

视图动画(View Animation)

原创xyzso1z最后发布于2020-04-19 23:47:57阅读数2☆收藏

编辑展开

本文章源码

视图动画

视图动画可分为：**补间动画**、**逐帧动画**。

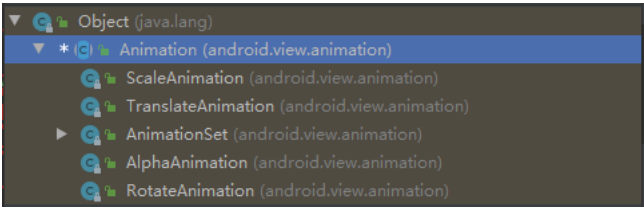
一、补间动画(Tweened Animation)

分类

补间动画：

- 平移动画(TranslateAnimation)
- 缩放动画(ScaleAnimation)
- 旋转动画(RotateAnimation)
- 透明度动画(AlphaAnimation)

如下图，



可以看到途中几个类都是Animation的子类。这4种动画既能分开独立实现，也可以组合实现复合动画 **AnimationSet**。

缺点

- 只能实现移动、缩放、旋转和淡入淡出这四种动画。
- 只能改变View的显示效果，不会改变View的属性。

属性动画可以完美避免这两缺点

优点

- 简单易用

使用

视图(View)动画的实现可以通过xml来定义，也可以通过Java代码来动态设置。对于视图动画，建议使用xml来定义动画，刻度性好，而且能够复用。

资源建立步骤:

1. 在res文件下右击New->Directory新建 **anim** 文件夹
2. **anim** 文件下右击New->Animation Resource File然后弹出弹窗；
File name：文件名
Root element：根节点类型(set、alpha、scale、translate、rotate...)

如下，使用xml来定义：

文件目录：**res/anim/animation_test.xml**

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <set xmlns:android="http://schemas.android.com/apk/res/android"
3      android:interpolator="@[package:]anim/interpolator_resource"
4      android:shareInterpolator=[ "true" | "false" ] >
5      <!-- 透明度动画 -->
6      <alpha
7          android:fromAlpha="float"
8          android:toAlpha="float" />
9      <!-- 缩放动画 -->
10     <scale
11         android:fromXScale="float"
12         android:toXScale="float"
13         android:fromYScale="float"
14         android:toYScale="float"
15         android:pivotX="float"
16         android:pivotY="float" />
17     <!-- 平移动画 -->
18     <translate
19         android:fromXDelta="float"
20         android:toXDelta="float"
21         android:fromYDelta="float"
22         android:toYDelta="float" />
23     <!-- 旋转动画 -->
24     <rotate
25         android:fromDegrees="float"
26         android:toDegrees="float"
27         android:pivotX="float"
28         android:pivotY="float" />
29     <set>
30         ...
31     </set>
32 </set>

```

公共属性介绍：

动画持续时间

- xml属性：`android:duration="100"`
- Java方法：`setDuration(100L)`
- 说明：默认值是0 (单位ms)

动画结束后是否保留动画前的状态

- xml属性：`android:fillAfter="true"`
- Java方法：`setFillAfter(boolean)`
- 说明：动画结束后是否保留动画后的状态，true保留动画后状态，false恢复原来状态，默认值是false

动画结束后是否保留动画前的状态

- xml属性：`android:fillBefore="true"`
- Java方法：`setFillBefore(boolean)`
- 说明：动画结束后是否保留动画前的状态，true恢复原来状态，false保留动画后状态，默认值是true

动画的变化速率 即插值器

- xml属性：`android:interpolator="@android:anim/accelerate_decelerate_interpolator"`
- Java方法：`setInterpolator(Interpolator)`
- 说明：设置动画的变化速率 即插值器，改变动画变换的速度，默认值是@android:anim/accelerate_decelerate_interpolator，即加速减速插值器，在动画开始和结束的时速度较慢，中间时候加速

动画重复执行的次数

- xml属性：`android:repeatCount="9"`
- Java方法：`setRepeatCount(int)`
- 说明：设置动画重复执行的次数，默认值是0

动画重复的模式

- xml属性：`android:repeatMode="reverse"`
- Java方法：`setRepeatMode(int)`
- 说明：设置动画重复模式，其值有restart(1):顺序播放，reverse(2):重复的时候逆向播放

开始的延迟的时间

- xml属性：`android:startOffset="0"`
- Java方法：`setStartOffset(long)`
- 说明：设置开始的延迟的时间（单位ms），默认值是0

加载资源文件动画

