Android Graphics

Topics

- Android Graphics
- Drawables
  - Using an image saved in your project resources
  - Using an XML file that defines the Drawable properties
- ShapeDrawable

Android Graphics

Android Graphics Support

- Android graphics are powered by
  - A custom 2D graphics library
  - OpenGL ES 1.0 for high performance 3D graphics.
- The most common 2D graphics APIs can be found in the drawable package.
- OpenGL APIs are available from the Khronos OpenGL ES package, plus some Android OpenGL utilities.

Drawables

What is a Drawable?

- A Drawable is a general abstraction for "something that can be drawn."
- Drawable class extends to define a variety of specific kinds of drawable graphics, including
  - BitmapDrawable, ShapeDrawable, PictureDrawable, LayerDrawable, and several more.
- You can also extend these to define your own custom Drawable objects that behave in unique ways.

Three ways to define & instantiate Drawables

- Using an image saved in your project resources
- Using an XML file that defines the Drawable properties
- Using the normal class constructors.

Drawables:
Using an image saved in your project resources

Creating from Resource Images

- A simple way to add graphics by referencing an image file from your project resources.
- Supported file types are
  - > PNG (preferred), JPG (acceptable) and GIF (discouraged).
- Preferred technique for application icons, logos, or other graphics such as those used in a game
- To use an image resource, just add your file to the res/drawable/ directory of your project
  - > From there, you can reference it from your code or your XML layout using a resource ID, which is the file name without the file type extension

Build an ImageView that uses an image from drawable resources and add it to the layout

```
LinearLayout mLinearLayout;

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Create a LinearLayout in which to add the ImageView
    mLinearLayout = new LinearLayout(this);

    // Instantiate an ImageView and define its properties
    ImageView i = new ImageView(this);
    i.setImageResource(R.drawable.my_image);
    i.setAdjustViewBounds(true); // set the ImageView bounds to match the
                                 //  Drawable's dimensions
    i.setLayoutParams(new Gallery.LayoutParams(LayoutParams.WRAP_CONTENT,
                             LayoutParams.WRAP_CONTENT));

    // Add the ImageView to the layout and set the layout as the content view
    mLinearLayout.addView(i);
    setContentView(mLinearLayout);
}
```

Handle your image resource as a Drawable object

```
Resources res = mContext.getResources();
Drawable myImage = res.getDrawable(R.drawable.my_image);
```

Add a resource Drawable to an ImageView in the XML layout

```
<ImageView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:tint="#55ff0000"
  android:src="@drawable/my_image"/>
```

Drawables:
Using XML

Creating from Resource XML

- If there is a Drawable object that you'd like to create, which is not initially dependent on variables defined by your application code or user interaction, then defining the Drawable in XML is a good option.
- Even if you expect your Drawable to change its properties during the user's experience with your application, you should consider defining the object in XML, as you can always modify properties once it is instantiated.

Creating from Resource XML

- Once you've defined your Drawable in XML, save the file in the res/drawable/ directory of your project.
- Then, retrieve and instantiate the object by calling *Resources.getDrawable()*, passing it the resource ID of your XML file.

XML that defines a TransitionDrawable

- Let's assume below is saved as res/drawable/expand_collapse.xml.

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

XML that defines a TransitionDrawable

```
Resources res = mContext.getResources();

TransitionDrawable transition =  (TransitionDrawable)
                        res.getDrawable(R.drawable.expand_collapse);

ImageView image =
      (ImageView) findViewById(R.id.toggle_image);
image.setImageDrawable(transition);
```

The above code will instantiate the *TransitionDrawable* and set it as the content of an ImageView

ShapeDrawable

ShapeDrawable

- When you want to dynamically draw some two-dimensional graphics, a ShapeDrawable object will probably suit your needs.
- With a ShapeDrawable, you can programmatically draw primitive shapes and style them in any way imaginable.