can find out the device’s screen density in a similar fashion:

```
        private final float LOW_LEVEL=0.75f;
        private final float MEDIUM_LEVEL=1.0f;
        private final float HIGH_LEVEL=1.5f;

        float level = getApplicationContext().getResources().getDisplayMetrics().density;

        if(level == LOW_LEVEL){

        //do something here

        }else if(level == MEDIUM_LEVEL){

        //do smoothing here

        }else if(level == HIGH_LEVEL){
        //do something here
        }
```
Adjusting Density on the Emulator

The Android SDK includes emulator skins which have their own default display sizes and densities. You can also modify the default size, density and resolution of an emulator skin to test any possible screen configuration. Adjusting the emulator density is as follows under the eclipse IDE:

Click on **Launch -> AVD Manager -> Start**



Skin: WQVGA400 (240x400)
Density: Low (120)
[✓] Scale display to real size
Screen Size (in): 3
Monitor dpi: 96
Scale: 0.62
[ ] Wipe user data
[ ] Launch from snapshot
[ ] Save to snapshot

Figure 2: AVD Start

You are ready to adjust the emulator density.

Utilities:

Currently Android provides some individual tools that can help the developers at the initial stage of their projects development, in terms of designing and handling multiple screen size.

1. Android Asset Studio

The Android Assets Studio is a web-based tool which is still in beta version. It is used for generating drawable resources and other assets for Android applications.
The available assets supported are:

- Launcher icons
- Menu icons
- Tab icons

- Notification icons
- Device frames

2. UI Prototyping Stencils

UI Prototyping Stencils is a GUI prototyping tool. It helps developers draw GUI prototyping that everyone can use.

The available assets supported are:

- Built-in stencils for diagramming and prototyping
- Multi-page document with background page
- Inter-page linking’s
- On-screen text editing with rich-text supports
- Exporting to HTML, PNG, Openoffice.org document, Word document and PDF.
- Undo/redo supports
- Installing user-defined stencils and templates
- Standard drawing operations: aligning, z-ordering, scaling, rotating...
- Cross-platforms
- Adding external objects
- Personal Collection
- Clipart Browser
- Object snapping
- Sketchy Stencil

ref:http://developer.samsung.com/android/technical-docs/Handling-Multiple-Screen-Size-in-Android