```
1 | Animation animation = AnimationUtils.loadAnimation(this, R.anim.test_alpha);
```

为 View 添加动画

```
1 | view.setAnimation(animation);
```

设置动画监听

```
1 | animation.setAnimationListener(new Animation.AnimationListener() {
2 |     @Override
3 |     public void onAnimationStart(Animation animation) {
4 |         // 动画开始时回调
5 |     }
6 |
7 |     @Override
8 |     public void onAnimationEnd(Animation animation) {
9 |         // 动画结束时回调(动画最终结束)
10 |    }
11 |
12 |    @Override
13 |    public void onAnimationRepeat(Animation animation) {
14 |        // 动画重复时回调
15 |    }
16 | });
17 |
```

启动动画

```
1 | animation.start();
2 | // 或
3 | view.startAnimation(animation);
```

结束动画

```
1 | animation.cancel();
```

详细讲解各动画属性

AlphaAnimation

作用：透明度动画，改变视图整体透明度，透明度值由1~0，从可见到不可见的变化。

xml实现

步骤：在 `anim` 文件，右击new->Android Resource File, 文件名：`alpha_test`, 根节点：`alpha`

```
1 | <?xml version="1.0" encoding="utf-8"?>
2 | <alpha xmlns:android="http://schemas.android.com/apk/res/android"
3 |     android:fromAlpha="0.0"
4 |     android:toAlpha="1.0"
```

```

4     android:duration="2000">
5     </alpha>
6

```

各属性含义：

- **android:fromAlpha** : 表示透明度的起始值，这里设置为0.0，表示完全透明，取值范围0~1；
- **android:toAlpha** : 表示透明度的结束值，这里设置为1.0，表示完全不透明，取值范围0~1；
- **android:duration** : 表示动画持续的时间，这里设置为2000，单位是毫秒；

Java代码中使用xml：使用AnimationUtils类的静态方法loadAnimation()来加载XML文件，得到一个Animation对象，如下：

```

1 Animation animation = AnimationUtils.loadAnimation(this, R.anim.alpha);
2 mImage.startAnimation(animation);

```

Java实现

```

1 AlphaAnimation alphaAnimation = new AlphaAnimation(0f, 1f);
2 alphaAnimation.setDuration(3000);
3 mImage.startAnimation(alphaAnimation);

```

构造方法：

1. 通过加载资源文件

```

1 /**
2  * Constructor used when an AlphaAnimation is loaded from a resource.
3  *
4  * @param context Application context to use
5  * @param attrs Attribute set from which to read values
6  */
7 public AlphaAnimation(Context context, AttributeSet attrs) {
8     super(context, attrs);
9 }

```

2. 通过代码构建动画

```

1 /**
2  * Constructor to use when building an AlphaAnimation from code
3  *
4  * @param fromAlpha Starting alpha value for the animation, where 1.0 means
5  *         fully opaque and 0.0 means fully transparent.
6  * @param toAlpha Ending alpha value for the animation.
7  */
8 public AlphaAnimation(float fromAlpha, float toAlpha) {
9     mFromAlpha = fromAlpha;
10    mToAlpha = toAlpha;
11 }

```

ScaleAnimation

缩放动画，需要一个坐标点来，即轴点，实现以不同的轴点缩放效果，因此需要先指定 **pivotX**、**pivotY** 确定轴坐标。默认情况下，从对象 **View** 的左上角开始缩放。

xml实现

该动画通过标签 **<scale/>** 实现的，如下：

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <scale xmlns:android="http://schemas.android.com/apk/res/android"
3     android:duration="3000"
4     android:fromXScale="0.0"
5     android:fromYScale="0.0"
6     android:pivotX="50%"
7     android:pivotY="50%"
8     android:toXScale="1.0"
9     android:toYScale="1.0">
10
11 </scale>

