UNIVERSITY of WASHINGTON

Android 250 - Lecture 2
Resources

Margaret Maynard-Reid
April 6, 2015

Agenda

Topics

- Resources
- Drawables
- 9-Patch Drawable
- Homework 1 Requirements

Sample Code

- SampleDrawables
- SampleAssets

Android Stories

- Google's ARC Beta runs Android apps on Chrome OS, Windows, Mac, and Linux
- http://www.androidpolice.com/2015/04/04/getting-know-android-5-0-lollipop-edition/

Review from Last Week

- What is Screen Density?
- What 2 Android units of measure account for Screen Density?
- What is the size of the minimal touch area in UI?
- What is the View Hierarchy?

Resources

Overview

Resources

Resources are

- Elements of application content
- Play a key role in Android architecture, as well as supporting multiple screens
- XML definition or raw files (images, audio video or text files)

Type of Resources

- **Animation** - compiled animation files, show predetermined animations.
- **Color** - define a color resources that changes based on the View state.
- **Drawable**
  - BitmapDrawable
  - NinePatchDrawable
  - LayerDrawable
  - StateListDrawable
  - LevelListDrawable
  - TransitionDrawable
  - InsetDrawable
  - ClipDrawable
  - ScaleDrawable
  - ShapeDrawable

- Layout - define UI and views
- Menu - define menus
- Values
  - arrays.xml
  - colors.xml
  - dimens.xml
  - strings.xml
  - styles.xml
- XML: compiled arbitrary XML files
- Raw: non- compiled raw files
  - text files
  - audio files
  - video files

Demo

- Take a look at where the resources live

Create Resources XML/Code

For each Resource there is generally a corresponding code element and an XML element

Sometimes they don’t match up cleanly

- <bitmap> = BitmapDrawable
- <layer-list> = LayerDrawable
- <selector> = StateListDrawable
- <transition> = TransitionDrawable

Resources

Drawables

Drawable - What is it?

A general concept for a graphic that can be drawn.

BitmapDrawable

1. Typically a GIF, JPG, or PNG source file or
2. An XML reference (with options)

Can be created in code

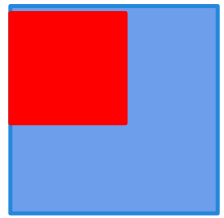
- Bitmap.createBitmap(...) to create a mutable Bitmap
- Pass it to BitmapDrawable(Resources, Bitmap)

LayerDrawable

Can layer Drawables on top of one another

- Create a LayerDrawable file - /res/drawable/composite.xml

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/large_blue_image" />
    <item android:drawable="@drawable/small_red_image" />
</layer-list>
```



- Now you can refer to R.id.composite

TransitionDrawable

A TransitionDrawable is a (2-layer) LayerDrawable that has smooth transition from showing one layer to another.

- Define TransitionDrawable in XML -

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:drawable="@drawable/image1" />
  <item android:drawable="@drawable/image2" />
</transition>
```

- Use startTransition() method to start transition on the TransitionDrawable

```
ImageView image = (ImageView) findViewById(R.id.image);
TransitionDrawable drawable = (TransitionDrawable) image.getDrawable();
drawable.startTransition(500);
```

StateListDrawable

A selector with a drawable defined for each state

- pressed, focused, hovered, selected, checkable, checked, enabled, activated, window_focused
- Items are traversed top to bottom
  - Often there is a default drawable at the bottom

