Android Developer Fundamentals

# Delightful User Experience

Lesson 5

# 5.1 Drawables, Styles, Themes

# Contents

- Drawables
- Creating image assets
- Styles
- Themes

# Drawables

# Drawables

- Drawable—generic Android class used to represent any kind of graphic
- All drawables are stored in the **res/drawable** project folder

# Drawable classes

Bitmap File
Nine-Patch File
Layer List Drawable
Shape Drawable
State List Drawable
Level List Drawable

Transition Drawable
Vector Drawable
… and more
Custom Drawables

# Bitmaps

- PNG (.png), JPG (.jpg), or GIF (.gif) format
- Uncompressed BMP (.bmp)
- WebP (4.0 and higher)
- Creates a BitmapDrawable data type
- Placed directly in res/drawables

# Referencing Drawables

- XML: `@[package:]drawable/filename`

```
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />
```

- Java code: `R.drawable.filename`

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

# Nine-Patch Files

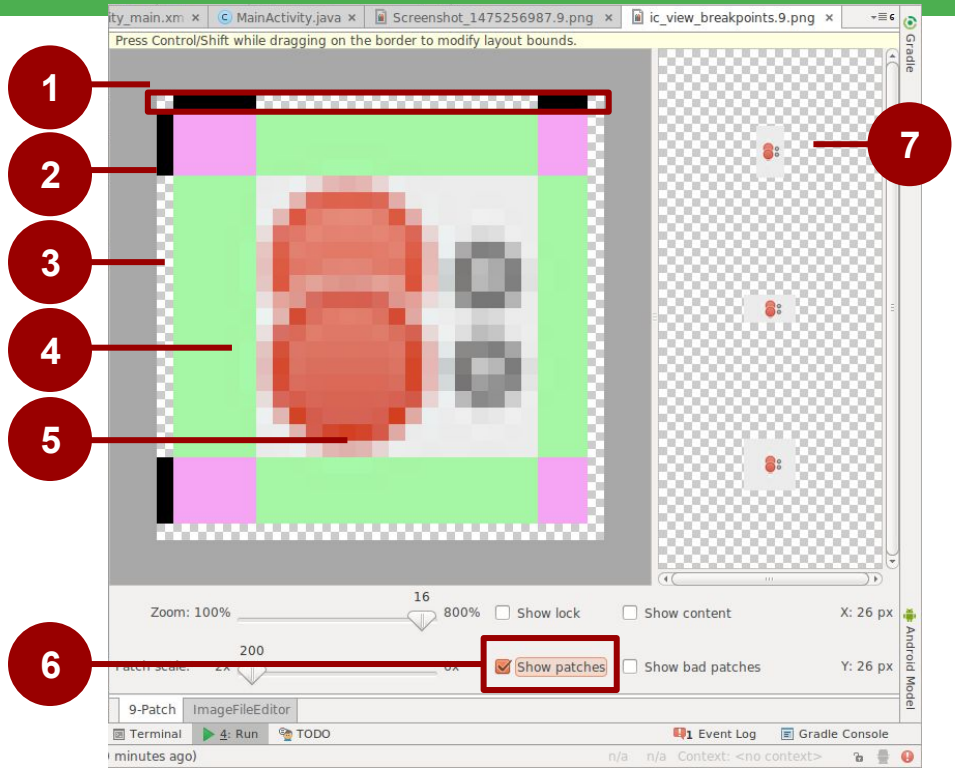
- Nine-patch files (.9.png) are PNG with stretchable regions
- Only stretches bigger, not smaller, so start with small image
- Often used for backgrounds of UI elements
- Example: button background changes size with label length
- Good intro

# Creating Nine-Patch Files

1. Put a small PNG file into **res/drawable**
2. Right-click and choose **Create 9-Patch file**
3. **Double-click** 9-Patch file to open editor
4. Specify the stretchable regions (next slide)

# Editing Nine-Patch Files

1. Border to mark stretchable regions for width
2. Stretchable regions marked for height
   Pink == both directions
3. Click to turn pixels black. Shift-click (ctrl-click on Mac) to unmark
4. Stretchable area
5. Not stretchable
6. Check **Show patches**
7. Preview of stretched image



# Layer List

- You can create layered images, just like with drawing tools, such as Gimp
- In Android, each layer is represented by a drawable
- Layers are organized and managed in XML
- List and the items can have properties
- Layers are drawn on top of each other in the order defined in the XML file
- LayerDrawable

