

ITW202: Mobile Application

Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

May 12, 2021

Mobile Application Development

Drawables, styles, and themes



Mobile Application Development

Drawables



Drawables

- Drawable—generic Android class used to represent any kind of graphic
- All drawables are stored in the res/drawable project folder
- Example: You apply a Drawable to an XML resource using attributes such as android:drawable and android:icon.

Using drawables

To display a Drawable, use the ImageView class to create a View. In the <ImageView> element in your XML file, define how the Drawable is displayed and where the Drawable file is located.

```
<ImageView  
    android:id="@+id/tiles"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/birthdaycake" />
```

Using drawables

To represent a drawable in your app, use the Drawable class or one of its subclasses.

```
Resources res = getResources();  
Drawable drawable =  
res.getDrawable(R.drawable.birthdaycake);
```

Drawable classes

- Bitmap File
- Nine-Patch File
- Layer List Drawable
- Shape Drawable
- State List Drawable
- Level List Drawable
- ...and more Custom Drawables

Mobile Application
Development



Image files

Mobile Application Development

- An image file is a generic bitmap file. Android supports image files in several formats: WebP (preferred), PNG (preferred), and JPG (acceptable). GIF and BMP formats are supported, but discouraged.

Image files

Mobile Application Development

- The WebP format is fully supported from Android 4.2. WebP compresses better than other formats for lossless and lossy compression, potentially resulting in images more than 25% smaller than JPEG formats.

Image files

- Store image files in the res/drawable folder. Use them with the android:src attribute for an ImageView.

Mobile Application
Development



Nine-patch files

- A 9-patch is a PNG image in which you define stretchable regions. Use a 9-patch as the background image for a View to make sure the View looks correct for different screen sizes and orientations.
- If you use a 9-patch image instead, the 9-patch stretches as the View stretches.

Nine-patch files

- The Android standard Button is an example of a View that uses a 9-patch as its background image. The 9-patch stretches to accommodate the text or image inside the Button.

Layer list drawables

- Each layer is represented by an individual Drawable.
- The drawables that make up a single image are organized and managed in a `<layer-list>` element in XML.
- Within the `<layer-list>`, each Drawable is represented by an `<item>` element.



Layer list drawables

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <bitmap android:src="@drawable/android_red"
            android:gravity="center" />
    </item>
    <item android:top="10dp" android:left="10dp">
        <bitmap android:src="@drawable/android_green"
            android:gravity="center" />
    </item>
    <item android:top="20dp" android:left="20dp">
        <bitmap android:src="@drawable/android_blue"
            android:gravity="center" />
    </item>
</layer-list>
```

Shape drawables

- A shape Drawable is a rectangle, oval, line, or ring that you define in XML.
- Styled with attributes such as `<corners>`, `<gradient>`, `<padding>`, `<size>`, `<solid>` and `<stroke>`

Shape drawables

For example, this XML file creates a rectangle with rounded corners and a color gradient.

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <corners android:radius="8dp" />
    <gradient
        android:startColor="#000000"
        android:endColor="#0000dd"
        android:angle="45"/>
    <padding android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
</shape>
```


Shape drawables

Assuming that the shape Drawable XML file is saved at `res/drawable/gradient_box.xml`, the following layout XML applies the shape Drawable as the background to a View:

```
<TextView  
    android:background="@drawable/gradient_box"  
    android:layout_height="wrap_content"  
    android:layout_width="wrap_content" />
```

Shape drawables

The following code shows how to programmatically get the shape Drawable and use it as the background for a View, as an alternative to defining the background attribute in XML:

```
Resources res = getResources();  
Drawable shape = res. getDrawable(R.drawable.gradient_box);  
  
TextView tv = (TextView)findViewById(R.id.textview);  
tv.setBackground(shape);
```

Vector drawables

- In Android 5.0 (API Level 21) and above, you can define vector drawables, which are images that are defined by a path.
- A vector Drawable scales without losing definition.
- Most vector drawables use SVG files, which are plain text files or compressed binary files that include two-dimensional coordinates for how the image is drawn on the screen.

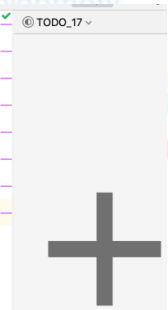
Vector drawables

Mobile Application Development

- Because SVG files are text, they are more space efficient than most other image files.
- Also, you only need one file for a vector image instead of a file for each screen density, as is the case for bitmap images.