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/pressed_image" android:
    state_pressed="true">
    <item android:drawable="@drawable/default_image" >
</selector>
```

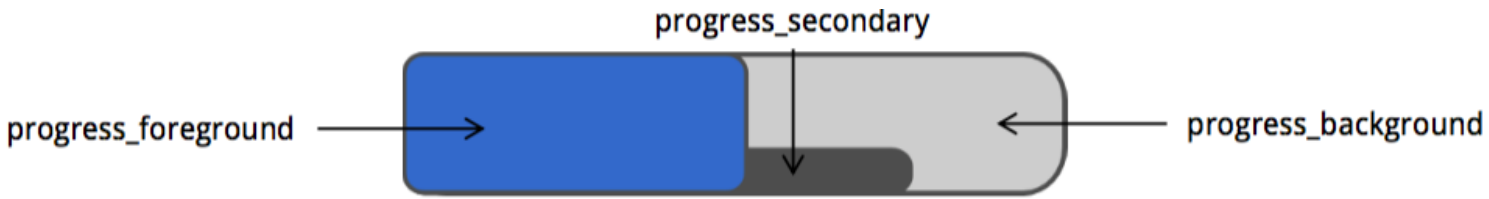
ClipDrawable

One Drawable that clips another

- Good for revealing a Drawable
  - Progress Bar (Horizontal), Filled Battery (Vertical), etc.
- Drawable is revealed through calls to setLevel().
  - Fully clipped at 0
  - Fully revealed at 10000

ClipDrawable

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@android:id/background"
        android:drawable="@drawable/progress_background" />
    <item android:id="@android:id/progress_secondary">
        <clip>
            <shape>
                <corners android:radius="4dp" />
                <solid android:color="#666666" />
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip android:drawable="@drawable/progress_foreground" />
    </item>
</layer-list>
```



ShapeDrawable

XML definitions of primitive geometric objects

- Used like a normal Drawable
- Automatically adjust to the correct size
- Rectangle (default), oval, line, ring
- Has borders, gradients, colors, etc.

/res/drawable/blue_rect.xml

```
<shape xmlns:android="http//schemas.android.com/apk/res/android"
        android:shape="rectangle">
        <solid android:color="#00000f" />
</shape>
```

/drawable vs. /mipmap folders

Android Studio (since 1.1) now creates the /mipmap folders by default.

- **/mipmap-*/** folders ← launcher icons only
- **/drawable-*/** folders ← all other drawable resources
  - you will need to create the folders yourself

Some reference reading:

- Managing Launcher Icons as mipmap Resources
- Goodbye launcher drawables, hello mipmaps!

Demo

- Walk through SampleDrawables

Break

Hands-on ShapeDrawable

Create a ShapeDrawable

1. Define shape
2. Define stroke
3. Define gradient
4. Define corners (for rectangle shape)
5. Use it as a button background

9-Patch Drawable

9-Patch Drawable

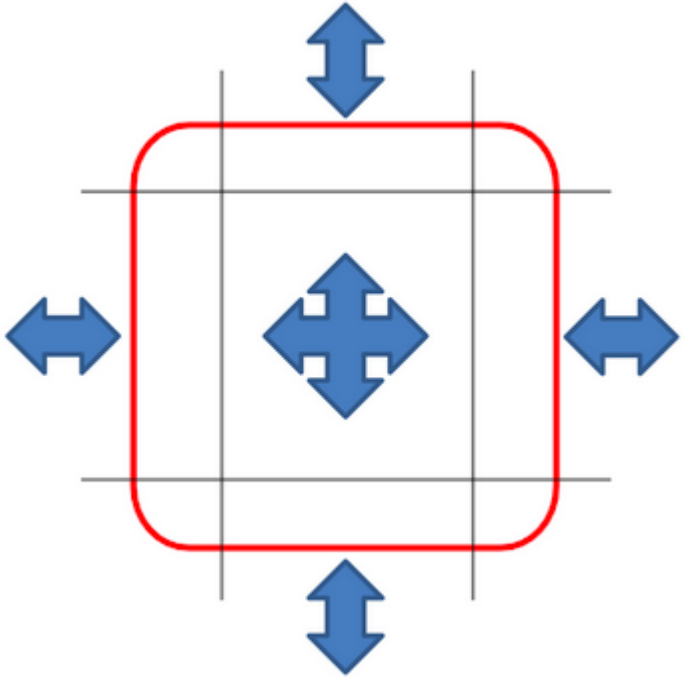
What is a 9-Patch Drawable?