```

各属性如下：

- **android:fromXScale**: 动画开始时, 水平方向缩放系数。
- **android:fromYScale**: 动画开始时, 垂直方向缩放系数。
- **android:toXScale**: 动画结束时, 水平方向缩放系数。
- **android:toYScale**: 动画结束时, 垂直方向缩放系数。
- **android:pivotX**: 缩放轴点的X坐标 (其值可以为: 数值、百分数、百分数p), 例如: 如50表示以当前View左上角坐标加50px为初始点、50%表示以当前View的左上角加上当前View宽高的50%做为初始点、50%p表示以当前View的左上角加上父控件宽高的50%做为初始点)。
- **android:pivotY**: 缩放轴点的Y坐标, 规律同pivotX。

java实现

需要使用Animation的子类ScaleAnimation来实现, 代码如下:

```
1 | ScaleAnimation scaleAnimation = new ScaleAnimation(0, 1, 0, 1,
2 |     Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
3 | scaleAnimation.setDuration(3000);
4 | mImage.startAnimation(scaleAnimation);
```

构造方法:

```
1 | // 1.
2 | /**
3 |  * Constructor used when a ScaleAnimation is loaded from a resource.
4 |  *
5 |  * @param context Application context to use
6 |  * @param attrs Attribute set from which to read values
7 |  */
8 | public ScaleAnimation(Context context, AttributeSet attrs) {
9 |     super(context, attrs);
10 | }
11 |
12 | // 2.
13 | /**
14 |  * Constructor to use when building a ScaleAnimation from code
15 |  *
16 |  * @param fromX Horizontal scaling factor to apply at the start of the
17 |  *     animation
18 |  * @param toX Horizontal scaling factor to apply at the end of the animation
19 |  * @param fromY Vertical scaling factor to apply at the start of the
20 |  *     animation
21 |  * @param toY Vertical scaling factor to apply at the end of the animation
22 |  */
23 | public ScaleAnimation(float fromX, float toX, float fromY, float toY) {
24 |     mResources = null;
25 |     mFromX = fromX;
26 |     mToX = toX;
27 |     mFromY = fromY;
28 |     mToY = toY;
29 |     mPivotX = 0;
30 |     mPivotY = 0;
31 | }
32 |
33 | // 3.
34 | /**
35 |  * Constructor to use when building a ScaleAnimation from code
36 |  *
37 |  * @param fromX Horizontal scaling factor to apply at the start of the
38 |  *     animation
39 |  * @param toX Horizontal scaling factor to apply at the end of the animation
40 |  * @param fromY Vertical scaling factor to apply at the start of the
41 |  *     animation
42 |  * @param toY Vertical scaling factor to apply at the end of the animation
43 |  * @param pivotX The X coordinate of the point about which the object is
44 |  *     being scaled, specified as an absolute number where 0 is the left
45 |  *     edge. (This point remains fixed while the object changes size.)
46 |  * @param pivotY The Y coordinate of the point about which the object is
47 |  *     being scaled, specified as an absolute number where 0 is the top
48 |  *     edge. (This point remains fixed while the object changes size.)
49 |  */
50 | public ScaleAnimation(float fromX, float toX, float fromY, float toY,
51 |     float pivotX, float pivotY) {
```

```

48 // 4.
49 /**
50  * Constructor to use when building a ScaleAnimation from code
51  *
52  * @param fromX Horizontal scaling factor to apply at the start of the
53  *   animation
54  * @param toX Horizontal scaling factor to apply at the end of the animation
55  * @param fromY Vertical scaling factor to apply at the start of the
56  *   animation
57  * @param toY Vertical scaling factor to apply at the end of the animation
58  * @param pivotXType Specifies how pivotXValue should be interpreted. One of
59  *   Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
60  *   Animation.RELATIVE_TO_PARENT.
61  * @param pivotXValue The X coordinate of the point about which the object
62  *   is being scaled, specified as an absolute number where 0 is the
63  *   left edge. (This point remains fixed while the object changes
64  *   size.) This value can either be an absolute number if pivotXType
65  *   is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
66  * @param pivotYType Specifies how pivotYValue should be interpreted. One of
67  *   Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
68  *   Animation.RELATIVE_TO_PARENT.
69  * @param pivotYValue The Y coordinate of the point about which the object
70  *   is being scaled, specified as an absolute number where 0 is the
71  *   top edge. (This point remains fixed while the object changes
72  *   size.) This value can either be an absolute number if pivotYType
73  *   is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
74  */
75 public ScaleAnimation(float fromX, float toX, float fromY, float toY,
76                       int pivotXType, float pivotXValue, int pivotYType, float pivotYValue) {
77
78 }
79

