Introduction

Android support two types of Animation

1. Tweened Animation

   Tweened animations are applied to Views which define a series of changes in position, size, rotation, and opacity that animate the View contents.

2. Frame-by-Frame animation

   This is traditional cell-based animations in which a different Drawable is displayed in each frame. Frame-by-Frame animations are displayed within a View, using its Canvas as a projection screen.

Tweened Animation

Tweened animations offer a simple way of implementing animation in Android; it provides depth, movement, or feedback to users at a minimal resource cost.
Using Tweened animations we apply a set of orientation, scale, position, and opacity changes in much less resource-intensive way than manually redrawing the Canvas to achieve similar effects.

Creating Tweened Animation

Tweened animations are created using the Animation class. The following list mentioned the available types of animation.

1. AlphaAnimation lets you animate a change in the View’s transparency (opacity or alpha blending).
2. RotateAnimation lets you spin the selected View canvas in the XY plane.
3. ScaleAnimation Allows you to zoom in to or out from the selected View.
4. TranslateAnimation lets you move the selected View around the screen (although it will only be drawn within its original bounds).

Android offers the AnimationSet class to group and configure animations to be run as a set. We can define the start time and duration of each animation used within a set to control the timing and order of the animation sequence.

Tweened Animation sample code snippet

Animation sample code snippet demonstrates the usage of animations and interpolators, it also adds an "AnimationListener" which listen animation events.

Creating animation resource file and load into the Activity

The animation can be defined via XML resource file in the directory “res/anim”. The XML file we created here is "highlevelanimation.xml". In XML file we can group animations and define their start time, duration and offset value of each animation.

highlevelanimation.xml

```
...

        <?xml version="1.0" encoding="utf-8"?>
        <set xmlns:android="http://schemas.android.com/apk/res/android"
```

```
                android:shareInterpolator="true">

        <rotate         android:fromDegrees="0"
                                android:toDegrees="360"
                                android:duration="5000"
                                android:pivotX="50%"
                                android:pivotY="90%"
                                android:startOffset="0">
        </rotate>

        </set>
...
```

Loading resource file via Animation.Util is as follows:

```
...

        Animation animation1 =
AnimationUtils.loadAnimation(context,R.anim.highlevelanimation);

...
```

Using AnimationListener

We can apply animation to a view via view.startAnimation(Animation) by passing the AnimationSet as a parameter. By default AnimationSet will run once, unless we use setRepeatModel(int) or setRepeatCount(int). Applying animation set is as follows,

```
...

        View animatedView1 = findViewById(R.id.rotatetext);
        animatedView1.startAnimation(animation1);

...
```

After a tweened animation finishes the view returns to its original state. To change the view by reacting on Animation we need to implement "Animation.AnimationListener". This listener is notified when an animation starts, ends or repeats:

```
...

import android.view.animation.Animation.AnimationListener;

        public class AnimationActivity extends Activity implements AnimationListener{

        public void onAnimationStart(Animation animation) {
        }
        public void onAnimationEnd(Animation animation) {
```

```
        }
        public void onAnimationRepeat(Animation animation) {
        }
}

...
```

Sample Code for Tweened Animation

main.xml

```
...

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:id="@+id/layout"
    android:layout_height="fill_parent"
    android:baselineAligned="false"
    android:orientation="vertical" >

        <TextView  android:id="@+id/rotatetext"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="Will be rotated."
        />
        <Button     android:id="@+id/button_1"
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"
                     android:text="Rotate"
                     android:onClick="startAnimation"/>

        <TextView android:id="@+id/scrolltext"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:lines="1"
                    android:text="Will be scrolled to the left."
                    android:scrollHorizontally="true" />

        <Button   android:id="@+id/button_2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Animate to left + bounce"
                    android:onClick="startAnimation"/>
```

```
        <TextView android:id="@+id/fadeout"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:lines="1"
                    android:text="Will be faded out / in"
                    android:scrollHorizontally="true" />

        <Button   android:id="@+id/button_3"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Fade"
                    android:onClick="startAnimation"/>

        <Button   android:id="@+id/button_4"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Add / Remove component"
                   android:onClick="startAnimation"/>

</LinearLayout>


...
```

highlevelanimation.xml

```
...

<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
        android:shareInterpolator="true">
        <rotate android:fromDegrees="0" android:toDegrees="360"
                 android:duration="5000" android:pivotX="50%" android:pivotY="90%"
                 android:startOffset="0">
        </rotate>
</set>


...
```