Vector drawables

```
<vector android:height="40dp"
    android:tint="?attr/colorControlNormal"
    android:viewportHeight="24"
    android:viewportWidth="24"
    android:width="40dp"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="@android:color/white"
        android:pathData="M19,13h-6v6h-2v-6H5v-2h6V5h2v6h6v2z"/>
</vector>
```



Vector drawables

Advantages of using vector drawables as icons:

- Vector drawables can reduce your APK file size dramatically, because you don't have to include multiple versions of each icon image. You can use one vector image to scale seamlessly to any resolution.
- Users might be more likely to download an app that has smaller files and a smaller package size

Vector drawables

Mobile Application Development

Disadvantages of using vector drawables as icons:

- A vector drawable can include only a limited amount of detail. Vector drawables are mostly used for less detailed icons.
- Vector drawables are not supported on devices running API level 20 or below.

Vector drawables

To use vector drawables on devices running API level 20 or below, you have to decide between two methods of backward-compatibility:

- By default, at build time the system creates bitmap versions of your vector drawables in different resolutions. This allows the icons to run on devices that aren't able to draw vector drawables.
- The **VectorDrawableCompat** class in the Android Support Library allows you to support vector drawables in Android 2.1 (API level 7) and higher.

Mobile Application Development

Image Asset Studio

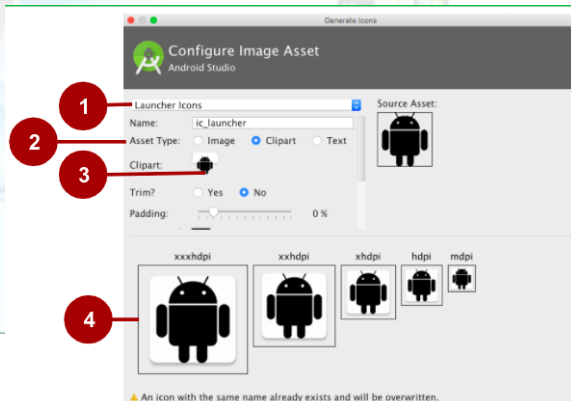


What is Image Asset Studio?

- Create icons from material icons, images, and text
- Launcher, action bar, tab, notification icons
- Generates a set of icons for generalized screen density
- Stored in /res folder
- To start Image Asset Studio
 - 1 Right-click the res folder of your project
 - 2 Choose New > Image Asset

Using Image Asset Studio

- 1 Chose icon type and change name
- 2 Choose Image, Clipart, or Text
- 3 Click icon to chose clipart
- 4 Inspect assets for multiple screen sizes



Mobile Application Development

Vector Asset Studio

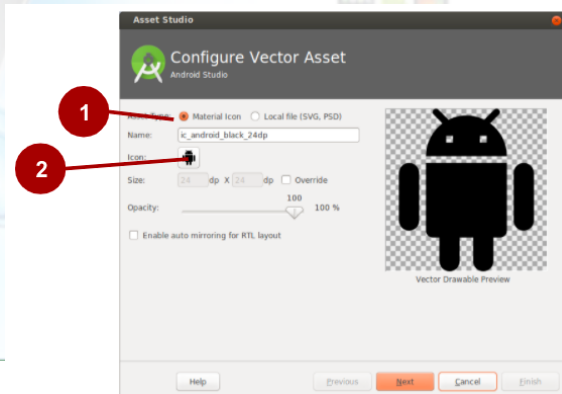


What is Vector Asset Studio?

- Create icons from material icons or supply your own vector drawings for API 21 and later
- Launcher, action bar, tab, notification icons
- Generates a scalable vector drawable
- Stored in res folder
- To start Image Asset Studio
 - 1 Right-click res folder of your project
 - 2 Choose New > Vector Asset

Using Image Asset Studio

- 1 Choose from Material Icon library, or supply your own SVG or PSD vector drawing
- 2 Opens Material Icon library



Mobile Application Development

Styles



What is a Style?

