视图动画

您可以使用视图动画系统对视图执行补间动画。补间动画根据动画的起点、终点、大小、旋转情况和其他常见方面来计算动画。

补间动画可对视图对象的内容执行一系列简单的转换（位置、大小、旋转情况和透明度）。因此，如果您有一个 `TextView` (/reference/android/widget/TextView) 对象，则可以移动、旋转、放大或缩小文本。如果此对象具有背景图片，则该背景图片会与文本一同转换。`animation package` (/reference/android/view/animation/package-summary) 提供了补间动画所用的所有类。

您可以使用一系列动画指令通过 XML 或 Android 代码定义补间动画。与定义布局一样，我们建议使用 XML 文件，因为它的易读性、可重用性和可交换性比硬编码动画更高。在下面的示例中，我们使用了 XML。（要详细了解如何使用应用代码（而非 XML）来定义动画，请参阅 `AnimationSet` (/reference/android/view/animation/AnimationSet) 类和其他 `Animation` (/reference/android/view/animation/Animation) 子类。）

这些动画指令定义了您希望发生的转换、发生的时间以及应用这些转换所需的时间。转换可以按顺序发生或同时发生；例如，您可以让 TextView 的内容从左侧移至右侧，然后旋转 180 度，或者，您也可以让文本同时移动和旋转。每个转换都会采用一组特定于该转换的参数（若是大小变化，则为起始大小和最终大小；若是旋转，则为起始角度和最终角度等）以及一组通用参数（如开始时间和持续时间）。要使多个转换同时发生，请为它们指定相同的开始时间；要使它们按顺序发生，请在计算开始时间时加上前一个转换的持续时间。

动画 XML 文件位于 Android 项目的 `res/anim/` 目录中。该文件必须具有单个根元素：它可以是单个 `<alpha>`、`<scale>`、`<translate>`、`<rotate>` 或插值器元素，也可以是包含这些元素组的 `<set>` 元素（可能包含其他 `<set>`）。默认情况下，系统会同时应用所有动画指令。要使它们按顺序发生，您必须指定 `startOffset` 属性，如以下示例所示。

以下来自某个 ApiDemos 的 XML 用于拉伸，然后同时转动和旋转视图对象。

```
<set 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/decelerate_interpolator">
        <scale
           android:fromXScale="1.4"
           android:toXScale="0.0"
           android:fromYScale="0.6"
           android:toYScale="0.0"
           android:pivotX="50%"
           android:pivotY="50%"
           android:startOffset="700"
           android:duration="400"
           android:fillBefore="false" />
        <rotate
           android:fromDegrees="0"
           android:toDegrees="-45"
           android:toYScale="0.0"
           android:pivotX="50%"
           android:pivotY="50%"
           android:startOffset="700"
           android:duration="400" />
    </set>
</set>
```

屏幕左上角处的坐标（此示例中未使用）为 (0,0)，它们会随着您向下和向右移动而增加。

某些值（例如 pivotX）可以相对于对象本身或相对于父级对象进行指定。请务必根据需要使用正确的格式（“50”表示相对于父级对象的 50%；“50%”表示相对于对象本身的 50%）。

您可以通过分配 `Interpolator` (/reference/android/view/animation/Interpolator) 来确定如何随着时间的推移应用转换。Android 提供多个用于指定各种速度曲线的插值器子类；例如，`AccelerateInterpolator` (/reference/android/view/animation/AccelerateInterpolator) 表示转换在开始时较慢，然后会加速。每个子类都具有一个可在 XML 中应用的属性值。

将此 XML 保存为项目 `res/anim/` 目录下的 `hyperspace_jump.xml` 后，以下代码将引用该 XML 文件并将其应用到布局的 `ImageView` (/reference/android/widget/ImageView) 对象。

KOTLIN (#KOTLIN)**JAVA**

```
ImageView spaceshipImage = (ImageView) findViewById(R.id.spaceshipImage);
Animation hyperspaceJumpAnimation = AnimationUtils.loadAnimation(this, R.anim.hyperspace_jump);
spaceshipImage.startAnimation(hyperspaceJumpAnimation);
```

作为 `startAnimation()` 的替代方法，您可以使用 `Animation.setStartTime()` (/reference/android/view/animation/Animation#setStartTime(long)) 定义动画的开始时间，然后使用 `View.setAnimation()` (/reference/android/view/View#setAnimation(android.view.animation.Animation)) 将动画分配到视图。

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)。

注意：无论动画如何移动或调整大小，用于容纳动画的视图的边界都不会通过自动调整来适应它。即便如此，动画仍会被绘制到相应视图的边界之外，并且不会被剪裁。但是，如果动画超出了父级视图的边界，系统将会进行剪裁。