Assignment 01

COS30017 - Software Development for Mobile Devices

Daniel Parker 971328X

August 21, 2014

1. Task 1

1.1. Source

1.1.1. activity_melbourne.xml

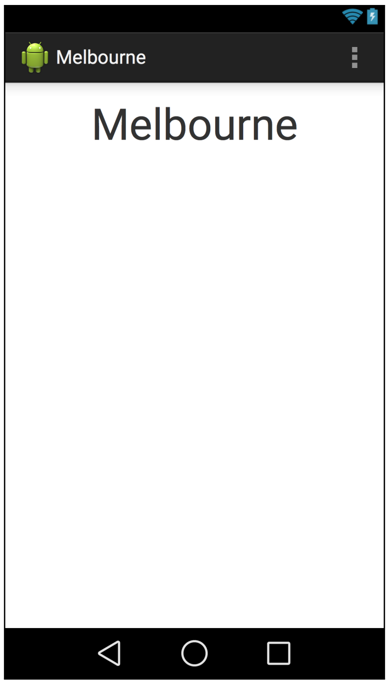
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MelbourneActivity">

    <TextView
        android:text="@string/melbourne"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20pt"
        android:gravity="center"/>

</RelativeLayout>
```

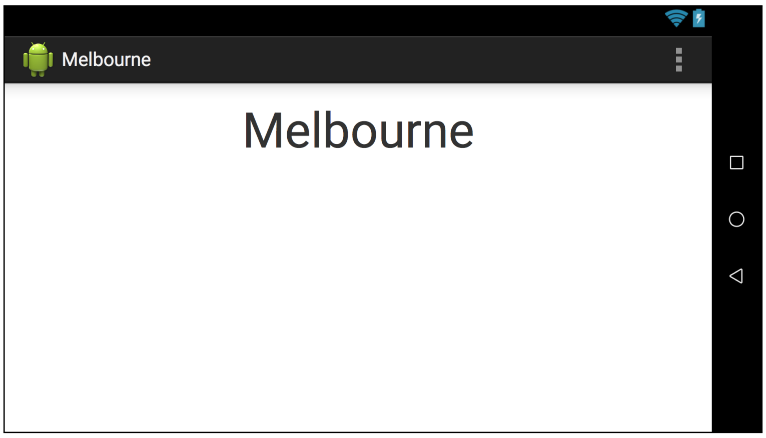
1.1.2. strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <string name="app_name">Melbourne</string>
    <string name="melbourne">Melbourne</string>
    <string name="action_settings">Settings</string>

</resources>
```

1.2. Screenshots



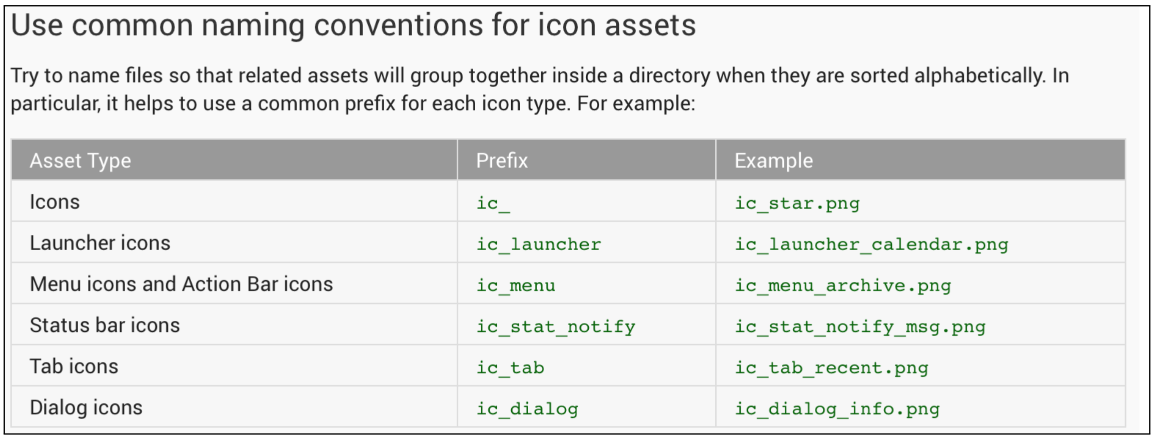


2. Task 2

A common design paradigm used in the software industry is that of convention over configuration. The purpose of convention over configuration is to reduce the number of decisions that have to be made by developers by providing a series of conventions as solutions for recurring decisions. Some examples are naming conventions, project structure conventions and coding conventions.