- Really just a special PNG image
  - It defines stretch and content areas in a transparent border
- Android will scale the 9-patch to fit the space
- 9-patch drawables use a special ".9" pre-extension
  - Allows them to be identified and processed by the system
  - Looks like myninepatch.9.png

NinePatchDrawable

A formatted stretchable bitmap

- Corners aren't scaled
- Edges scaled in one axis
- Middle is scaled both axis



9-Patch Bounds

Stretch
- Single pixel black lines across LEFT and TOP edge
- Specifies how the content is carved up to stretch the 9-patch

Content
- Single pixel black lines across RIGHT and BOTTOM edge
- Specifies how the content fits within the 9-patch

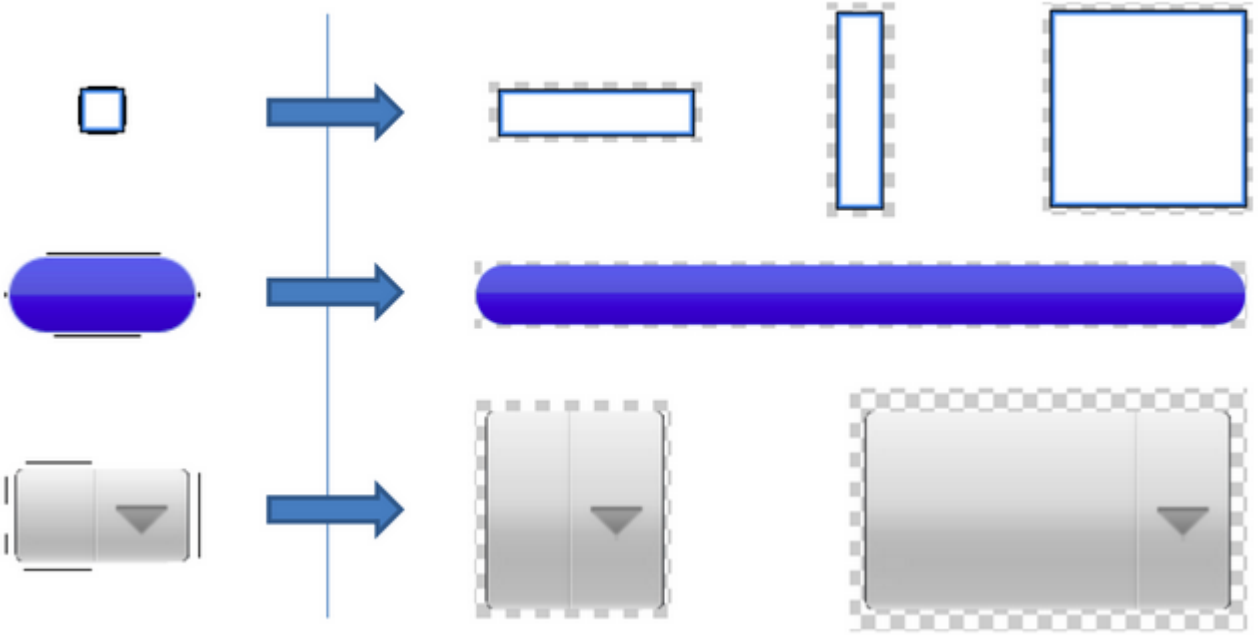
Optical (Introduced in 4.3)
- Single pixel red lines across RIGHT and BOTTOM edge
- Allows for alignment to "optical" bounds rather than "clip" bounds
- Really just helps align the background 9-patches to neighbors
- The android:layoutMode for a parent needs to be set to "optical"

9-Patch Bounds

drawable.png

drawable.9.png

NinePatchDrawable



Creating 9-patch drawables

You can create them using any image editor

However, you probably wont...

