动画资源

一个动画资源可以定义以下两种动画类型之一：

属性动画 (#Property)

通过使用 `Animator` (/reference/android/animation/Animator) 在设定的时间段内修改对象的属性值来创建动画。

视图动画 (#View)

使用视图动画框架可以创建两种类型的动画：

- 补间动画 (#Tween)：通过使用 `Animation` (/reference/android/view/animation/Animation) 对单张图片执行一系列转换来创建动画
- 帧动画 (#Frame)：通过使用 `AnimationDrawable` (/reference/android/graphics/drawable/AnimationDrawable) 按顺序显示一系列图片来创建动画。

属性动画

在 XML 中定义的动画，用于在设定的一段时间内修改目标对象的属性，例如背景颜色或 Alpha 值。

文件位置：

`res/animator/`*`filename`*`.xml`
该文件名将用作资源 ID。

编译后的资源数据类型：

指向 `ValueAnimator` (/reference/android/animation/ValueAnimator)、`ObjectAnimator` (/reference/android/animation/ObjectAnimator) 或 `AnimatorSet` (/reference/android/animation/AnimatorSet) 的资源指针。

资源引用：

在 Java 或 Kotlin 代码中：`R.animator.`*`filename`*
在 XML 中：`@[`*`package`*`:]animator/`*`filename`*

语法：

```
<set (#animator-set-element)
  android:ordering=["together" | "sequentially"]>

    <objectAnimator (#obj-animator-element)
        android:propertyName="string"
        android:duration="int"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=["repeat" | "reverse"]
        android:valueType=["intType" | "floatType"]/>

    <animator (#val-animator-element)
        android:duration="int"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=["repeat" | "reverse"]
        android:valueType=["intType" | "floatType"]/>

    <set (#animator-set-element)>
        ...
    </set>
```

```
</set>
```

该文件必须具有一个根元素，可以是 <set>、<objectAnimator> 或 <valueAnimator>。您可以将动画元素（包括其他 <set> 元素）组合到 <set> 元素中。

元素：

<set>

容纳其他动画元素（<objectAnimator>、<valueAnimator> 或其他 <set> 元素）的容器。代表 AnimatorSet (/reference/android/animation/AnimatorSet)。

您可以指定嵌套的 <set> 标记来将动画进一步组合在一起。每个 <set> 都可以定义自己的 ordering 属性。

属性：

android:ordering

关键字。指定此集合中动画的播放顺序。

值	说明
sequentially	依序播放此集合中的动画
together（默认）	同时播放此集合中的动画。

<objectAnimator>

在特定的一段时间内为对象的特定属性创建动画。代表 ObjectAnimator (/reference/android/animation/ObjectAnimator)。

属性：

android:propertyName

字符串。**必需。**要添加动画的对象属性，通过其名称引用。例如，您可以为 View 对象指定 "alpha" 或 "backgroundColor"。 但是，objectAnimator 元素不包含 target 属性，因此您无法在 XML 声明中设置要添加动画的对象。您必须通过调用 loadAnimator() (/reference/android/animation/AnimatorInflater#loadAnimator(android.content.Context, int)) 来扩充您的动画 XML 资源，然后调用 setTarget() (/reference/android/animation/ObjectAnimator#setTarget(java.lang.Object)) 来设置包含此属性的目标对象。

android:valueTo

浮点数、整数或颜色。**必需。**动画属性的结束值。颜色以六位十六进制数字表示（例如，#333333）。

android:valueFrom

浮点数、整数或颜色。动画属性的开始值。如果未指定，则动画将从属性的 get 方法获得的值开始。颜色以六位十六进制数字表示（例如，#333333）。

android:duration

整数。动画的时间，以毫秒为单位。默认为 300 毫秒。

android:startOffset

整数。调用 start() (/reference/android/animation/ObjectAnimator#start()) 后动画延迟的毫秒数。

android:repeatCount

整数。动画的重复次数。设为 "-1" 表示无限次重复，也可设为正整数。例如，值 "1" 表示动画在初次播放后重复播放一次，因此动画总共播放两次。默认值为 "0"，表示不重复。

android:repeatMode

整数。动画播放到结尾处的行为。android:repeatCount 必须设置为正整数或 "-1"，该属性才有效。设置为 "reverse" 可让动画在每次迭代时反向播放，设置为 "repeat" 则可让动画每次从头开始循环播放。

android:valueType

关键字。如果值为颜色，则不要指定此属性。动画框架会自动处理颜色值

值	说明
`intType`	指定动画值为整数
`floatType`（默认）	指定动画值为浮点数

`<animator>`

在指定的时间段内执行动画。代表 `ValueAnimator` (/reference/android/animation/ValueAnimator)。

属性:

