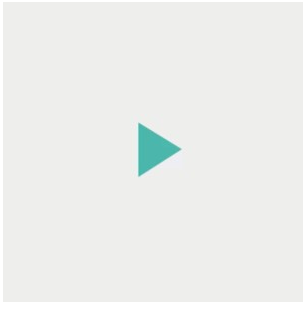
为可绘制图形添加动画效果



在某些情况下，图片需要在屏幕上呈现动画效果。如果您希望显示由多张图片组成的自定义加载动画，或者希望一个图标在用户执行操作后变为另一个图标，这种做法就非常实用。Android 提供了两个选项供您为可绘制对象添加动画效果。

第一个选项是使用 AnimationDrawable (#AnimationDrawable)。使用该选项，您可以指定多个静态可绘制对象文件 (/guide/topics/graphics/2d-graphics)（每次展示一个）来创建动画。第二个选项是使用 AnimatedVectorDrawable (#AnimVector)。使用该选项，您可以为矢量可绘制对象 (/guide/topics/graphics/vector-drawable-resources)的属性添加动画效果。

使用 AnimationDrawable

要为 **Drawables** (/reference/android/graphics/drawable/Drawable) 添加动画效果，一种方法是接连加载一系列可绘制资源以创建动画。从某种意义上来说，这是一种传统动画，使用一系列不同的图片创建而成，然后像一卷胶卷一样按顺序播放。**AnimationDrawable** (/reference/android/graphics/drawable/AnimationDrawable) 类是可绘制动画的基础。

虽然您可以使用 **AnimationDrawable** (/reference/android/graphics/drawable/AnimationDrawable) 类 API 在代码中定义动画帧，但使用单个 XML 文件（其中列出构成动画的各个帧）可更轻松地完成此操作。此类动画的 XML 文件位于 Android 项目的 **res/drawable/** 目录中。在这种情况下，具体指令是每个动画帧的顺序和时长。

XML 文件包含一个 **<animation-list>** 元素（用作根节点）和一系列子 **<item>** 节点（每个节点定义一个帧）：帧和帧时长的可绘制资源。以下是可绘制动画的 XML 文件示例：

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>
```

此动画仅运行 3 帧。通过将列表的 **android:oneshot** 属性设置为 ***true***，此动画仅循环一次，然后便停止并保持在最后一帧。如果设置为 ***false***，则动画会一直循环。将此 XML 文件另存为 **rocket_thrust.xml**（位于项目的 **res/drawable/** 目录中）后，可将其作为背景图片添加到视图，然后进行调用以播放。在以下示例 Activity 中，该动画添加到了 **ImageView** (/reference/android/widget/ImageView) 中，然后在用户轻触屏幕时呈现动画效果：

KOTLIN (#KOTLIN)JAVA

```
AnimationDrawable rocketAnimation;

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

  ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
  rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
  rocketAnimation = (AnimationDrawable) rocketImage.getBackground();

  rocketImage.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        rocketAnimation.start();
```

```
        }
    });
}
```

请务必注意，对 **`AnimationDrawable`** (/reference/android/graphics/drawable/AnimationDrawable) 调用的 **`start()`** 方法不能在 Activity 的 **`onCreate()`** 方法期间调用，因为 **`AnimationDrawable`** (/reference/android/graphics/drawable/AnimationDrawable) 尚未完全附加到窗口。如果您想立即播放动画而无需互动，那么您可能需要从 Activity 中的 **`onStart()`** (/reference/android/app/Activity#onStart()) 方法进行调用，该方法会在 Android 在屏幕上呈现视图时调用。

如需详细了解 XML 语法、可用标记和属性，请参阅动画资源 (/guide/topics/resources/animation-resource)。

使用 AnimatedVectorDrawable

矢量可绘制对象 (/guide/topics/graphics/vector-drawable-resources)是一种无需像素化或进行模糊处理即可缩放的可绘制对象。借助 **`AnimatedVectorDrawable`** (/reference/android/graphics/drawable/AnimatedVectorDrawable) 类（以及用于实现向后兼容的 **`AnimatedVectorDrawableCompat`** (/reference/androidx/vectordrawable/graphics/drawable/AnimatedVectorDrawableCompat)），您可以为矢量可绘制对象的属性添加动画效果，例如旋转或更改路径数据以将其变为其他图片。

您通常在三个 XML 文件中定义添加动画效果之后的矢量可绘制对象：

- 一个矢量可绘制对象，其中 **`<vector>`** 元素位于 **`res/drawable/`**
- 一个添加动画效果之后的矢量可绘制对象，其中 **`<animated-vector>`** 位于 **`res/drawable/`**
- 一个或多个对象 Animator，其中 **`<objectAnimator>`** 元素位于 **`res/animator/`**

添加动画效果之后的矢量可绘制对象可以为 **`<group>`** 和 **`<path>`** 元素的属性添加动画效果。**`<group>`** 元素定义一组路径或子组，而 **`<path>`** 元素定义要绘制的路径。

当您定义要添加动画效果的矢量可绘制对象时，请使用 **`android:name`** 属性，为各个组和路径指定唯一名称，以便您可以从 Animator 定义中引用它们。例如：

res/drawable/vectordrawable.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600">
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

添加动画效果之后的矢量可绘制对象定义会按矢量可绘制对象中各个组和路径的名称对其进行引用：

res/drawable/animatorvectordrawable.xml

```
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
  android:drawable="@drawable/vectordrawable" >
    <target
        android:name="rotationGroup"
        android:animation="@animator/rotation" />
    <target
        android:name="v"
```

```
        android:animation="@animator/path_morph" />
</animated-vector>
```

动画定义表示 **`ObjectAnimator`** (/reference/android/animation/ObjectAnimator) 或 **`AnimatorSet`** (/reference/android/animation/AnimatorSet) 对象。以下示例中的第一个 Animator 会将目标组旋转 360 度：

res/animator/rotation.xml

```
<objectAnimator
    android:duration="6000"
    android:propertyName="rotation"
    android:valueFrom="0"
    android:valueTo="360" />
```

以下示例中的第二个 Animator 会将矢量可绘制对象的路径从一个形状变为另一个形状。这两个路径都必须兼容变形：它们必须具有相同数量的命令，并且对于每个命令，必须具有相同数量的参数。

res/animator/path_morph.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:duration="3000"
        android:propertyName="pathData"
        android:valueFrom="M300,70 l 0,-70 70,70 0,0   -70,70z"
        android:valueTo="M300,70 l 0,-70 70,0  0,140 -70,0 z"
        android:valueType="pathType" />
</set>
```

以下是生成的 **`AnimatedVectorDrawable`** (/reference/android/graphics/drawable/AnimatedVectorDrawable)：

如需了解详情，请参阅 **`AnimatedVectorDrawable`** (/reference/android/graphics/drawable/AnimatedVectorDrawable) 的 API 参考文档。

Last updated 2019-12-27 UTC.