Android development encourages use of conventions, so much so that Android Studio, the integrated development environment developed by Google for Android development, will ensure certain conventions are followed and generate code automatically that follows those conventions.

The Android developer documentation recommends a naming convention for icon files as shown in this screenshot from the website.

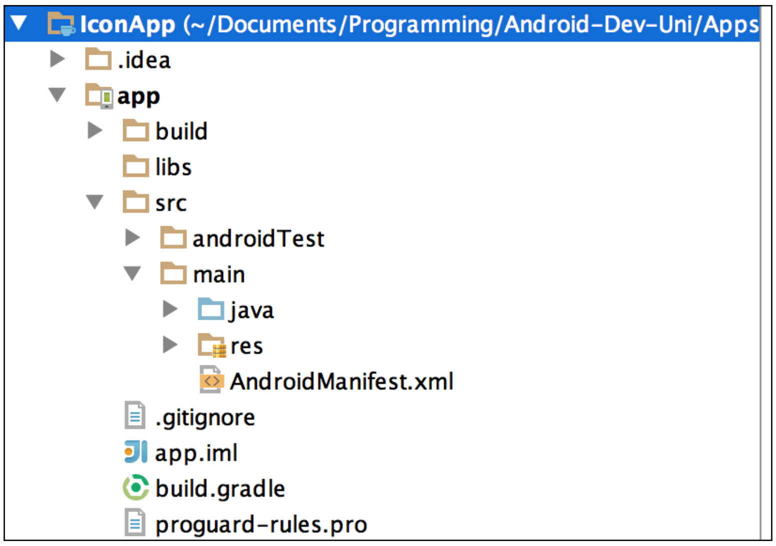


Use common naming conventions for icon assets

Try to name files so that related assets will group together inside a directory when they are sorted alphabetically. In particular, it helps to use a common prefix for each icon type. For example:

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`

Another good example of convention used in Android development is the structure of the project as generated by Android Studio when a project is setup. In the screenshot below we can see that the 'src' directory contains all the program source code but divided into java code in the 'java' directory and resources (such as images and XML) in the 'res' directory. Further directory conventions are used within each directory, all the way down to different drawable directories for different screen pixel densities.


```
IconApp (~/Documents/Programming/Android-Dev-Uni/Apps
  .idea
  app
    build
    libs
    src
      androidTest
      main
        java
        res
        AndroidManifest.xml
    .gitignore
    app.iml
    build.gradle
    proguard-rules.pro
```

3. Task 3

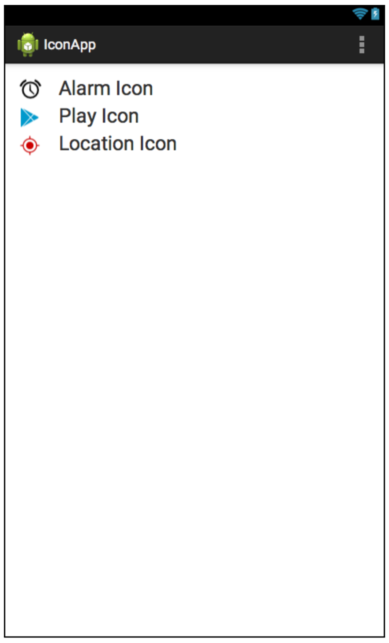
3.1. Source

3.1.1. activity_icon.xml

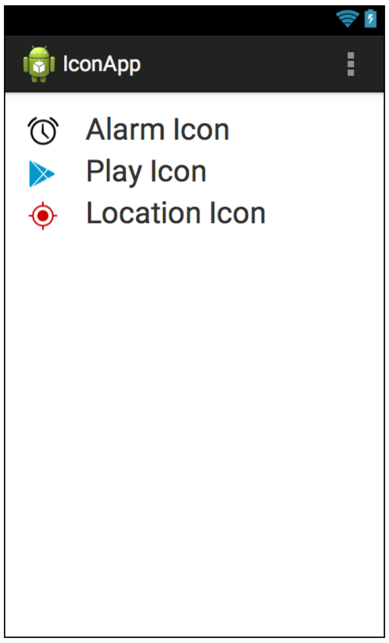
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".IconActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="35dp"
        android:src="@drawable/ic_action_alarm_2"
        android:id="@+id/alarm_icon"/>

