Android Action Bar from the Options Menu

As of March 5, 2012 over 95% of devices accessing the Android Market (now called Google Play) use Android 2.3.7 or less (source for that fact can be found here). Users (and application developers) of these devices have become accustomed to the options menu. That is, the menu that pops up on the bottom (typically) of the device when the user presses the Menu button. Below, an example set of 3 menu items on an options menu are displayed when the user pushes the menu button on the device.
Bye Bye Menu Button

As of Android 3.0 (Honeycomb), the standard four Android buttons of Home, Back, Menu and Search are replaced by Home, Back and a Recent Apps button. Below is a picture of Galaxy Nexus (running Android 4.0 offered by Verizon) that shows an example of the new 3 button system.
Tablets, which run Android 3 and for which Android 3 was really created, have no hardware buttons. The three buttons (Home, Back, and Recent Apps) are offered as “virtual buttons” on the screen (see Motorola Xoom as an example of such device below).
In fact, and as a side note, the future of hardware buttons on Android platforms (smartphones, tablets, etc.) is questionable. Many believe the time has come to move to only virtual buttons (see editorial comment [here](#)). Regardless of whether the buttons are represented in hardware or virtually, the Menu button and the options menu associated to it are gone. So, how does one provide the type of capability provided by the options menu on Android 3 and later devices? The answer is through action bar. In a recent blog post, Scott Main who is the lead tech writer for developer.android.com, warns developers that they need to start making this transition – and soon.

**Transitioning to the Action Bar**

The good news is that the transition from options menu to action bar is quite simple. Indeed, the action bar provides for a lot more capability than the options menu. Therefore, you can spend a lot of time on improving and upgrading your applications to take advantage of its full capability. See the action bar [documentation](#) to look into all the action bar features. In this post, however, I want to show you how easy it is to take a typical options menu and move it to an action bar, and then show you what that looks like to the user. To begin, say your application has an options menu defined in an XML menu resource like that below.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/courses" android:icon="@drawable/course"
      android:title="@string/courses" />
   <item android:id="@+id/scores" android:icon="@drawable/score"
      android:title="@string/scores" />
   <item android:id="@+id/handicap" android:icon="@drawable/handicap"
      android:title="@string/handicap" />
</menu>
```
In an application where the AndroidManifest.xml file suggests that the minimum version of the Android SDK is 10 (Android 2.3.3), then the menu buttons appear as shown below in a smartphone AVD.

<uses-sdk android:minSdkVersion="10" />
If you were to simply change the minimum version of the SDK to be 11 (suggesting Android 3.0), then it might surprise you to see the AVD displays as shown below when you press the menu button.

![Image of AVD displaying menu items]

In order to use the new action bar, add an android:showAsAction attribute to your menu items. The value of the showAsAction attribute can be any one of the values described in the table below.

<table>
<thead>
<tr>
<th>showAsAction value option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“always”</td>
<td>Always have the item appear on the action bar (as opposed to being placed in the overflow menu when space is limited – see below).</td>
</tr>
<tr>
<td>“ifRoom”</td>
<td>Have the item appear in the action bar, but only if there is room available on the action bar. If there’s not enough room, the item appears in the overflow menu.</td>
</tr>
<tr>
<td>“never”</td>
<td>Never have the item appear in the action bar. Always display the item on the overflow menu.</td>
</tr>
</tbody>
</table>
In the example below, the same menu is defined in XML, but this time with the showAsAction attribute is set for each item.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/courses" android:icon="@drawable/course"
          android:title="@string/courses" android:showAsAction="ifRoom"/>
    <item android:id="@+id/scores" android:icon="@drawable/score"
          android:title="@string/scores" android:showAsAction="ifRoom"/>
    <item android:id="@+id/handicap" android:icon="@drawable/handicap"
          android:title="@string/handicap" android:showAsAction="ifRoom"/>
</menu>
```

This causes the action bar to display with the items’ icons displayed in the bar.
You might notice only 2 of the 3 original menu items are displayed in the new action bar, and furthermore, the title string is missing. Given the “ifRoom” value to the showAsAction attribute, Android determined there was only room for 2 of the options. To get the other menu items, you must go to the “overflow menu.” To get to the overflow menu, press the Menu button on the AVD. On actual devices, a soft options menu button appears as three vertical dots (see example below). It should be noted that this allows for the “normal” options menu to be used although not recommended.

