Animation

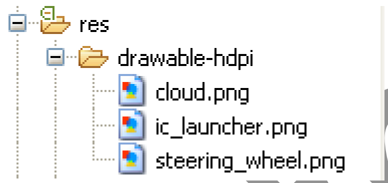
Step 1: Start an Android Project

You will also need to choose a minimum SDK of 11, so make sure your project Manifest indicates an appropriate level as in the following excerpt:

```
<uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="21" />
```

Step 2: Add Images to the Project

We will be creating a few drawable files in XML for the animation, but will also be using a couple of PNG images. We will use the cloud and steering wheel images.



```
res
  drawable-hdpi
    cloud.png
    ic_launcher.png
    steering_wheel.png
```

Step 3: Create the Drawables

In your application drawables folder(s), create the first new file by selecting the folder and choosing "File", "New", "File". Enter "sun.xml" as the file name. In the new file, enter the following code to define a sun shape:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:dither="true"
    android:shape="oval">


  <gradient
```

```
        android:startColor="#FFFFCC00"
        android:endColor="#FFFF6600"
        android:gradientRadius="150"
        android:type="radial"
        android:useLevel="false"
        />
    <size android:width="100dp"
        android:height="100dp"/>
</shape>
```

The drawable is an oval shape, with a radial gradient fill and specified size. When you finish entering code for each of the drawables files, save them and copy them into each drawables folder your app is using.

Create another new file in your app's drawables resources, this time naming it "ground.xml". Enter the following shape drawable.

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:dither="true"
    android:shape="rectangle">
<solid android:color="#339933"/>
</shape>
```

Create another drawables file, naming it "window.xml" and entering the following shape.

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#00000000"/>
    <stroke android:width="40dp"
        android:color="#CCCCCC"
        />
</shape>
```

Step 4: Design the Layout

Now we can include our drawables in the layout. We will be using Image Views, with which we need to supply a *content description* String describing the image in each case. In preparation for this, open your app's "res/values/strings.xml" file and add the following values.

```
<string name="wheel">Steering Wheel</string>
<string name="ground">The Ground</string>
<string name="window">Window Frame</string>
<string name="sun">The Sun</string>
<string name="cloud">A Cloud</string>
```

Now open your layout file. Replace the contents with the following Relative Layout.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#66CCFF"
   android:id="@+id/car_layout"
   tools:context="com.nnk.prj25.MainActivity" ></RelativeLayout>
```

Notice that we apply a background color to the layout - we also include an ID attribute for referring to the layout in Java.

First inside the layout, add an Image View to display the sun shape we created.

```
<ImageView
      android:id="@+id/sun"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
        android:contentDescription="@string/sun"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="45dp"
        android:src="@drawable/sun" />
```

We include an ID attribute so that we can refer to the View when animating it. We also refer to one of the content description Strings we created and list the name of the drawable file as source *src* attribute. We also position the View within the layout.

Next add two more Image Views for the clouds.

```
<ImageView
        android:id="@+id/cloud1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/cloud"
        android:paddingLeft="170dp"
        android:paddingTop="70dp"
        android:src="@drawable/cloud" />
```

```
    <ImageView
        android:id="@+id/cloud2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/cloud"
        android:paddingLeft="200dp"
        android:paddingTop="90dp"
        android:src="@drawable/cloud" />
```

Next add the ground.

```
<ImageView
        android:id="@+id/ground"
        android:layout_width="fill_parent"
        android:layout_height="200dp"
        android:layout_alignParentBottom="true"
        android:contentDescription="@string/ground"
        android:padding="40dp"
        android:src="@drawable/ground" />
```

The ground is aligned to the bottom of the parent View.

Notice that we also include padding on the bottom - this is to accommodate the **window** frame we will add next.

```
<ImageView
        android:id="@+id/window"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:contentDescription="@string/window"
        android:src="@drawable/window" />
```

The window shape uses a 40dp wide stroke, which is why we added 40dp of padding on the ground shape.

Each Image View added will be displayed on top of previous items in terms of the z-index so we add the steering wheel last:

```
<ImageView
        android:id="@+id/wheel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:contentDescription="@string/wheel"
        android:padding="3dp"
        android:src="@drawable/steering_wheel" />
```

Step 5: Define the Wheel Turning Animation

In your app's "res" folder, create a new sub-folder by selecting "res" and choosing "File", "New", "Folder". Enter "animator" as the folder name.

We will be adding two XML files to this folder, defining the wheel turning and sun moving animations.