2 good tools for creating 9-patch drawables

1. Draw9Patch - In the SDK tools directory
2. Android Asset Studio

http://android-ui-utils.googlecode.com/hg/asset-studio/dist/nine-patches.html

Using 9-patch drawables

A 9-patch is just a drawable so treat it as such

- Use it where the size of the view is variable but the edges have special requirements for size and clarity
- Use it where a drawable would be used
  - Usually as a background image
  - Almost always used with a Selector
- Use the "Show layout bounds" developer option to see the effect of a 9-patch live

Hands-on: Nine Patch

- Walk through SampleDrawables (9-patch) section
- Demo - Draw9Patch tool
- Demo - use Android Assets Studio to create a nine-patch png

Break

Resources

Strings

String Resource Types

- String Resource
  - Used for a single string
- String Array Resource
  - Used for an array of strings
- Quantity Strings
  - Used for handling plurals

String Resources

- *Define the strings in strings.xml*

```
<resources>
    <string name="app_name">WonderMoose</string>
    <string name="app_tagline">A moose on a mission!</string>
</resources>
```

- Reference the string in code

```
String appName = getResources().getString(R.string.app_name);
```

String Array Resources

```
<resources>
        <string-array name="cat_array">
                <item>@string/cat_bobtail</item>
                <item>@string/cat_cymric</item
                <item>@string/cat_siamese</item>
                <item>@string/cat_toyger</item>
        </string-array>
</resources>
```

```
String[] cats = getResources().getStringArray(R.array.cat_array);
```

String Plural Resources

```
<resources>
    <plurals name="ravens">
        <item quantity="zero">No</item>
        <item quantity="one">One</item>
        <item quantity="two">Pair</item>
        <item quantity="many">Unkindness</item>
    </plurals>
</resources>
```

```
String ravens = getResources().getQuantityString(R.plurals.ravens, 99);
```

Accessing Resources

Project Resources

- /assets - An indexable folder for storing assets
- /res - An ID referenceable folder for storing resources

Project Resources

The typical Android resource structure

res/
- anim/
- animator/
- color/
- drawable/
- layout/
- menu/
- raw/
- values/
- xml/

String Resource Example

- /res/values/company_strings.xml
  <string name="company_name">XYZ Corp</string>
  <string name="tagline">Your business is our business.
  </string>

- /res/values/product_strings.xml
  <string name="product_name">Product X</string>
  <string name="test">Test</string>

What Android Creates

R.java

```
public static final class string {
    public static final int company_name=0x7f0b0000;
    public static final int product_name=0x7f0b0001;
    public static final int tagline=0x7f0b0002;
    public static final int test=0x7f0b0003;
}
```

Accessing Resources in XML

Resource Reference Syntax

@[<package_name>:]<resource_type>/<resource_name>

resource_type:

- drawable
- id
- layout
- string
- attr

Accessing Resources in Code

– Code Reference Syntax

[<package_name>.]R.<resource_type>.<resource_name>

Example:

ImageView iv1 = (ImageView) this.findViewById(R.id.imageView1);

How Android Finds Resources

- Scans all the files in the /res folder
- Aggregates into a R.java definition file
  - File names and conventions don't really matter beyond the normal limits (no hyphens, etc.)
- More specific resources are used if available
  - Specificity is defined by Resource Qualifiers

Raw Resources

Raw resources are uncooked files

- Located under res/raw/
- Not compiled
- Moved to the app package as they are
- Referenced via R.raw.*

Examples: audio, video, text files

Assets - not really resources

- Located under **/assets**, not part of **/res,** allow subfolders
- Do not generate IDs in R.java
- Use AssetManager to access the files

```
String getStringFromAssetFile(Activity activity) {
    AssetManager assetManager = activity.getAssets();
    INputStream inputStream = assetManager.open("files/test.txt");
    String string = convertStreamToSTring(inputStream);
    inputStream.close();
    return string;
}
```

/raw vs. /assets - which to use?

Both are uncompiled

- /assets allows subfolders
- /raw has R.java ids, slower access so if you have a db, put it under /assets

Demo

- Walk through SampleAssets

Upcoming

- We will cover Styles & Themes next Monday (4/13)
- 4/20 - Homework 1 due, no late homework accepted

Appendix

- Android Assets Studio

Making buttons:

- http://dabuttonfactory.com/
- http://angrytools.com/android/button/