# Creating Layer List

```
<layer-list>
    <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 & GradientDrawable

- Define a shape and its properties in XML
  - Rectangle, oval, ring, line
- Styled with attributes such as <corners>, <gradient>, <padding>, <size>, <solid> and <stroke>

  See Shape Drawables for more attributes
- Can be inflated for a GradientDrawable

# Creating a GradientDrawable

```
<shape … android:shape="rectangle">
  <gradient
     android:startColor="@color/white"
     android:endColor="@color/blue"
     android:angle="45"/>
  <corners android:radius="8dp" />
</shape>
```


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

TextView tv = (TextView)findViewByID(R.id.textview);
tv.setBackground(shape);
```

# Transition Drawables

- Drawable that can cross-fade between two other drawables
- Each graphic represented by <item> inside <selector>
- Represented by TransitionDrawable in Java code
- Transition forward by calling `startTransition()`
- Transition backward with `reverseTransition()`

# Creating Transition Drawables

```
<transition ...>
  <selector> <item android:drawable="@drawable/on" />
             <item android:drawable="@drawable/off" />
  </selector>
</transition>
```

```
<ImageButton
    android:id="@+id/button"
    android:src="@drawable/transition" />
```

```
ImageButton button = (ImageButton) findViewById(R.id.button);
TransitionDrawable drawable =
        (TransitionDrawable) button.getDrawable();
drawable.startTransition(500);
```

# Vector drawables

- Scale smoothly for all screen sizes
- Android API Level 21 and up
- Use Vector Asset Studio to create (slides below)
- VectorDrawable

# Creating Vector drawables

```
<vector ...
    android:height="256dp" android:width="256dp"
    android:viewportWidth="32" android:viewportHeight="32">
  <path android:fillColor="@color/red"
      android:pathData="M20.5,9.5
                        c-1.955,0,-3.83,1.268,-4.5,3
                        c-0.67,-1.732,-2.547,-3,-4.5,-3 ... />
</vector>
```

*pathData for heart shape*

# Image Asset Studio

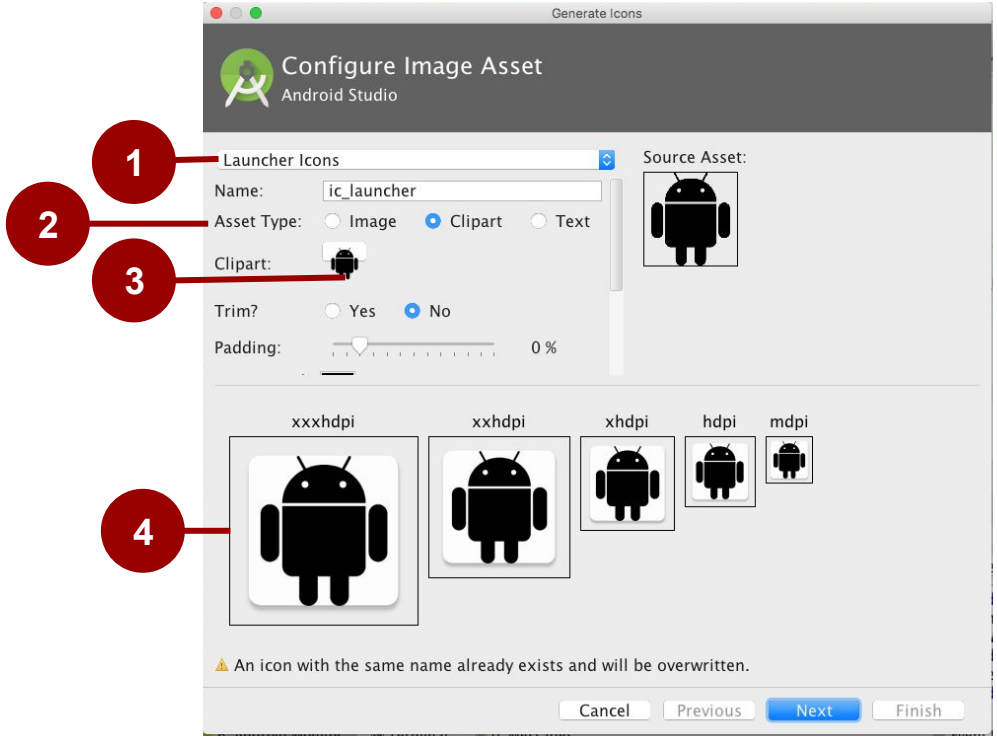
# What is Image Asset Studio?