Start by creating a new file in "animator" and naming it "wheel_spin.xml".

```
<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/accelerate_decelerate_interpolat
or"
    android:ordering="sequentially">
    …….
</set>
```

Inside the ***set*** we will define the details of the animation.

In the opening ***set*** tag we specify an **interpolator**, in this case the ***accelerate decelarate*** interpolator so that the animation will speed up at the start and slow down at the end.

We also specify ordering, which will not actually have an effect in this case as we are only going to define one animator within the set. If you have more than one, you can use this attribute to carry out the animations at the same time or sequentially.

Inside the set, include an animator to make the wheel turn.

```
<objectAnimator
        android:duration="3000"
        android:propertyName="rotation"
        android:repeatCount="infinite"
```

```
    android:repeatMode="reverse"
    android:valueTo="180"
    android:valueType="floatType">
</objectAnimator>
```

This is an Object Animator, which first defines a duration and property to animate over this period.

The *valueTo* attribute indicates 180 degrees, which is how much the wheel will turn over the duration. When the animation completes, we set it to reverse and then repeat continuously.

Step 6: Apply the Wheel Turning Animation

Now let's turn to the Activity class.

```
        ImageView wheel = (ImageView)findViewById(R.id.wheel);
        AnimatorSet wheelSet = (AnimatorSet)
AnimatorInflater.loadAnimator(this, R.animator.wheel_spin);
      wheelSet.setTarget(wheel);
      wheelSet.start();
```

First get references to the wheel shape using the ID we gave its ImageView.

```
      ImageView wheel = (ImageView)findViewById(R.id.wheel);
```

Now create an Animator Set to load the animation we defined.

```
      AnimatorSet wheelSet = (AnimatorSet)
AnimatorInflater.loadAnimator (this, R.animator.wheel_spin);
```

AnimatorSet is a sub class of **android.animation.Animator** class

This class plays a set of Animator objects in the specified order. Animations can be set up to play together, in sequence, or after a specified delay.

android.animation.AnimatorInflater

This class is used to instantiate animator XML files into Animator objects.

public static Animator loadAnimator (Context context, int id)

This method Loads an **Animator** object from a resource.

public void setTarget (Object target)

Sets the target object for all current child animations of this AnimatorSet.

```
wheelSet.setTarget(wheel);
```

public void start ( )

Starts this animation.

Step 7: Define the Sun Moving Animation

Let's make the sun move to create the impression that we are moving as a result of the steering. Create a new file in your "animator" folder, this time named "sun_swing.xml". Enter the following code.

```
<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/accelerate_decelerate_interpolat
or"
   android:ordering="sequentially">
```

```
    <objectAnimator
        android:duration="3000"
        android:propertyName="x"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:valueTo="-400"
        android:valueType="floatType" ></objectAnimator>

</set>
```

This time the Object Animator animates the *x* property, moving the object to the left, as the steering wheel turns us to the right. The duration and repeat properties are the same as the wheel turn animation, as this effect is intended to coincide with it.

Step 8: Apply the Sun Moving Animation

Back in the Activity class *onCreate* method, apply this new animation to the sun View using the same technique as before.

```
//get the sun view
        ImageView sun = (ImageView)findViewById(R.id.sun);
//load the sun movement animation
        AnimatorSet sunSet = (AnimatorSet)
AnimatorInflater.loadAnimator(this, R.animator.sun_swing);
//set the view as target
        sunSet.setTarget(sun);
//start the animation
        sunSet.start();
```

Step 9: Implement a Sky Darkening Animation

Let's now explore creating animations in Java from the Activity class. We will make the sky go slightly darker when the sun moves out of view. Still inside *onCreate*, instantiate a Value Animator.

```
            ValueAnimator skyAnim =
ObjectAnimator.ofInt(findViewById(R.id.car_layout),
"backgroundColor",Color.rgb(0x66, 0xcc, 0xff), Color.rgb(0x00, 0x66,
0x99));
```

We pass the ID of the layout element which has the background color applied to it, also specifying "**backgroundColor**" as the property we wish to animate. Finally, we specify colors to animate from and to, which are lighter and darker shades of blue.

ValueAnimator is the sub class of **android.animation.Animator** class

This class provides a simple timing engine for running animations which calculate animated values and set them on target objects.

