视图动画(View Animation)

本文章源码

视图动画

视图动画可分为：**补间动画**、**逐帧动画**。

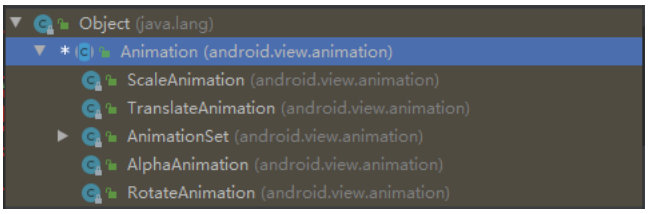
一、补间动画(`Tweened Animation`)

> 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`

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource"
    android:shareInterpolator=["true" | "false"] >
     <!--透明度动画-->
    <alpha
        android:fromAlpha="float"
        android:toAlpha="float" />
    <!--缩放动画-->
    <scale
        android:fromXScale="float"
        android:toXScale="float"
        android:fromYScale="float"
        android:toYScale="float"
        android:pivotX="float"
        android:pivotY="float" />
    <!--平移动画-->
    <translate
        android:fromXDelta="float"
        android:toXDelta="float"
        android:fromYDelta="float"
        android:toYDelta="float" />
    <!--旋转动画-->
    <rotate
        android:fromDegrees="float"
        android:toDegrees="float"
        android:pivotX="float"
        android:pivotY="float" />
    <set>
        ...
    </set>
</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

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

为`View`添加动画

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

设置动画监听

```
animation.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
                //动画开始时回掉
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                //动画结束时回掉(动画最终结束)
            }

            @Override
            public void onAnimationRepeat(Animation animation) {
                 //动画重复时回掉
            }
        });

```

启动动画

```
animation.start();
//或
view.startAnimation(animation);
```

结束动画

```
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`

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

```
    android:duration="2000">
</alpha>
```

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：

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

Java实现

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

构造方法：

1. 通过加载资源文件

```
/**
 * Constructor used when an AlphaAnimation is loaded from a resource.
 *
 * @param context Application context to use
 * @param attrs Attribute set from which to read values
 */
public AlphaAnimation(Context context, AttributeSet attrs) {
    super(context, attrs);
}
```

2. 通过代码构建动画

```
/**
 * Constructor to use when building an AlphaAnimation from code
 *
 * @param fromAlpha Starting alpha value for the animation, where 1.0 means
 *        fully opaque and 0.0 means fully transparent.
 * @param toAlpha Ending alpha value for the animation.
 */
public AlphaAnimation(float fromAlpha, float toAlpha) {
    mFromAlpha = fromAlpha;
    mToAlpha = toAlpha;
}
```

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：

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
       android:duration="3000"
       android:fromXScale="0.0"
       android:fromYScale="0.0"
       android:pivotX="50%"
       android:pivotY="50%"
       android:toXScale="1.0"
       android:toYScale="1.0">
</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：

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

构造方法：

```
    // 1.
    /**
     * Constructor used when a ScaleAnimation is loaded from a resource.
     *
     * @param context Application context to use
     * @param attrs Attribute set from which to read values
     */
    public ScaleAnimation(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    // 2.
    /**
     * Constructor to use when building a ScaleAnimation from code
     *
     * @param fromX Horizontal scaling factor to apply at the start of the
     *        animation
     * @param toX Horizontal scaling factor to apply at the end of the animation
     * @param fromY Vertical scaling factor to apply at the start of the
     *        animation
     * @param toY Vertical scaling factor to apply at the end of the animation
     */
    public ScaleAnimation(float fromX, float toX, float fromY, float toY) {
        mResources = null;
        mFromX = fromX;
        mToX = toX;
        mFromY = fromY;
        mToY = toY;
        mPivotX = 0;
        mPivotY = 0;
    }

    //3.
    /**
     * Constructor to use when building a ScaleAnimation from code
     *
     * @param fromX Horizontal scaling factor to apply at the start of the
     *        animation
     * @param toX Horizontal scaling factor to apply at the end of the animation
     * @param fromY Vertical scaling factor to apply at the start of the
     *        animation
     * @param toY Vertical scaling factor to apply at the end of the animation
     * @param pivotX The X coordinate of the point about which the object is
     *        being scaled, specified as an absolute number where 0 is the left
     *        edge. (This point remains fixed while the object changes size.)
     * @param pivotY The Y coordinate of the point about which the object is
     *        being scaled, specified as an absolute number where 0 is the top
     *        edge. (This point remains fixed while the object changes size.)
     */
    public ScaleAnimation(float fromX, float toX, float fromY, float toY,
            float pivotX, float pivotY) {