`android:valueTo`

浮点数、整数或颜色。**必需**。动画的结束值。颜色以六位十六进制数字表示（例如，#333333）。

`android:valueFrom`

浮点数、整数或颜色。**必需**。动画的开始值。颜色以六位十六进制数字表示（例如，#333333）。

`android:duration`

整数。动画的时间，以毫秒为单位。默认为 300ms。

`android:startOffset`

整数。调用 `start()` (/reference/android/animation/ValueAnimator#start()) 后动画延迟的毫秒数。。

`android:repeatCount`

整数。动画的重复次数。设为 `"-1"` 表示无限次重复，也可设为正整数。例如，值 `"1"` 表示动画在初次播放后重复播放一次，因此动画总共播放两次。默认值为 `"0"`，表示不重复。

`android:repeatMode`

整数。动画播放到结尾处的行为。`android:repeatCount` 必须设置为正整数或 `"-1"`，该属性才有效。设置为 `"reverse"` 可让动画在每次迭代时反向播放，设置为 `"repeat"` 则可让动画每次从头开始循环播放。

`android:valueType`

关键字。如果值为颜色，则不要指定此属性。动画框架会自动处理颜色值。

值	说明
`intType`	指定动画值为整数
`floatType`（默认）	指定动画值为浮点数

示例:

保存在 `res/animator/property_animator.xml` 的 XML 文件:

```
<set android:ordering="sequentially">
    <set>
        <objectAnimator
            android:propertyName="x"
            android:duration="500"
            android:valueTo="400"
            android:valueType="intType"/>
        <objectAnimator
            android:propertyName="y"
            android:duration="500"
            android:valueTo="300"
            android:valueType="intType"/>
    </set>
    <objectAnimator
```

```
        android:propertyName="alpha"
        android:duration="500"
        android:valueTo="1f"/>
</set>
```

为了运行此动画，您必须将代码中的 XML 资源扩充为 AnimatorSet (/reference/android/animation/AnimatorSet) 对象，然后在开始运行动画集之前为所有动画设置目标对象。为方便起见，调用 setTarget() (/reference/android/animation/AnimatorSet#setTarget(java.lang.Object)) 即可设置一个用于 AnimatorSet (/reference/android/animation/AnimatorSet) 的所有子项的目标对象。以下代码展示了如何执行此操作：

KOTLIN (#KOTLIN)JAVA

```
AnimatorSet set = (AnimatorSet) AnimatorInflater.loadAnimator(myContext,
    R.animator.property_animator);
set.setTarget(myObject);
set.start();
```

另请参阅：

- 属性动画 (/guide/topics/graphics/prop-animation)
- 有关如何使用属性动画系统的示例，请参阅 API 演示 (/resources/samples/ApiDemos/src/com/example/android/apis/animation)。

视图动画

视图动画框架可支持补间动画和逐帧动画，两者都可以在 XML 中声明。以下几个部分介绍如何使用这两种方法。

补间动画

在 XML 中定义的动画，用于对图形执行旋转、淡出、移动和拉伸等转换。

文件位置：

`res/anim/`*`filename`*`.xml`
该文件名将用作资源 ID。

编译后的资源数据类型：

指向 Animation (/reference/android/view/animation/Animation) 的资源指针。

资源引用：

在 Java 中：`R.anim.`*`filename`*
在 XML 中：`@[`*`package`*`:]anim/`*`filename`*

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<set (#set-element) xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource"
    android:shareInterpolator=["true" | "false"] >
    <alpha (#alpha-element)
        android:fromAlpha="float"
        android:toAlpha="float" />
    <scale (#scale-element)
        android:fromXScale="float"
        android:toXScale="float"
        android:fromYScale="float"
        android:toYScale="float"
        android:pivotX="float"
        android:pivotY="float" />
    <translate (#translate-element)
```