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="35dp"
        android:layout_toRightOf="@id/alarm_icon"
        android:id="@+id/alarm_text"
        android:layout_marginLeft="20dp"
        android:text="Alarm Icon"
        android:textSize="25sp"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="35dp"
        android:layout_below="@id/alarm_icon"
        android:src="@drawable/ic_action_google_play"
        android:id="@+id/play_icon"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="35dp"
        android:layout_toRightOf="@id/alarm_icon"
        android:id="@+id/play_text"
        android:layout_below="@id/alarm_text"
        android:layout_marginLeft="20dp"
        android:text="Play Icon"
        android:textSize="25sp"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="35dp"
        android:layout_below="@id/play_icon"
        android:src="@drawable/ic_action_location_2"
        android:id="@+id/location_icon"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="35dp"
        android:layout_toRightOf="@id/alarm_icon"
        android:id="@+id/location_text"
        android:layout_below="@id/play_text"
        android:layout_marginLeft="20dp"
        android:text="Location Icon"
        android:textSize="25sp"/>

</RelativeLayout>
```

3.2. Screenshots

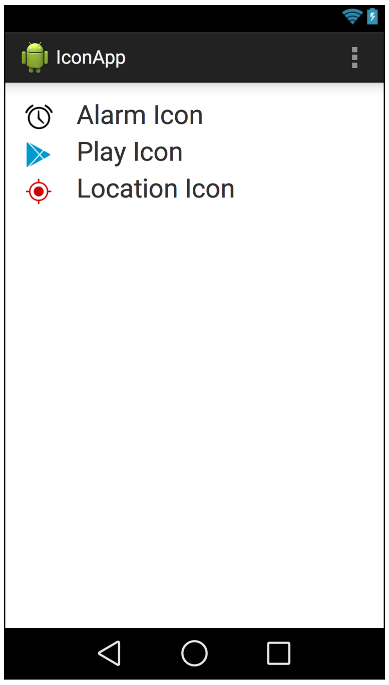
3.2.1. mdpi



3.2.2. hdpi

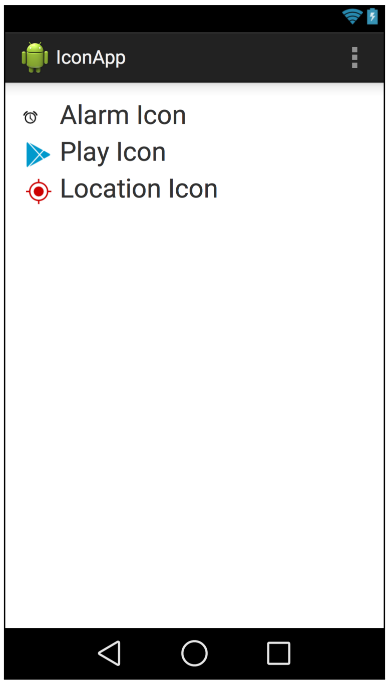


3.2.3. xxhdpi



3.3. Low density icon on XXHDPI screen

It is possible to force a high density screen to show a low density icon but the icon will either display much smaller than it should be or it will be the correct size but look pixelated. That will be determined by whether the XML defines sized in 'dp' or 'px'. 'dp' sizes will scale proportionally based on the device's pixel density. The screenshot below shows how the low density alarm image will look smaller on a high density screen.



4. Task 4

There is a need to use device independent font sizing because:

- The pixel density is different between devices.
- Fonts will not show correctly across different devices if device dependent font sizing is used, ie. 'px' instead of 'sp'.
- Fonts need to scale correctly so that they are readable on different density screens.

4.1. Snippet

```
android:textSize="25sp"/>
```