      }
```

```
    // 4.
    /**
     * Constructor to use when building a ScaleAnimation from code
     *
     * @param fromX Horizontal scaling factor to apply at the start of the
     *        animation
     * @param toX Horizontal scaling factor to apply at the end of the animation
     * @param fromY Vertical scaling factor to apply at the start of the
     *        animation
     * @param toY Vertical scaling factor to apply at the end of the animation
     * @param pivotXType Specifies how pivotXValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param pivotXValue The X coordinate of the point about which the object
     *        is being scaled, specified as an absolute number where 0 is the
     *        left edge. (This point remains fixed while the object changes
     *        size.) This value can either be an absolute number if pivotXType
     *        is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
     * @param pivotYType Specifies how pivotYValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param pivotYValue The Y coordinate of the point about which the object
     *        is being scaled, specified as an absolute number where 0 is the
     *        top edge. (This point remains fixed while the object changes
     *        size.) This value can either be an absolute number if pivotYType
     *        is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
     */
    public ScaleAnimation(float fromX, float toX, float fromY, float toY,
            int pivotXType, float pivotXValue, int pivotYType, float pivotYValue) {

     }
```

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：

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="3000"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fromDegrees="0"
        android:toDegrees="90">


</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：

```
            RotateAnimation rotateAnimation = new RotateAnimation(0,180);
        rotateAnimation.setDuration(3000);
        mImage.startAnimation(rotateAnimation);
```

构造方法：

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

```
     * Constructor used when a RotateAnimation is loaded from a resource.
     *
     * @param context Application context to use
     * @param attrs Attribute set from which to read values
     */
    public RotateAnimation(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    // 2.
    /**
     * Constructor to use when building a RotateAnimation from code.
     * Default pivotX/pivotY point is (0,0).
     *
     * @param fromDegrees Rotation offset to apply at the start of the
     *        animation.
     *
     * @param toDegrees Rotation offset to apply at the end of the animation.
     */
    public RotateAnimation(float fromDegrees, float toDegrees) {

    }

    // 3.
    /**
     * Constructor to use when building a RotateAnimation from code
     *
     * @param fromDegrees Rotation offset to apply at the start of the
     *        animation.
     *
     * @param toDegrees Rotation offset to apply at the end of the animation.
     *
     * @param pivotX The X coordinate of the point about which the object is
     *        being rotated, specified as an absolute number where 0 is the left
     *        edge.
     * @param pivotY The Y coordinate of the point about which the object is
     *        being rotated, specified as an absolute number where 0 is the top
     *        edge.
     */
    public RotateAnimation(float fromDegrees, float toDegrees, float pivotX, float pivotY) {
        mFromDegrees = fromDegrees;
        mToDegrees = toDegrees;

        mPivotXType = ABSOLUTE;
        mPivotYType = ABSOLUTE;
        mPivotXValue = pivotX;
        mPivotYValue = pivotY;
        initializePivotPoint();
    }

    //4.
    /**
     * Constructor to use when building a RotateAnimation from code
     *
     * @param fromDegrees Rotation offset to apply at the start of the
     *        animation.
     *
     * @param toDegrees Rotation offset to apply at the end of the animation.
     *
     * @param pivotXType Specifies how pivotXValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param pivotXValue The X coordinate of the point about which the object
     *        is being rotated, specified as an absolute number where 0 is the
     *        left edge. This value can either be an absolute number if
     *        pivotXType is ABSOLUTE, or a percentage (where 1.0 is 100%)
     *        otherwise.
     * @param pivotYType Specifies how pivotYValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param pivotYValue The Y coordinate of the point about which the object
     *        is being rotated, specified as an absolute number where 0 is the
     *        top edge. This value can either be an absolute number if
     *        pivotYType is ABSOLUTE, or a percentage (where 1.0 is 100%)
     *        otherwise.
     */
    public RotateAnimation(float fromDegrees, float toDegrees, int pivotXType, float pivotXValue,
            int pivotYType, float pivotYValue) {
        mFromDegrees = fromDegrees;
        mToDegrees = toDegrees;

```