Mobile Application Development

- Collection of attributes that define the visual appearance of a View
- Reduce duplication
- Manage visual appearance of many components with one style

Defining and applying styles

A `<style>` element includes the following:

- A name attribute. Use the style's name when you apply the style to a View.
- An optional parent attribute.
- Any number of `<item>` elements as child elements of `<style>`. Each `<item>` element includes one style attribute.

Defining and applying styles

styles.xml is in **res/values**

```
<resources>
    <style name="CodeFont">
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

Inheritance: Parent

Mobile Application Development

Define a parent style...

```
<resources>
    <style name="CodeFont">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

Inheritance: Define child

Mobile Application Development

Define child with Codefont as parent

```
<resources>
  <style name="RedCode" parent="@style/Codefont">
    <item name="android:textColor">#FF0000</item>
  </style>
</resources>
```

Mobile Application Development

Themes



Themes

Mobile Application Development

- A Theme is a style applied to an entire activity or even the entire application
- Themes are applied in AndroidManifest.xml
`<application
android:theme="@style/AppTheme">`

Themes

Mobile Application

What's the difference between a style and a theme?

- A style applies to a View. In XML, you apply a style using the style attribute.
- A theme applies to an Activity or an entire app, rather than to an individual View. In XML, you apply a theme using the android:theme attribute.

Applying themes

Mobile Application Development

To apply a theme to your app, declare it inside an `<application>` element inside the `AndroidManifest.xml` file.

- `android:theme="@style/AppTheme"`

Applying themes

Mobile Application Development

To apply a theme to an Activity, declare it inside an `<activity>` element in the `AndroidManifest.xml` file.

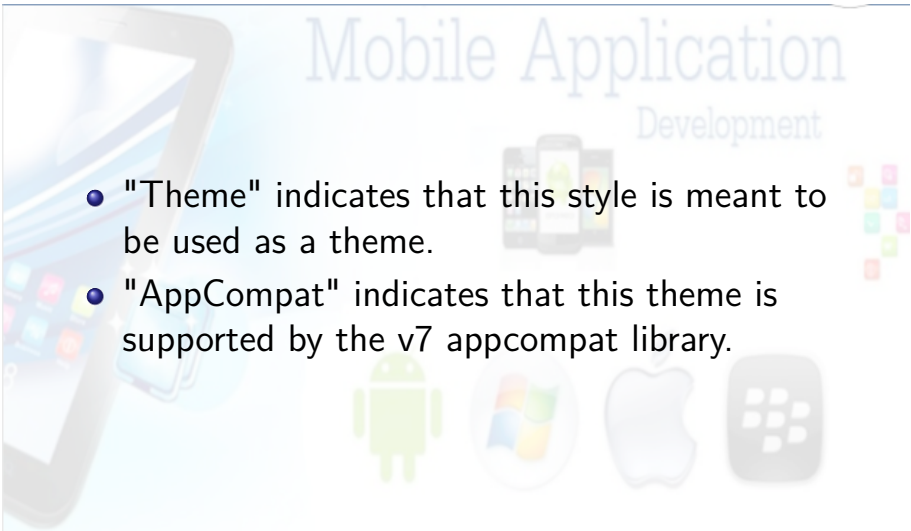
- `<activity android:theme="@android:style/Theme.Dialog">`

Default theme

When you create a new project in Android Studio, a default theme is defined for you within the styles.xml file.

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Default theme

- 
- The background of the slide features a light blue gradient. On the left, there is a large, stylized illustration of a tablet and a smartphone. In the center, the text "Mobile Application Development" is written in a large, light blue, sans-serif font. Below this text, there are several small, circular icons: a green Android robot, a Windows logo, an Apple logo, and a BlackBerry logo. To the right of the text, there is a small cluster of colorful squares.
- "Theme" indicates that this style is meant to be used as a theme.
 - "AppCompat" indicates that this theme is supported by the v7 appcompat library.

Default theme

- "Light" indicates that the theme consists of light background, white by default. All the text colors in this theme are dark, to contrast with the light background. (If you wanted a dark background and light text, your theme could inherit from a theme such as Theme.AppCompat, without "Light" in the name.)

Default theme

- "DarkActionBar" indicates that a dark color is used for the action bar, so any text or icons in the action bar are a light color.
- Another useful theme is `Theme.AppCompat.DayNight`, which enables the user to browse in a low-contrast "night mode" at night.

Mobile Application Development

THANK YOU