AnimationActivity class

```
...

package com.app.animation;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.graphics.Paint;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.view.animation.Animation.AnimationListener;
import android.view.animation.AnimationUtils;
import android.view.animation.BounceInterpolator;
import android.view.animation.LayoutAnimationController;
import android.view.animation.TranslateAnimation;
import android.widget.TextView;
import android.widget.Toast;

public class AnimationActivity extends Activity implements AnimationListener{

        private TextView animatedView3;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void startAnimation(View view) {

                switch (view.getId()) {

                case R.id.button_1:
                        // Show how to load an animation from XML
                        Animation animation1 =
AnimationUtils.loadAnimation(this,R.anim.highlevelanimation);
                        animation1.setAnimationListener(this);
                        View animatedView1 = findViewById(R.id.rotatetext);
                        animatedView1.startAnimation(animation1);
                        break;
                case R.id.button_2:
                        // Shows how to define a animation via code
                        // Also use an Interpolator (BounceInterpolator)

                        Paint paint = new Paint();
```

```
                        TextView animatedView2 = (TextView)
findViewById(R.id.scrolltext);
                        float measureTextCenter =
paint.measureText(animatedView2.getText().toString());
                        Animation animation2 = new
TranslateAnimation(0f,-measureTextCenter, 0.0f, 0.0f);
                        animation2.setDuration(5000);
                        animation2.setInterpolator(new BounceInterpolator());
                        animatedView2.startAnimation(animation2);
                        break;
                case R.id.button_3:
                        // Demonstrate fading and adding an AnimationListener
                        animatedView3 = (TextView) findViewById(R.id.fadeout);
                        float from = 1.0f;
                        float to = 0.0f;
                        if (animatedView3.getVisibility() == View.INVISIBLE) {
                                from = to;
                                to = 1.0f;
                        }
                        Animation animation3 = new AlphaAnimation(from, to);
                        animation3.setDuration(5000);
                        animation3.setAnimationListener(this);
                        animatedView3.startAnimation(animation3);
                        break;
                case R.id.button_4:
                        // Demonstrate LayoutAnimation
                        ViewGroup layout = (ViewGroup) findViewById(R.id.layout);
                        Animation animation4 = new AlphaAnimation(0.0f, 1.0f);
                        animation4.setDuration(5000);
                        LayoutAnimationController controller = new
LayoutAnimationController(animation4, 0);
                        layout.setLayoutAnimation(controller);
                        View button = findViewById(R.id.button_3);
                        try{
                                        if(button!=null)
                                                layout.removeView(button);
                        }catch(Exception ex){
                                ex.printStackTrace();
                        }
                        break;
                default:
                        break;
                }
```

```
        }

        public void onAnimationStart(Animation animation) {
                Toast.makeText(this, "Animation started", Toast.LENGTH_SHORT).show();
        }
        public void onAnimationEnd(Animation animation) {
                Toast.makeText(this, "Animation ended", Toast.LENGTH_SHORT).show();
        }
        public void onAnimationRepeat(Animation animation) {
                Toast.makeText(this, "Animation rep", Toast.LENGTH_SHORT).show();
        }
}

...
```

Frame-by-Frame Animations

Frame-by-Frame animations shows different drawables in View, it's an old way of show animation by choosing each frame. Here animation is limited to the original size of the view.

The AnimationDrawable class is used to create a new frame-by-frame animation presented as a Drawable resource. We can define Animation Drawable resource as an external resource by placing resources in res/drawable folder using XML.
Use the <animation-list> tag to group a collection of <item> nodes, each of which uses a drawable attribute to define an image to display, and a duration attribute to specify the time (in milliseconds) to display it.

Sample Code for Frame-by-Frame Animation

The animation can be defined via XML resource file in the directory "res/drawable", simpleanimation.xml is as follows,

simpleanimation.xml

```
...

<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="false">

 <item android:drawable="@drawable/image_1" android:duration="100" />
<item android:drawable="@drawable/image_2" android:duration="100" />
<item android:drawable="@drawable/image_3" android:duration="100" />
<item android:drawable="@drawable/image_4" android:duration="100" />
<item android:drawable="@drawable/image_5" android:duration="100" />

</animation-list>
```

```
...
```

main.xml

```
...

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >

        <ImageView
                android:id="@+id/simple_anim"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:layout_centerHorizontal="true"
        />
        <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Frame Animation!!!"
        />

</LinearLayout>

...
```

FramebyFrameAnimation class

```
...

package com.app.animation;

import android.app.Activity;
import android.os.Bundle;
import android.graphics.drawable.AnimationDrawable;
import android.widget.ImageView;
import java.util.Timer;
import java.util.TimerTask;

public class FramebyFrameAnimation extends Activity {
        /** Called when the activity is first created. */
```

```
ImageView img = null;

@Override
public void onCreate(Bundle icicle) {

        super.onCreate(icicle);
        setContentView(R.layout.main);
        img = (ImageView) findViewById(R.id.simple_anim);
        img.setBackgroundResource(R.drawable.simpleanimation);
        MyAnimationRoutine mar = new MyAnimationRoutine();
        MyAnimationRoutine2 mar2 = new MyAnimationRoutine2();
        Timer t = new Timer(false);
        t.schedule(mar, 100);
        Timer t2 = new Timer(false);
        t2.schedule(mar2, 5000);
}

class MyAnimationRoutine extends TimerTask {
        MyAnimationRoutine() {
        }

        public void run() {
                ImageView img = (ImageView) findViewById(R.id.simple_anim);
                // Get the background, which has been compiled to an
                // AnimationDrawable object.
                AnimationDrawable frameAnimation = (AnimationDrawable) img
                                .getBackground();
                // Start the animation (looped playback by default).
                frameAnimation.start();
        }
}

class MyAnimationRoutine2 extends TimerTask {
        MyAnimationRoutine2() {
        }

        public void run() {
                ImageView img = (ImageView) findViewById(R.id.simple_anim);
                // Get the background, which has been compiled to an
                // AnimationDrawable object.
                AnimationDrawable frameAnimation = (AnimationDrawable) img
                                .getBackground();
```

```
                        // stop the animation (looped playback by default).
                        frameAnimation.stop();
                }
        }

}


...
```

ref:http://developer.samsung.com/android/technical-docs/How-to-use-Animation-in-Android