- 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 **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



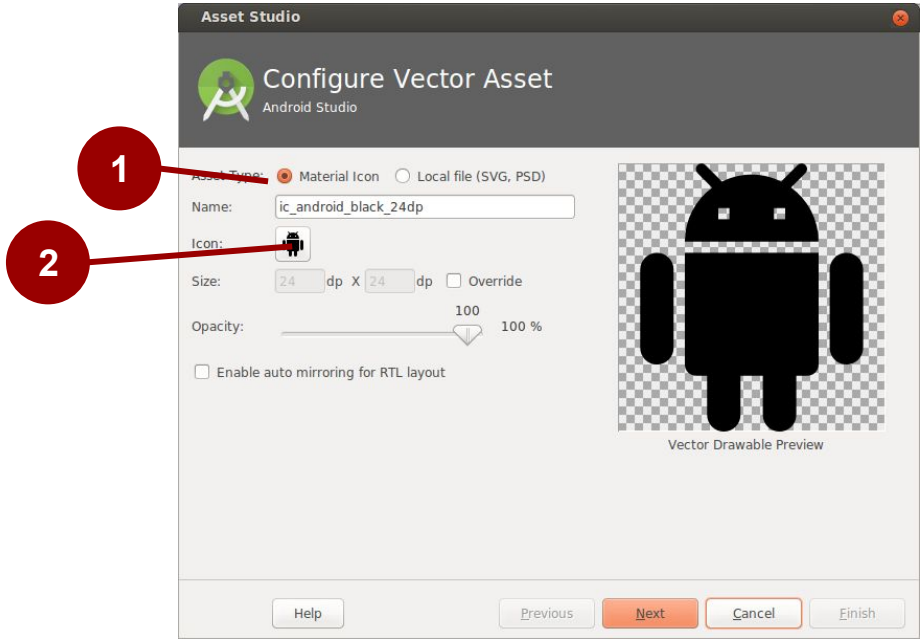
# Vector Asset Studio

# What is Vector Asset Studio?

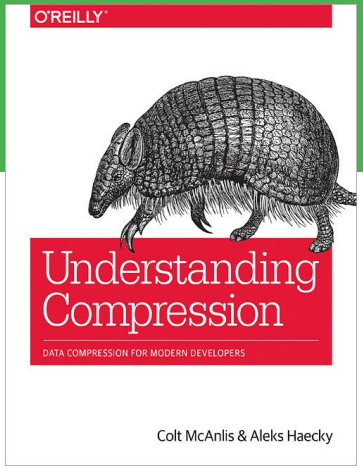
- Create icons from material icons or supply your own vector drawings for API 21 and later
- Launcher, action bar, tab, notification icons
- Generates a scalable vector drawable
- Stored in **/res** folder
- To start Image Asset Studio
  1. Right-click **res** folder of your project
  2. Choose **New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



# Images, memory, and performance




- Use smallest resolution picture necessary
- Resize, crop, compress
- Vector drawings for simple images
- Use Libraries: Glide or Picasso
- Choose appropriate image formats for image type and size
- Use lossy image formats and adjust quality where possible
- Learn about data compression for developers from Understanding Compression

# Styles

# What is a Style?

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

# Styles reduce clutter

```
<TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:textColor="#00FF00"
  android:typeface="monospace"
  android:text="@string/hello" />
```

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello"
/>
```

# Define styles in styles.xml

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

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont">
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

# Inheritance: Parent

Define a parent style...

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

Define child with Codefont as parent

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="RedCode" parent="@style/Codefont>
        <item name="android:textColor">#FF0000</item>
    </style>
</resources>
```

# Themes

# Themes

- A Theme is a style applied to an entire activity or even the entire application
- Themes are applied in the Android Manifest

```
<application android:theme="@style/AppTheme">
```

# Customize AppTheme of Your Project

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

# Styles and Themes Resources

Android platform has collection of built in styles and themes

- Android Styles
- Android Themes
- Styles and Themes Guide
- DayNight Theme Guide

# Learn more

- Drawable Resource Documentation
- ShapeDrawable
- LinearLayout Guide
- Drawable Resource Guide
- Supported Media formats
- 9-Patch
- Understanding Compression

# What's Next?

- Concept Chapter: 5.1 C Drawables, Styles, and Themes
- Practical: 5.1 P Drawables, Styles, and Themes

# END