# 28. Animaciones

La plataforma Android proporciona cuatro tipos de animaciones:

- Imágenes GIF animadas. Los GIF animados son ficheros gráficos que contienen varios fotogramas (*frames*).
- Animaciones fotograma a fotograma. Mediante la clase `AnimationDrawable`, el programador suministra los fotogramas y las transiciones entre ellos.
- Animaciones de interpolación (*tweening*). Estas animaciones pueden resolverse con código XML y se aplican a cualquier vista.
- Animaciones con la biblioteca OPEN GL ES.

En esta unidad nos vamos a centrar en las animaciones de interpolación, de las que hay cuatro tipos:

- Animación `alpha` para cambiar la transparencia de una vista.
- Animación `rotate` para rotar una vista un cierto ángulo alrededor de un eje o punto de pivote.
- Animación `scale` para agrandar o disminuir una vista según el eje X e Y.
- Animación `translate` para desplazar una vista a lo largo del eje X e Y.

Las animaciones de interpolación se pueden definir tanto en XML como en Java. Por ejemplo, el proyecto de esta unidad, `CCC28`, añade una animación `scale` en XML para la figura inicial de nuestro solitario.

Para crear una animación `scale` en XML, añadiremos una carpeta de nombre `anim` a la carpeta `res` de nuestro proyecto. Dentro de esta carpeta, añadiremos un archivo de nombre `initial.xml` como el siguiente:

`/res/anim/initial.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="4000"
    android:fromXScale="1.0"
    android:fromYScale="0.2"
    android:interpolator="@android:anim/bounce_interpolator"
    android:toXScale="1.0"
    android:toYScale="1.0" />
```

El atributo `android:fromXScale` especifica el valor inicial del escalado según el eje x. Por su lado, `android:toXScale` especifica el valor final. Como queremos conseguir un efecto de caída de telón, tanto el primero como el segundo valen 1.0, de tal forma que la vista no se modifique horizontalmente. Sin embargo, `android:fromYScale` se iguala a 0.2. El atributo `android:duration` especifica la duración de la animación. Añadiremos un interpolador para conseguir el efecto de rebote.

La actividad `Initial`, en su método `onCreate()`, se encarga de conseguir una referencia al `ImageView` de su pantalla y arrancar la animación:

/src/Initial.java

```
package es.uam.eps.android.ccc28;

...

public class Initial extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.initial);

        Animation animation = AnimationUtils.loadAnimation(this, R.anim.initial);
        ImageView imageView = (ImageView) findViewById(R.id.initial);
        imageView.startAnimation(animation);
    }

    ...
}
```

Para ello se utiliza la clase de ayuda `AnimationUtils`. Primero hemos de cargar la animación con el método `loadAnimation()`, que tiene dos argumentos: el contexto y el identificador del fichero XML donde se especifica la animación. Finalmente, la animación se arranca desde la vista que queremos animar llamando al método `startAnimation()`.

Las animaciones se pueden combinar en XML mediante un elemento `<set>`. Por ejemplo, en el siguiente fichero se especifica una animación de escala seguida por una de rotación. La segunda empieza cuando acaba la primera pues el atributo `android:startOffSet` de la segunda animación se iguala a la duración de la primera (5000 milisegundos). La primera animación dobla el tamaño de la vista según el eje X y lo triplica según el eje Y. La segunda animación rota la vista alrededor de su punto medio (`android:pivotY="50%"`):

/res/anim/initial.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
         android:fromXScale="1.0"
         android:toXScale="2.0"
         android:fromYScale="1.0"
         android:toYScale="3.0"
         android:pivotX="50%"
         android:pivotY="50%"
         android:duration="5000"/>
    <rotate
         android:fromDegrees="0"
         android:toDegrees="360"
         android:pivotX="50%"
         android:pivotY="50%"
         android:duration="4000"
         android:startOffset="5000"/>
</set>
```

El atributo `android:fillAfter` permite especificar si se quiere que la vista vuelva o no a su estado inicial. Si le asignamos el valor `true`, la vista no volverá a su estado inicial.