```
            android:fromXDelta="float"
            android:toXDelta="float"
            android:fromYDelta="float"
            android:toYDelta="float" />
        <rotate (#rotate-element)
            android:fromDegrees="float"
            android:toDegrees="float"
            android:pivotX="float"
            android:pivotY="float" />
        <set (#set-element)>
            ...
        </set>
    </set>
```

该文件必须具有一个根元素，可以是 `<alpha>`、`<scale>`、`<translate>`、`<rotate>` 或包含一组（或多组）其他动画元素（甚至是嵌套的 `<set>` 元素）的 `<set>` 元素。

元素：

`<set>`

容纳其他动画元素（`<alpha>`、`<scale>`、`<translate>`、`<rotate>`）或其他 `<set>` 元素的容器。代表 AnimationSet (/reference/android/view/animation/AnimationSet)。

属性：

`android:interpolator`

插值器资源。 要应用于动画的 Interpolator (/reference/android/view/animation/Interpolator)。 该值必须是对指定插值器的资源的引用（而不是插值器类名称）。您可以使用平台提供的默认插值器资源，也可以创建自己的插值器资源。有关插值器 (#Interpolators)的详细信息，请参阅以下说明。

`android:shareInterpolator`

布尔值。如果要在所有子元素中共用同一插值器，则为“true”。

`<alpha>`

淡入或淡出动画。代表 AlphaAnimation (/reference/android/view/animation/AlphaAnimation)。

属性：

`android:fromAlpha`

浮点数。起始不透明度偏移，0.0 表示透明，1.0 表示不透明。

`android:toAlpha`

浮点数。结束不透明度偏移，0.0 表示透明，1.0 表示不透明。

要了解 `<alpha>` 支持的更多属性，请参阅 Animation (/reference/android/view/animation/Animation) 类引用（其中所有 XML 属性均由此元素继承）。

`<scale>`

大小调整动画。您可以通过指定 `pivotX` 和 `pivotY`，来指定图片向外（或向内）扩展的中心点。例如，如果这两个值为 0、0（左上角），则所有扩展均向右下方向进行。代表 ScaleAnimation (/reference/android/view/animation/ScaleAnimation)。

属性：

`android:fromXScale`

浮点数。起始 X 尺寸偏移，其中 1.0 表示不变。

`android:toXScale`

浮点数。结束 X 尺寸偏移，其中 1.0 表示不变。

android:fromYScale

浮点数。起始 Y 尺寸偏移，其中 1.0 表示不变。

android:toYScale

浮点数。结束 Y 尺寸偏移，其中 1.0 表示不变。

android:pivotX

浮点数。在对象缩放时要保持不变的 X 坐标。

android:pivotY

浮点数。在对象缩放时要保持不变的 Y 坐标。

要了解 <scale> 支持的更多属性，请参阅 Animation (/reference/android/view/animation/Animation) 类引用（其中所有 XML 属性均由此元素继承）。

<translate>

垂直和/或水平移动。支持采用以下三种格式之一的以下属性：从 -100 到 100 的以“%”结尾的值，表示相对于自身的百分比；从 -100 到 100 的以“%p”结尾的值，表示相对于其父项的百分比；不带后缀的浮点值，表示绝对值。代表 TranslateAnimation (/reference/android/view/animation/TranslateAnimation)。

属性：

android:fromXDelta

浮动数或百分比。起始 X 偏移。表示方式：相对于正常位置的像素数（例如 "5"），相对于元素宽度的百分比（例如 "5%"），或相对于父项宽度的百分比（例如 "5%p"）。

android:toXDelta

浮动数或百分比。结束 X 偏移。表示方式：相对于正常位置的像素数（例如 "5"），相对于元素宽度的百分比（例如 "5%"），或相对于父项宽度的百分比（例如 "5%p"）。

android:fromYDelta

浮动数或百分比。起始 Y 偏移。表示方式：相对于正常位置的像素数（例如 "5"），相对于元素高度的百分比（例如 "5%"），或相对于父项高度的百分比（例如 "5%p"）。

android:toYDelta

浮动数或百分比。结束 Y 偏移。表示方式：相对于正常位置的像素数（例如 "5"），相对于元素高度的百分比（例如 "5%"），或相对于父项高度的百分比（例如 "5%p"）。

要了解 <translate> 支持的更多属性，请参阅 Animation (/reference/android/view/animation/Animation) 类引用（其中所有 XML 属性均由此元素继承）。

<rotate>

旋转动画。代表 RotateAnimation (/reference/android/view/animation/RotateAnimation)。

属性：

android:fromDegrees

浮点数。起始角度位置，以度为单位。

android:toDegrees

浮点数。结束角度位置，以度为单位。

android:pivotX

浮动数或百分比。旋转中心的 X 坐标。表示方式：相对于对象左边缘的像素数（例如 "5"），相对于对象左边缘的百分比（例如 "5%"），或相对于父级容器左边缘的百分比（例如 "5%p"）。

android:pivotY

浮点数或百分比。旋转中心的 Y 坐标。表示方式：相对于对象上边缘的像素数（例如 "5"），相对于对象上边缘的百分比（例如 "5%"），或相对于父级容器上边缘的百分比（例如 "5%p"）。

要了解 <rotate> 支持的更多属性，请参阅 Animation (/reference/android/view/animation/Animation) 类引用（其中所有 XML 属性均由此元素继承）。

示例：

保存在 res/anim/hyperspace_jump.xml 的 XML 文件：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale="1.0"
        android:toXScale="1.4"
        android:fromYScale="1.0"
        android:toYScale="0.6"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
        android:duration="700" />
    <set
        android:interpolator="@android:anim/accelerate_interpolator"
        android:startOffset="700">
        <scale
            android:fromXScale="1.4"
            android:toXScale="0.0"
            android:fromYScale="0.6"
            android:toYScale="0.0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:duration="400" />
        <rotate
            android:fromDegrees="0"
            android:toDegrees="-45"
            android:toYScale="0.0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:duration="400" />
    </set>
</set>
```