```

RotateAnimation

旋转动画，与缩放动画较为相似，也需要一个轴点来实现旋转。默认情况下，从对象view的左上角开始旋转，也即是相对于当前view坐标为（0,0）位置。

在xml实现

该动画是通过标签实现的，实现代码如下：

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <rotate xmlns:android="http://schemas.android.com/apk/res/android"
3     android:duration="3000"
4     android:pivotX="50%"
5     android:pivotY="50%"
6     android:fromDegrees="0"
7     android:toDegrees="90">
8
9 </rotate>

```

各属性含义：

- **android:pivotX** :旋转轴点的X坐标（其值可以为：数值、百分数、百分数p），例如：如50表示以当前View左上角坐标加50px为初始点、50%表示以当前View的左上角加上当前View宽高的50%做为初始点、50%p表示以当前View的左上角加上父控件宽高的50%做为初始点）。
- **android:pivotY** :旋转轴点的Y坐标，规律同 **android:pivotX**。
- **android:fromDegrees** :旋转开始的角度，其值可以为正负。
- **android:toDegrees** : 旋转结束的角度，其值可以为正负。
- ps：**toDegrees - fromDegrees > 0**，则顺时针旋转；否则，逆时针旋转。

java实现

需要使用Animation的子类RotateAnimation 来实现，代码如下：

```

1 RotateAnimation rotateAnimation = new RotateAnimation(0,180);
2 rotateAnimation.setDuration(3000);
3 mImage.startAnimation(rotateAnimation);

```

构造方法：

```

1 // 1.
2 /**
3  * Constructor used when a RotateAnimation is loaded from a resource

```