```
        mPivotXValue = pivotXValue;
        mPivotXType = pivotXType;
        mPivotYValue = pivotYValue;
        mPivotYType = pivotYType;
        initializePivotPoint();
    }
```

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:

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
           android:fillAfter="true"
           android:duration="3000"
           android:fromXDelta="50%"
           android:fromYDelta="50%"
           android:toXDelta="50%p"
           android:toYDelta="50%p">
</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：

```
        TranslateAnimation translateAnimation = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0,Animation.RELATIVE_TO_SEL
        translateAnimation.setDuration(3000);
        translateAnimation.setFillAfter(true);
        mImage.startAnimation(translateAnimation);
```

构造函数：

```
    // 1.
    /**
     * Constructor used when a TranslateAnimation is loaded from a resource.
     *
     * @param context Application context to use
     * @param attrs Attribute set from which to read values
     */
    public TranslateAnimation(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

     // 2.
     /**
     * Constructor to use when building a TranslateAnimation from code
     *
     * @param fromXDelta Change in X coordinate to apply at the start of the
     *        animation
     * @param toXDelta Change in X coordinate to apply at the end of the
     *        animation
     * @param fromYDelta Change in Y coordinate to apply at the start of the
     *        animation
     * @param toYDelta Change in Y coordinate to apply at the end of the
     *        animation
     */
    public TranslateAnimation(float fromXDelta, float toXDelta, float fromYDelta, float toYDelta) {
        mFromXValue = fromXDelta;
        mToXValue = toXDelta;
        mFromYValue = fromYDelta;
```

```
        mToYValue = toYDelta;

        mFromXType = ABSOLUTE;
        mToXType = ABSOLUTE;
        mFromYType = ABSOLUTE;
        mToYType = ABSOLUTE;
    }

    // 3.
    /**
     * Constructor to use when building a TranslateAnimation from code
     *
     * @param fromXType Specifies how fromXValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param fromXValue Change in X coordinate to apply at the start of the
     *        animation. This value can either be an absolute number if fromXType
     *        is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
     * @param toXType Specifies how toXValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param toXValue Change in X coordinate to apply at the end of the
     *        animation. This value can either be an absolute number if toXType
     *        is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
     * @param fromYType Specifies how fromYValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param fromYValue Change in Y coordinate to apply at the start of the
     *        animation. This value can either be an absolute number if fromYType
     *        is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
     * @param toYType Specifies how toYValue should be interpreted. One of
     *        Animation.ABSOLUTE, Animation.RELATIVE_TO_SELF, or
     *        Animation.RELATIVE_TO_PARENT.
     * @param toYValue Change in Y coordinate to apply at the end of the
     *        animation. This value can either be an absolute number if toYType
     *        is ABSOLUTE, or a percentage (where 1.0 is 100%) otherwise.
     */
    public TranslateAnimation(int fromXType, float fromXValue, int toXType, float toXValue,
            int fromYType, float fromYValue, int toYType, float toYValue) {

        mFromXValue = fromXValue;
        mToXValue = toXValue;
        mFromYValue = fromYValue;
        mToYValue = toYValue;

        mFromXType = fromXType;
        mToXType = toXType;
        mFromYType = fromYType;
        mToYType = toYType;
    }
```

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：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:shareInterpolator="false">
    <scale
        android:duration="2000"
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:interpolator="@android:anim/linear_interpolator"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="0"
        android:toYScale="0"/>

    <set
        android:duration="2000"
        android:interpolator="@android:anim/decelerate_interpolator"
        android:shareInterpolator="true"
        android:startOffset="2000">
        <scale
            android:fromXScale="0"
```

```
            android:fromYScale="0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:toXScale="1"
            android:toYScale="1"/>
        <rotate
            android:fromDegrees="0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:toDegrees="180"/>
    </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

```
        AnimationSet animationSet1 = new AnimationSet(false);//一级集合
        ScaleAnimation scaleAnimation1 = new ScaleAnimation(1, 1.4f, 1, 1.4f, Animation.RELATIVE_TO_SELF, 0.5f,Animation.RELA