以下应用代码会将动画应用到 ImageView (/reference/android/widget/ImageView) 并启动动画：

KOTLIN (#KOTLIN)JAVA

```
ImageView image = (ImageView) findViewById(R.id.image);
Animation hyperspaceJump = AnimationUtils.loadAnimation (/reference/android/view/animation/AnimationUtils#loadAnimation(ar
image.startAnimation (/reference/android/view/View#startAnimation(android.view.animation.Animation))(hyperspaceJump);
```

另请参阅：

- 2D 图形：补间动画 (/guide/topics/graphics/view-animation#tween-animation)

插值器

插值器是在 XML 中定义的动画修改器，它会影响动画的变化率。插值器可对现有的动画效果执行加速、减速、重复、退回等。

插值器通过 android:interpolator 属性应用于动画元素，该属性的值是对插值器资源的引用。

Android 中提供的所有插值器都是 Interpolator (/reference/android/view/animation/Interpolator) 类的子类。为便于您使用 android:interpolator 属性将插值器应用于动画，Android 针对每个插值器类包含了一个可供您引用的公共资源。下表指定了每个插值器

要使用的资源：

插值器类	资源 ID
AccelerateDecelerateInterpolator (/reference/android/view/animation/AccelerateDecelerateInterpolator)	`@android:anim/accelerate_decelerate_interpolator`
AccelerateInterpolator (/reference/android/view/animation/AccelerateInterpolator)	`@android:anim/accelerate_interpolator`
AnticipateInterpolator (/reference/android/view/animation/AnticipateInterpolator)	`@android:anim/anticipate_interpolator`
AnticipateOvershootInterpolator (/reference/android/view/animation/AnticipateOvershootInterpolator)	`@android:anim/anticipate_overshoot_interpolator`
BounceInterpolator (/reference/android/view/animation/BounceInterpolator)	`@android:anim/bounce_interpolator`
CycleInterpolator (/reference/android/view/animation/CycleInterpolator)	`@android:anim/cycle_interpolator`
DecelerateInterpolator (/reference/android/view/animation/DecelerateInterpolator)	`@android:anim/decelerate_interpolator`
LinearInterpolator (/reference/android/view/animation/LinearInterpolator)	`@android:anim/linear_interpolator`
OvershootInterpolator (/reference/android/view/animation/OvershootInterpolator)	`@android:anim/overshoot_interpolator`

您可以通过以下方式使用 `android:interpolator` 属性应用上述某个插值器：

```
<set android:interpolator="@android:anim/accelerate_interpolator">
    ...
</set>
```

自定义插值器

如果您对平台提供的插值器（在上表中列出）不满意，则可以使用修改过的属性创建自定义插值器资源。例如，您可以调整 AnticipateInterpolator (/reference/android/view/animation/AnticipateInterpolator) 的加速率或调整 CycleInterpolator (/reference/android/view/animation/CycleInterpolator) 的循环次数。为此，您需要在 XML 文件中创建自己的插值器资源。