```

4      Constructor used when a RotateAnimation is loaded from a resource.
5      *
6      * @param context Application context to use
7      * @param attrs Attribute set from which to read values
8      */
9      public RotateAnimation(Context context, AttributeSet attrs) {
10         super(context, attrs);
11     }
12
13     // 2.
14     /**
15      * Constructor to use when building a RotateAnimation from code.
16      * Default pivotX/pivotY point is (0,0).
17      *
18      * @param fromDegrees Rotation offset to apply at the start of the
19      * animation.
20      *
21      * @param toDegrees Rotation offset to apply at the end of the animation.
22      */
23     public RotateAnimation(float fromDegrees, float toDegrees) {
24     }
25
26     // 3.
27     /**
28      * Constructor to use when building a RotateAnimation from code
29      *
30      * @param fromDegrees Rotation offset to apply at the start of the
31      * animation.
32      *
33      * @param toDegrees Rotation offset to apply at the end of the animation.
34      *
35      * @param pivotX The X coordinate of the point about which the object is
36      * being rotated, specified as an absolute number where 0 is the left
37      * edge.
38      *
39      * @param pivotY The Y coordinate of the point about which the object is
40      * being rotated, specified as an absolute number where 0 is the top
41      * edge.
42      */
43     public RotateAnimation(float fromDegrees, float toDegrees, float pivotX, float pivotY) {
44         mFromDegrees = fromDegrees;
45         mToDegrees = toDegrees;
46
47         mPivotXType = ABSOLUTE;
48         mPivotYType = ABSOLUTE;
49         mPivotXValue = pivotX;
50         mPivotYValue = pivotY;
51         initializePivotPoint();
52     }
53
54     //4.
55     /**
56      * Constructor to use when building a RotateAnimation from code
57      *
58      * @param fromDegrees Rotation offset to apply at the start of the
59      * animation.
60      *
61      * @param toDegrees Rotation offset to apply at the end of the animation.
62      *
63      * @param pivotXType Specifies how pivotXValue should be interpreted. One of
64      * Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
65      * Animation.RELATIVE_TO_PARENT.
66      *
67      * @param pivotXValue The X coordinate of the point about which the object
68      * is being rotated, specified as an absolute number where 0 is the
69      * left edge. This value can either be an absolute number if
70      * pivotXType is ABSOLUTE, or a percentage (where 1.0 is 100%)
71      * otherwise.
72      *
73      * @param pivotYType Specifies how pivotYValue should be interpreted. One of
74      * Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
75      * Animation.RELATIVE_TO_PARENT.
76      *
77      * @param pivotYValue The Y coordinate of the point about which the object
78      * is being rotated, specified as an absolute number where 0 is the
79      * top edge. This value can either be an absolute number if
80      * pivotYType is ABSOLUTE, or a percentage (where 1.0 is 100%)
81      * otherwise.
82      */
83     public RotateAnimation(float fromDegrees, float toDegrees, int pivotXType, float pivotXValue,
84         int pivotYType, float pivotYValue) {
85         mFromDegrees = fromDegrees;
86         mToDegrees = toDegrees;
87     }

```

```

74         mPivotXValue = pivotXValue;
75         mPivotXType = pivotXType;
76         mPivotYValue = pivotYValue;
77         mPivotYType = pivotYType;
78         initializePivotPoint();
79     }
80
81

```

TranslateAnimation

平移动画，实现视图垂直/水平方向位移变化，指定开始的位置，和结束的位置即可。

xml实现：

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <translate xmlns:android="http://schemas.android.com/apk/res/android"
3             android:fillAfter="true"
4             android:duration="3000"
5             android:fromXDelta="50%"
6             android:fromYDelta="50%"
7             android:toXDelta="50%p"
8             android:toYDelta="50%p">
9  </translate>

```

各属性含义：

- **android:fromXDelta**：平移开始X方向坐标（其值可以为：数值、百分数、百分数p，且多可以为正负），其值含义与 **android:pivotX** 类似。
- **android:fromYDelta**：平移开始Y方向坐标（其值可以为：数值、百分数、百分数p，且多可以为正负），其值含义与 **android:pivotY** 类似。
- **android:toXDelta**：平移结束X方向坐标（其值可以为：数值、百分数、百分数p，且多可以为正负），其值含义与 **android:pivotX** 类似。
- **android:toYDelta**：平移结束Y方向坐标（其值可以为：数值、百分数、百分数p，且多可以为正负），其值含义与 **android:pivotY** 类似。

java实现

需要使用Animation的子类TranslateAnimation来实现，代码如下：

```

1  TranslateAnimation translateAnimation = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0,Animation.RELATIVE_TO_SELF,0);
2  translateAnimation.setDuration(3000);
3  translateAnimation.setFillAfter(true);
4  mImage.startAnimation(translateAnimation);

```

构造函数：

```

1  // 1.
2  /**
3   * Constructor used when a TranslateAnimation is loaded from a resource.
4   *
5   * @param context Application context to use
6   * @param attrs Attribute set from which to read values
7   */
8  public TranslateAnimation(Context context, AttributeSet attrs) {
9      super(context, attrs);
10 }
11
12 // 2.
13 /**
14 * Constructor to use when building a TranslateAnimation from code
15 *
16 * @param fromXDelta Change in X coordinate to apply at the start of the
17 *                    animation
18 * @param toXDelta Change in X coordinate to apply at the end of the
19 *                    animation
20 * @param fromYDelta Change in Y coordinate to apply at the start of the
21 *                    animation
22 * @param toYDelta Change in Y coordinate to apply at the end of the
23 *                    animation
24 */
25 public TranslateAnimation(float fromXDelta, float toXDelta, float fromYDelta, float toYDelta) {
26     mFromXValue = fromXDelta;
27     mToXValue = toXDelta;
28     mFromYValue = fromYDelta;
29     mToYValue = toYDelta;
30 }

