Lesson Four: Launcher Icons, Button Images, and Activity Backgrounds

Images can be used several additional ways inside an Android application. In this lesson, you'll learn how to configure application icons, button images, and backgrounds for your activity screen.

Launcher Icons

Every program has a *launcher icon*. This is image identifies your program in the application launcher on the device. When you create a new project, you receive a default icon of a little Android.

While this is a nice image, it's not very descriptive of your application. If you are writing a program for Google Play, you should make a more creative image that better describes or "brands" your program. What should your icon look like? It's your decision, although you should follow some common rules:

- Don't include the name of your application in the image. The application launcher will write the name of your program under the icon automatically.
- Make sure your icon is crisp and colorful so that it stands out on any background.
- Don't make your icon needlessly complicated. Regardless of the screen density, the icon will be relatively small on the device. Tiny details on a small image never turn out very well!
- Take your time when creating this icon. It is the first impression for your application. Make it count!

The Android team has come up with many other tips and tricks for creating great launcher icons. You can check out http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html or search online for "android launcher icon design". Once you settle on a design for your application icon, you will need to create three or four versions: one for each of the screen densities that you are targeting. Here are the overall sizes for the different densities:

- Low Density (ldpi) – 36 pixels x 36 pixels
- Medium Density (mdpi) – 48 pixels x 48 pixels
- High Density (hdpi) – 72 pixels x 72 pixels
- Tablets (xhdpi) – 96 pixels x 96 pixels

Each image should be added to the corresponding drawable directory in your project. By default, the launcher icon image is called "ic_launcher.png", but you can select a different name in the "AndroidManifest.xml" file. To change the name, open the XML and find the <Application> tag. You can change this attribute to any other valid drawable resource in your project:

```
    android:icon="@drawable/ic_launcher"
```

How do you test your new launch icons in the emulator? Once you load your application into the emulator, you can exit it with the back arrow key. Then click on the application grid at the bottom to see a list of all installed applications and launch icons, including your own.

The ImageButton Control

ImageButtons are like regular buttons that allow you to display an image on the button instead of text. This can give your buttons a snazzy style or give button-like functionality to an image.

To create an **ImageButton** in your XML layout file, add the following code:

```
    <ImageButton
        android:id="@+id/MyImageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/snowflake" />
```

Notice that this definition is very similar to the plain **Button** control. The only real difference is the addition of the "android:src" attribute, which points to a drawable image in your project. In addition to one static image selected in the layout, you can also choose to use different images on the button, depending on the state of the button. For example, you can have one image for the normal button, another image which shows that the button has focus and yet another to show that the button is being pressed down on the screen.

To add multiple button images, you will need to create a selector XML file in one of your "drawable" folders. The XML file should contain a <selector> root element and one or more child <item> elements. Each <item> can contain a "state" attribute such as "android:state_pressed" or "android:state_focused" and a corresponding "android:drawable" attribute with the drawable ID.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:drawable="@drawable/button_pressed" />
    <item android:state_focused="true"
          android:drawable="@drawable/button_focused" />
    <item android:drawable="@drawable/button_normal" />
</selector>
```

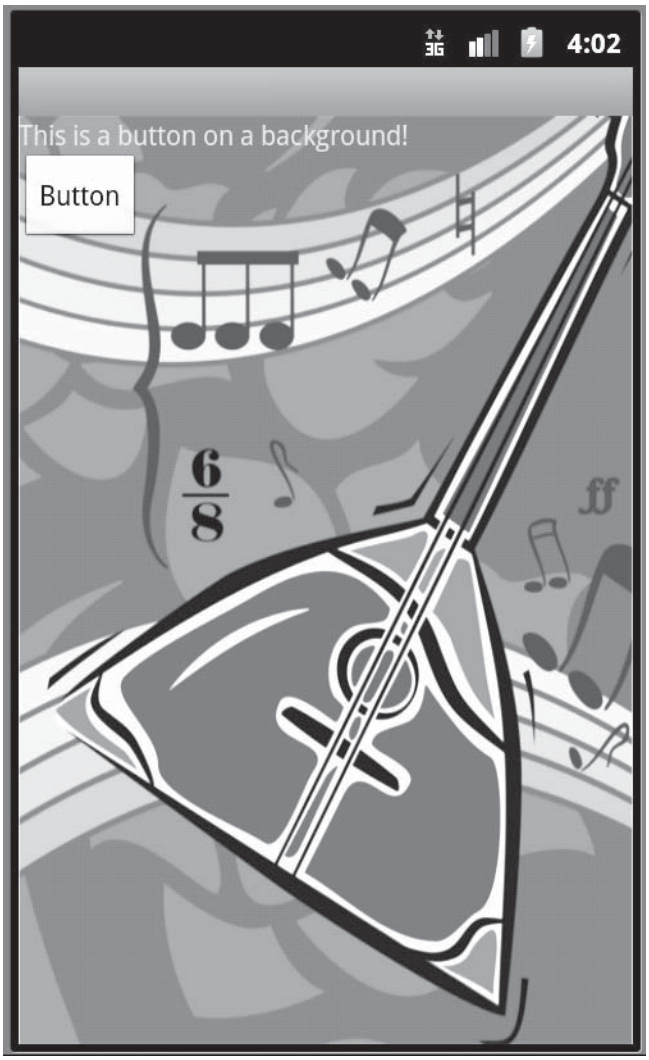
Put the <item> element for the normal button (without any special state attributes) last in the file, so it will be the default image if none of the <item> states above it match the current button condition. Then save your XML file with any valid filename such as "button_selector.xml".

Once this file is saved in project, you can use it as your **ImageButton**'s "android:src" in the XML layout file:

```
    android:src="@drawable/button_selector"
```

The **ImageButton** will then use the images listed in the selector XML to choose the correct picture to display depending on the current state of the ImageButton. Our snowflakes now have normal (left), focused (middle), and clicked (right) images.

Activity Backgrounds



You can also use an image as the background of a layout. This might be useful if you are creating a game program or if you just want to add a little extra style to your application. To select an image resource as the background of a layout, add the "android:background" attribute to your layout element. For example, here we have added an image called "bg.png" as the background of a **LinearLayout** that covers the entire screen:

```
    <LinearLayout
      ...
      android:background="@drawable/bg"
      ...
    </LinearLayout>
```

You will then see the image appear behind any other controls or views on the screen.