文件位置：

`res/anim/`*`filename`*`.xml`
该文件名将用作资源 ID。

编译后的资源数据类型：

指向相应插值器对象的资源指针。

资源引用：

在 XML 中：`@[`*`package`*`:]anim/`*`filename`*

语法：

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

如果您不应用任何属性，则您的插值器的运作方式将与平台提供的插值器（在上表中列出）完全相同。

元素：

请注意，在 XML 中定义的每个 Interpolator (/reference/android/view/animation/Interpolator) 实现的名称都以小写字母开头。

```
<accelerateDecelerateInterpolator>
```

变化率在开始和结束时缓慢，但在中间会加快。
没有属性。

`<accelerateInterpolator>`

变化率在开始时较为缓慢，然后会加快。
属性：

`android:factor`

浮点数。加速率（默认为 1）。

`<anticipateInterpolator>`

先反向变化，然后再急速正向变化。
属性：

`android:tension`

浮点数。要应用的张力（默认为 2）。

`<anticipateOvershootInterpolator>`

先反向变化，再急速正向变化并超过目标值，然后以最终值结束。
属性：

`android:tension`

浮点数。要应用的张力（默认为 2）。

`android:extraTension`

浮点数。张力要乘以的倍数（默认值为 1.5）。

`<bounceInterpolator>`

变化会在结束时退回。
没有属性。

`<cycleInterpolator>`

按指定的循环次数重复动画。变化率符合正弦曲线图。

属性：

`android:cycles`

整数。循环次数（默认值为 1）。

`<decelerateInterpolator>`

变化率开始时很快，然后减慢。
属性：

`android:factor`

浮点数。减速率（默认值为 1）。

`<linearInterpolator>`

变化率恒定不变。
没有属性。

`<overshootInterpolator>`

先急速正向变化，再超过最终值，然后回到最终值。
属性：

`android:tension`

浮点数。要应用的张力（默认为 2）。

示例：

保存在 `res/anim/my_overshoot_interpolator.xml` 的 XML 文件：

```
<?xml version="1.0" encoding="utf-8"?>
<overshootInterpolator xmlns:android="http://schemas.android.com/apk/res/android"
    android:tension="7.0"
    />
```

此动画 XML 将应用插值器：

```
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@anim/my_overshoot_interpolator"
    android:fromXScale="1.0"
    android:toXScale="3.0"
    android:fromYScale="1.0"
    android:toYScale="3.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="700" />
```

帧动画

在 XML 中定义的按顺序显示一系列图片的动画（如电影）。

文件位置：

`res/drawable/`*`filename`*`.xml`
该文件名将用作资源 ID。

编译后的资源数据类型：

指向 `AnimationDrawable` (/reference/android/graphics/drawable/AnimationDrawable) 的资源指针。

资源引用：

在 Java 中：`R.drawable.`*`filename`*
在 XML 中：`@[`*`package`*`:]drawable.`*`filename`*

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list (#animation-list-element) xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot=["true" | "false"] >
    <item (#item-element)
        android:drawable="@[package:]drawable/drawable_resource_name"
        android:duration="integer" />
</animation-list>
```

元素：

`<animation-list>`

必需。此元素必须是根元素。包含一个或多个 `<item>` 元素。

属性：

`android:oneshot`

布尔值。如果您想要执行一次动画，则为“true”；如果要循环播放动画，则为“false”。

`<item>`

单帧动画。必须是 `<animation-list>` 元素的子元素。

属性：

`android:drawable`

可绘制资源。要用于此帧的可绘制对象。

`android:duration`

整数。显示此帧的持续时间，以毫秒为单位。

示例：

保存在 `res/drawable/rocket.xml` 的 XML 文件：

```
<?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/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>
```

以下应用代码会将该动画设置为 View 的背景，然后播放动画：

KOTLIN (#KOTLIN)JAVA

```
ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
rocketImage.setBackgroundResource (/reference/android/view/View#setBackgroundResource(int))(R.drawable.rocket_thrus

rocketAnimation = rocketImage.getBackground() (/reference/android/view/View#getBackground());
if (rocketAnimation instanceof Animatable (/reference/android/graphics/drawable/Animatable)) {
    ((Animatable)rocketAnimation).start() (/reference/android/graphics/drawable/Animatable#start());
}
```

另请参阅：

- 2D 图形：帧动画 (/guide/topics/graphics/view-animation#frame-animation)

上一页
← 概览 (/guide/topics/resources/available-resources)

下一页
颜色状态列表 (/guide/topics/resources/color-list-resource) →