```



```

26         mToYValue = toYDelta;
27
28         mFromXType = ABSOLUTE;
29         mToXType = ABSOLUTE;
30         mFromYType = ABSOLUTE;
31         mToYType = ABSOLUTE;
32     }
33
34     // 3.
35     /**
36      * Constructor to use when building a TranslateAnimation from code
37      *
38      * @param fromXType Specifies how fromXValue should be interpreted. One of
39      *     Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
40      *     Animation.RELATIVE_TO_PARENT.
41      * @param fromXValue Change in X coordinate to apply at the start of the
42      *     animation. This value can either be an absolute number if fromXType
43      *     is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
44      * @param toXType Specifies how toXValue should be interpreted. One of
45      *     Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
46      *     Animation.RELATIVE_TO_PARENT.
47      * @param toXValue Change in X coordinate to apply at the end of the
48      *     animation. This value can either be an absolute number if toXType
49      *     is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
50      * @param fromYType Specifies how fromYValue should be interpreted. One of
51      *     Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
52      *     Animation.RELATIVE_TO_PARENT.
53      * @param fromYValue Change in Y coordinate to apply at the start of the
54      *     animation. This value can either be an absolute number if fromYType
55      *     is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
56      * @param toYType Specifies how toYValue should be interpreted. One of
57      *     Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
58      *     Animation.RELATIVE_TO_PARENT.
59      * @param toYValue Change in Y coordinate to apply at the end of the
60      *     animation. This value can either be an absolute number if toYType
61      *     is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
62      */
63     public TranslateAnimation(int fromXType, float fromXValue, int toXType, float toXValue,
64                             int fromYType, float fromYValue, int toYType, float toYValue) {
65
66         mFromXValue = fromXValue;
67         mToXValue = toXValue;
68         mFromYValue = fromYValue;
69         mToYValue = toYValue;
70
71         mFromXType = fromXType;
72         mToXType = toXType;
73         mFromYType = fromYType;
74         mToYType = toYType;
75     }
76

```

AnimationSet

集合动画，如果单一的动画太过于单调，那么就可以将这些单一的动画组合成个性酷炫的动画，同时指定播放的顺序，并且集合里面可以再包含集合。但注意的是，在集合设置的属性对该标签下的所有子控件都产生影响。

xml实现：

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <set xmlns:android="http://schemas.android.com/apk/res/android"
3      android:shareInterpolator="false">
4      <scale
5          android:duration="2000"
6          android:fromXScale="1.0"
7          android:fromYScale="1.0"
8          android:interpolator="@android:anim/linear_interpolator"
9          android:pivotX="50%"
10         android:pivotY="50%"
11         android:toXScale="0"
12         android:toYScale="0"/>
13
14     <set
15         android:duration="2000"
16         android:interpolator="@android:anim/decelerate_interpolator"
17         android:shareInterpolator="true"
18         android:startOffset="2000">
19         <scale
20             android:fromXScale="0"

```

```

18         android:fromYScale="0"
19         android:pivotX="50%"
20         android:pivotY="50%"
21         android:toXScale="1"
22         android:toYScale="1"/>
23     <rotate
24         android:fromDegrees="0"
25         android:pivotX="50%"
26         android:pivotY="50%"
27         android:toDegrees="180"/>
28 </set>
</set>

```

属性含义：

- **android:shareInterpolator**：子元素是否共享插值器，值为true，表示共同使用；值为false，表示不共享
其余属性多和其他类似，
- **android:startOffset**：设置需要设置播放的顺序。

java实现