ObjectAnimator is the sub class of **ValueAnimator**

public static ObjectAnimator ofInt ( )

ofInt is a pre-defined static method of ObjectAnimator class which Constructs and returns an ObjectAnimator that animates between int values.

android.graphics.Color
public static int rgb (int red, int green, int blue)

Return a color-int from red, green, blue components. These component values should be [0..255].

Set the duration and repeat properties to match the existing animations.

```
skyAnim.setDuration(3000);
skyAnim.setRepeatCount(ValueAnimator.INFINITE);
skyAnim.setRepeatMode(ValueAnimator.REVERSE);
```

Now set an evaluator to instruct the animator how to evaluate the passed color values.

```
skyAnim.setEvaluator(new ArgbEvaluator());
```

Start the animation.

```
skyAnim.start();
```

Step 10: Implement Two Cloud Moving Animations

Still in *onCreate*, instantiate an Object Animator for the first cloud View, specifying the *x* property and a value to move it to.

```
ObjectAnimator cloudAnim =
ObjectAnimator.ofFloat(findViewById(R.id.cloud1), "x", -350);
```

public static ObjectAnimator.ofFloat(Object target, String propertyName, float... values)

Constructs and returns an ObjectAnimator that animates between float values.

Set the duration and repeat properties as before, and then start the animation.

```
cloudAnim.setDuration(3000);
cloudAnim.setRepeatCount(ValueAnimator.INFINITE);
cloudAnim.setRepeatMode(ValueAnimator.REVERSE);
cloudAnim.start();
```

Do the same for the second cloud, passing a slightly different value to move it along the *x* axis.

```
		ObjectAnimator cloudAnim2 = 
ObjectAnimator.ofFloat(findViewById(R.id.cloud2), "x", -300);
		cloudAnim2.setDuration(3000);
		cloudAnim2.setRepeatCount(ValueAnimator.INFINITE);
		cloudAnim2.setRepeatMode(ValueAnimator.REVERSE);
		cloudAnim2.start();
```

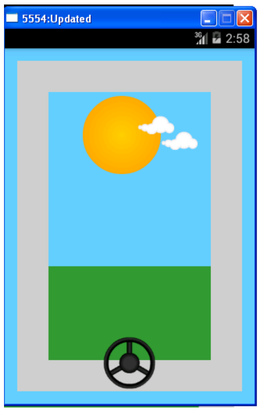
The difference in *x* value with an animation over the same duration will create the impression that the clouds are moving at slightly different speeds, one being closer than the other.

Java Code

```
import android.animation.AnimatorInflater;
import android.animation.AnimatorSet;
import android.animation.ArgbEvaluator;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.ImageView;

public class MainActivity extends Activity
{

	@Override
	protected void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		ImageView wheel = (ImageView)findViewById(R.id.wheel);
		AnimatorSet wheelSet = (AnimatorSet) 
AnimatorInflater.loadAnimator(this, R.animator.wheel_spin);
		wheelSet.setTarget(wheel);
		wheelSet.start();

		ImageView sun = (ImageView)findViewById(R.id.sun);
		AnimatorSet sunSet = (AnimatorSet) 
AnimatorInflater.loadAnimator(this, R.animator.sun_swing);
		sunSet.setTarget(sun);
		sunSet.start();

		ValueAnimator skyAnim = 
ObjectAnimator.ofInt(findViewById(R.id.car_layout), 
"backgroundColor",Color.rgb(0x66, 0xcc, 0xff), Color.rgb(0x00, 0x66, 
0x99));

		skyAnim.setDuration(3000);
		skyAnim.setRepeatCount(ValueAnimator.INFINITE);
		skyAnim.setRepeatMode(ValueAnimator.REVERSE);
		skyAnim.setEvaluator(new ArgbEvaluator());
		skyAnim.start();

		ObjectAnimator cloudAnim = 
ObjectAnimator.ofFloat(findViewById(R.id.cloud1), "x", -350);
		cloudAnim.setDuration(3000);
		cloudAnim.setRepeatCount(ValueAnimator.INFINITE);
		cloudAnim.setRepeatMode(ValueAnimator.REVERSE);
		cloudAnim.start();

		ObjectAnimator cloudAnim2 = 
ObjectAnimator.ofFloat(findViewById(R.id.cloud2), "x", -300);
		cloudAnim2.setDuration(3000);
		cloudAnim2.setRepeatCount(ValueAnimator.INFINITE);
		cloudAnim2.setRepeatMode(ValueAnimator.REVERSE);
```

```
                cloudAnim2.start();
        }
}
```

Output


5554:Updated
2:58