插值器

1、基本介绍

前面遇到过插值器的概念，不过没有介绍，所谓插值器其实就是用来定义动画的执行速度的，因为我们在动画设置了duration，但是在整个动画执行的过程都是按照同样的速度执行的，也就是线性速度执行，不同的差值器有着不同的执行速度变化，让动画执行变得更加有趣（**因为我们定义动画只是定义了起始和结束的状态，因此中间的状态变化是系统完成的，可以认为中间的变化就是系统插入的变化，因此称为插值器。个人理解**）

2、类型

系统提供9种类型的插值器

（1） AccelerateInterpolator **加速**

动画变化速度越来越快

（2）DecelerateInterpolator **减速**

动画变化速度越来越慢

（3）LinearInterpolator **匀速**

动画变化速度不变（默认的）

（4）CycleInterpolator **周期运动**

起点运动到结尾，然后又从结尾运动回起点，接着再从起点运动到结尾的反向状态，再回到起点状态

（比如要向右移动100，首先向右移动100，然后回到起点，接着向左移动100，回到起点）

（5）BounceInterpolator **碰撞反弹**

就和小球落地碰撞一样，弹起，再落地，再弹起，再落地...

（6）AccelerateDecelerateInterpolator **先加速后减速**

动画变化速度先变快，后变慢

（7）AnticipateInterpolator **先收缩再加速**

就像拉弓一样，先往后倒退一点，再加速

（8）OvershootInterpolator **匀速，最后超出终点一部分，再回到终点**

匀速的变化到结尾状态，但是会比结尾状态再继续变化一部分，最后再回到结尾状态

（9）AnticipateOvershootInterpolator　明显就是 7.8的组合

就是7,8状态变化的集合。先后退，然后加速变化到结尾状态，再超过结尾状态继续变化一部分，最后再回到结尾状态

3、使用

（1）xml方式

```
<set
    android:duration="500"
    android:interpolator="@android:anim/accelerate_interpolator"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:fromYDelta="0"
        android:toXDelta="400"
        android:toYDelta="0"/>
</set>
```

只需要选择对应的插值器即可

然后同样就是利用AnimationUtils加载即可

（2）java方式

```
TranslateAnimation animation = new TranslateAnimation(0,700,0,0);
animation.setDuration(1500);
animation.setInterpolator(new AccelerateInterpolator());
```

直接new 出对应的插值器类即可，当然可能会有些需要传入参数的构造器，具体的参数暂不介绍

4、自定义插值器

（1）xml方式（类似于复写）

```
<overshootInterpolator
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:tension="5"/>
```

步骤：

根据想要自定义的插值器类型，选择对应插值器的标签

然后根据该插值器提供的属性，进行设置

上面：

自定义Overshoot的插值器，因此选择OvershootInterpolator

接着修改了tension的值，表示超过结尾状态的一个值，默认是2,

最终效果：

我们自定义的Overshoot插值器，在超过结尾状态的效果大于默认的overShoot的效果（比如移动，默认可能是超过结尾100，我们的可能超过300，数字是随便写的，具体的变化需要看源码的数学公式）

其他插值器：

步骤都是一样的，只是某些插值器没有提供对应的属性，因此可能没办法进行自定义（或者个人能力有限）

（2）java代码实现

自定义类实现Interpolator接口，然后重写getInterpolation(int input)方法
input是动画时间， 0 <= input <=1 ,表示动画的开始和结束
我们需要返回一个float值，根据input的值对返回值不断的改变，就会导致动画的变化速度发生改变，具体可能会涉及到数学公式