```

1         AnimationSet animationSet1 = new AnimationSet(false); //一级集合
2         ScaleAnimation scaleAnimation1 = new ScaleAnimation(1, 1.4f, 1, 1.4f, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
3
4         AnimationSet animationSet2 = new AnimationSet(true); //二级集合
5         ScaleAnimation scaleAnimation2 = new ScaleAnimation(1.4f, 0, 1.4f, 0, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
6         RotateAnimation rotateAnimation = new RotateAnimation(0, 180, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
7
8         animationSet2.addAnimation(scaleAnimation2);
9         animationSet2.addAnimation(rotateAnimation);
10        animationSet2.setInterpolator(new DecelerateInterpolator());
11        animationSet2.setDuration(2000);
12        animationSet2.setStartOffset(2000);
13
14        animationSet1.addAnimation(scaleAnimation1);
15        animationSet1.addAnimation(animationSet2);
16        animationSet1.setInterpolator(new AccelerateDecelerateInterpolator());
17        animationSet1.setDuration(2000);
18
19        mImage.startAnimation(animationSet1);
20

```

构造函数：

```

1 // 1.
2 /**
3  * Constructor used when an AnimationSet is loaded from a resource.
4  *
5  * @param context Application context to use
6  * @param attrs Attribute set from which to read values
7  */
8 public AnimationSet(Context context, AttributeSet attrs) {
9     super(context, attrs);
10
11     TypedArray a =
12         context.obtainStyledAttributes(attrs, com.android.internal.R.styleable.AnimationSet);
13
14     setFlag(PROPERTY_SHARE_INTERPOLATOR_MASK,
15         a.getBoolean(com.android.internal.R.styleable.AnimationSet_shareInterpolator, true));
16     init();
17
18     if (context.getApplicationInfo().targetSdkVersion >=
19         Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
20         if (a.hasValue(com.android.internal.R.styleable.AnimationSet_duration)) {
21             mFlags |= PROPERTY_DURATION_MASK;
22         }
23         if (a.hasValue(com.android.internal.R.styleable.AnimationSet_fillBefore)) {
24             mFlags |= PROPERTY_FILL_BEFORE_MASK;
25         }
26     }
27 }

```

```

26         if (a.hasValue(com.android.internal.R.styleable.AnimationSet_fillAfter)) {
27             mFlags |= PROPERTY_FILL_AFTER_MASK;
28         }
29         if (a.hasValue(com.android.internal.R.styleable.AnimationSet_repeatMode)) {
30             mFlags |= PROPERTY_REPEAT_MODE_MASK;
31         }
32         if (a.hasValue(com.android.internal.R.styleable.AnimationSet_startOffset)) {
33             mFlags |= PROPERTY_START_OFFSET_MASK;
34         }
35     }
36
37     a.recycle();
38 }
39
40 // 2.
41 /**
42  * Constructor to use when building an AnimationSet from code
43  *
44  * @param shareInterpolator Pass true if all of the animations in this set
45  *                           should use the interpolator associated with this AnimationSet.
46  *                           Pass false if each animation should use its own interpolator.
47  */
48 public AnimationSet(boolean shareInterpolator) {
49     setFlag(PROPERTY_SHARE_INTERPOLATOR_MASK, shareInterpolator);
50     init();
51 }

```

问题

你是否真正的明白 pivotX 的含义？

轴点坐标值pivotX, pivotY, 有三种表达方式：在xml文件，其值有3种类型：数值、百分数、百分数p；在java代码中，其值要根据，三种坐标类型（前方有介绍）来确定。

一般，对于表达方式的第一种、第二种，应该都是比较常用的，而且也是比较好理解的，也比较直观。那么问题来了：

提出问题：

如果是使用第三种，相对于父控件定位，你是否能够准确找到轴点位置？是否知道其真正的含义？

猜想：

pivotX = 50%，那么轴点就在该控件（没有覆盖整个屏幕）的中间位置；pivotX = 50%p,那么中心点相对于父控件（覆盖了整个屏幕）就是屏幕的中间点。

对于这个想法，我...我...我...刚开始是这样子想的...

实践：

我们用平移动画来实践，保留动画结束的帧，将其参数设置为：