        AnimationSet animationSet2 = new AnimationSet(true);//二级集合
        ScaleAnimation scaleAnimation2 = new ScaleAnimation(1.4f, 0, 1.4f, 0, Animation.RELATIVE_TO_SELF, 0.5f,Animation.RELA
        RotateAnimation rotateAnimation = new RotateAnimation(0, 180, Animation.RELATIVE_TO_SELF, 0.5f,Animation.RELATIVE_TO_

        animationSet2.addAnimation(scaleAnimation2);
        animationSet2.addAnimation(rotateAnimation);
        animationSet2.setInterpolator(new DecelerateInterpolator());
        animationSet2.setDuration(2000);
        animationSet2.setStartOffset(2000);

        animationSet1.addAnimation(scaleAnimation1);
        animationSet1.addAnimation(animationSet2);
        animationSet1.setInterpolator(new AccelerateDecelerateInterpolator());
        animationSet1.setDuration(2000);

        mImage.startAnimation(animationSet1);
```

构造函数：

```
    // 1.
    /**
     * Constructor used when an AnimationSet is loaded from a resource.
     *
     * @param context Application context to use
     * @param attrs Attribute set from which to read values
     */
    public AnimationSet(Context context, AttributeSet attrs) {
        super(context, attrs);

        TypedArray a =
            context.obtainStyledAttributes(attrs, com.android.internal.R.styleable.AnimationSet);

        setFlag(PROPERTY_SHARE_INTERPOLATOR_MASK,
                a.getBoolean(com.android.internal.R.styleable.AnimationSet_shareInterpolator, true));
        init();

        if (context.getApplicationInfo().targetSdkVersion >=
                Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
            if (a.hasValue(com.android.internal.R.styleable.AnimationSet_duration)) {
                mFlags |= PROPERTY_DURATION_MASK;
            }
            if (a.hasValue(com.android.internal.R.styleable.AnimationSet_fillBefore)) {
                mFlags |= PROPERTY_FILL_BEFORE_MASK;
            }
```

```
            if (a.hasValue(com.android.internal.R.styleable.AnimationSet_fillAfter)) {
                mFlags |= PROPERTY_FILL_AFTER_MASK;
            }
            if (a.hasValue(com.android.internal.R.styleable.AnimationSet_repeatMode)) {
                mFlags |= PROPERTY_REPEAT_MODE_MASK;
            }
            if (a.hasValue(com.android.internal.R.styleable.AnimationSet_startOffset)) {
                mFlags |= PROPERTY_START_OFFSET_MASK;
            }
        }

        a.recycle();
    }

    // 2.
    /**
     * Constructor to use when building an AnimationSet from code
     *
     * @param shareInterpolator Pass true if all of the animations in this set
     *        should use the interpolator associated with this AnimationSet.
     *        Pass false if each animation should use its own interpolator.
     */
    public AnimationSet(boolean shareInterpolator) {
        setFlag(PROPERTY_SHARE_INTERPOLATOR_MASK, shareInterpolator);
        init();
    }
```

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：

```
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：

```
    @Override
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        mFromXDelta = resolveSize(mFromXType, mFromXValue, width, parentWidth);
        mToXDelta = resolveSize(mToXType, mToXValue, width, parentWidth);
        mFromYDelta = resolveSize(mFromYType, mFromYValue, height, parentHeight);
        mToYDelta = resolveSize(mToYType, mToYValue, height, parentHeight);
    }

    // 然后再看resolveSize()方法
    protected float resolveSize(int type, float value, int size, int parentSize) {
```

```
        switch (type) {
            case ABSOLUTE:
                return value;
            case RELATIVE_TO_SELF:
                return size * value;
            case RELATIVE_TO_PARENT:
                return parentSize * value;
            default:
                return value;
        }
    }

// 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：

    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        float dx = mFromXDelta;
        float dy = mFromYDelta;
        if (mFromXDelta != mToXDelta) {
            dx = mFromXDelta + ((mToXDelta - mFromXDelta) * interpolatedTime);
        }
        if (mFromYDelta != mToYDelta) {
            dy = mFromYDelta + ((mToYDelta - mFromYDelta) * interpolatedTime);
        }
        t.getMatrix().setTranslate(dx, dy);
    }
```

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`)

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

    <item
       android:drawable="@drawable/sign_icon_1"
       android:duration="50" />
    <item
       android:drawable="@drawable/sign_icon_2"
       android:duration="50" />
     <item
       android:drawable="@drawable/sign_icon_3"
       android:duration="50" />
     <item
       android:drawable="@drawable/sign_icon_4"
       android:duration="50" />

</animation-list>
```

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

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">

    <ImageView
        android:id="@+id/image"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/frame_view" />
</LinearLayout>
```

开启动画:

```
        ImageView imageView = findViewById(R.id.image);
        AnimationDrawable animationDrawable = (AnimationDrawable) imageView.getDrawable();
        if (animationDrawable != null) {
            animationDrawable.start();
        }

```

三、资源

本文章源码