When the overflow menu button is clicked, the other (or overflow) menu items appear.
As for the menu item title, you can add the withText modifier to the attribute value. If your menu items have a title and an icon only the icon is displayed in the action bar by default. Add “withText” as shown below to include the text with the icon in the the action bar. In this case, the screen is shown in landscape orientation (and with the overflow menu shown) in order to make sure there is room to seem some of the items with their text.

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/courses" android:icon="@drawable/course"
          android:title="@string/courses" android:showAsAction="ifRoom|withText"/>
    <item android:id="@+id/scores" android:icon="@drawable/score"
          android:title="@string/scores" android:showAsAction="ifRoom|withText"/>
    <item android:id="@+id/handicap" android:icon="@drawable/handicap"
          android:title="@string/handicap" android:showAsAction="ifRoom|withText"/>
</menu>
```
The Android documentation on action bars makes an important point with regard to the title and why you should always provide it even if you don’t always use the “withText” modifier.

It’s important that you always define android:title for each menu item, even if you don’t declare that the title appear with the action item, for three reasons:

1. If there’s not enough room in the action bar for the action item, the menu item appears in the overflow menu and only the title appears.
2. Screen readers for sight-impaired users read the menu item’s title.
3. If the action item appears with only the icon, a user can long-press the item to reveal a tool-tip that displays the action item’s title.
Creating, Inflating, and Reacting to Selection

In your activity code, you create and inflate the action bar in the same way that the options menu is created and inflated. Use the onCreateOptionsMenu method to create action bar items. The onPrepareOptionsMenu method is called right before the menu is shown. Given the action bar is always displayed, this method may not be called as often as it used to be with the options menu. However, it will be called before the overflow menu is shown too.

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.options, menu);
    return true;
}
```

```java
public boolean onPrepareOptionsMenu(Menu menu) {
    // preparation code here
    return super.onPrepareOptionsMenu(menu);
}
```

To react to an action bar item being selected, implement the onOptionsItemSelected method just as you would in the options menu case.

```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.courses) {
        startActivity(new Intent(this, CoursesActivity.class));
    }
    if (item.getItemId() == R.id.scores) {
        startActivity(new Intent(this, ScoresActivity.class));
    }
    if (item.getItemId() == R.id.handicap) {
```
Consultants Who Teach

startActivity(new Intent(this, HandicapActivity.class));

return super.onOptionsItemSelected(item);

Removing the Action Bar

You can remove the action bar for any activity. In the AndroidManifest.xml file, simply set the theme of the activity to Theme.Holo.NoActionBar.

<activity
    android:label="@string/app_name"
    android:name=".GolfAndroidActivity" android:theme="@android:style/Theme.Holo.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

Programmatically, you can also get the action bar with a call to getActionBar(). Once obtained, you can request to hide it. Android adjusts the layout/display of the screen when the action bar is removed.

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ActionBar bar = getActionBar();
    bar.hide();
}

A copy of a little demo app (as used in this post) is available for you to try out the action bar. You’ll find the Eclipse project (zipped) for download here.
Wrap up

Again, there are a number of additional and more advanced features that come with the new action bar. This includes split action bars, application navigation, action view (a widget that appears in the action bar as a substitute for an action item’s button) and navigational tabs to name a few. As you explore the action bar as a replacement for the options menu, you might also find this new UI mechanism replaces other complex UIs like the TabWidget.

Feel like you need to learn Android soon? Come join me at Intertech for Android training. Find details about the class [here](#).