```

1 TranslateAnimation translateAnimation = new TranslateAnimation(Animation.RELATIVE_TO_PARENT, 0, Animation.RELATIVE_TO_PAREN

```

从相当于父控件的（0,0）移动到父控件的（50%，50%），但实际效果是这样子的，与猜想不符：

[具体效果请看](#)

探索：

我们可以点进去看TranslateAnimation，可以发现：

```

1 @Override
2 public void initialize(int width, int height, int parentWidth, int parentHeight) {
3     super.initialize(width, height, parentWidth, parentHeight);
4     mFromXDelta = resolveSize(mFromXType, mFromXValue, width, parentWidth);
5     mToXDelta = resolveSize(mToXType, mToXValue, width, parentWidth);
6     mFromYDelta = resolveSize(mFromYType, mFromYValue, height, parentHeight);
7     mToYDelta = resolveSize(mToYType, mToYValue, height, parentHeight);
8 }
9
10 // 然后再看resolveSize()方法
11 protected float resolveSize(int type, float value, int size, int parentSize) {

```

```

12         switch (type) {
13             case ABSOLUTE:
14                 return value;
15             case RELATIVE_TO_SELF:
16                 return size * value;
17             case RELATIVE_TO_PARENT:
18                 return parentSize * value;
19             default:
20                 return value;
21         }
22     }
23
24     // 也许，看到这里，反而觉得猜想没有错。其实动画真正的实现是在这里：
25
26     @Override
27     protected void applyTransformation(float interpolatedTime, Transformation t) {
28         float dx = mFromXDelta;
29         float dy = mFromYDelta;
30         if (mFromXDelta != mToXDelta) {
31             dx = mFromXDelta + ((mToXDelta - mFromXDelta) * interpolatedTime);
32         }
33         if (mFromYDelta != mToYDelta) {
34             dy = mFromYDelta + ((mToYDelta - mFromYDelta) * interpolatedTime);
35         }
36         t.getMatrix().setTranslate(dx, dy);
37     }

```

这里，我们发现，动画需要绘制的轨迹是 $dx = mToXDelta$ ， $dy = mToYDelta$ ，这个值是相对于该控件 需要变化的距离，而不是最终的位置，那么最终猜想：

如果该控件刚好位于屏幕的左上角，则 $mToXValue$ 就是动画结束的位置；

如果该控件不在屏幕的左上角，则最终动画后的坐标需要加上该控件这个坐标；

最后用表达式表示：

```
toXValue = fromXType + dx;
```

```
toYType = fromYValue + dy;
```

二、帧动画(Frame Animation)

实现

1. 在res文件下右击New->Directory新建 **drawable** 文件夹
2. **anim** 文件下右击New->Drawable Resource File然后弹出弹窗；
File name：文件名
Root element：根节点类型(选择 **animation-list**)

创建资源xml代码(**res/drawable/frame_view.xml**)

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <animation-list xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <item
5          android:drawable="@drawable/sign_icon_1"
6          android:duration="50" />
7
8      <item
9          android:drawable="@drawable/sign_icon_2"
10         android:duration="50" />
11
12     <item
13         android:drawable="@drawable/sign_icon_3"
14         android:duration="50" />
15
16     <item
17         android:drawable="@drawable/sign_icon_4"
18         android:duration="50" />
19
20 </animation-list>

```

使用(**res/layout/activity_frame_animation.xml**)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:gravity="center">
6
7      <ImageView
8          android:id="@+id/image"
9          android:layout_width="100dp"
10         android:layout_height="100dp"
11         android:src="@drawable/frame_view" />
12 </LinearLayout>
```

开启动画:

```
1  ImageView imageView = findViewById(R.id.image);
2  AnimationDrawable animationDrawable = (AnimationDrawable) imageView.getDrawable();
3
4  if (animationDrawable != null) {
5      animationDrawable.start();
6  }
```

三、资源

[本文章源码](#)