如果想同时改变多个属性，有两种思路：

1. 构造多个对象Animator ，（ Animator可以是ValueAnimator、ObjectAnimator和AnimatorSet）然后最后把它们放在一个AnimatorSet中。
2. 把多个属性的改变放在同一个 ValueAnimator 中（ObjectAnimator也是 ValueAnimator）。而这就要借助PropertyValuesHolder。本文主要讲这种方法。

**PropertyValuesHolder**

　　PropertyValuesHolder是API Level 11加进来的。根据名字就可以判断出它是某种属性的持有者。使用工厂方法构造PropertyValuesHolder对象，指定属性名和一系列属性值。

　　代码例子：MultiPropertyAnimation中：

// 第二个小球：加速下落并且alpha变化

ball = balls.get(1);

// 利用ofFloat工厂方法构造PropertyValuesHolder类型对象，控制y属性

PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("y",

ball.getY(), getHeight() - BALL\_SIZE);

// 利用ofFloat工厂方法构造另一个PropertyValuesHolder类型对象，控制alpha属性

PropertyValuesHolder pvhAlpha = PropertyValuesHolder.ofFloat(

"alpha", 1.0f, 0f);

// 利用ofPropertyValuesHolder方法来构造ObjectAnimator对象

// 把多个属性变化结合到一个动画中去

ObjectAnimator yAlphaBouncer = ObjectAnimator

.ofPropertyValuesHolder(ball, pvhY, pvhAlpha)

.setDuration(DURATION / 2);

yAlphaBouncer.setInterpolator(new AccelerateInterpolator());

yAlphaBouncer.setRepeatCount(1);

yAlphaBouncer.setRepeatMode(ValueAnimator.REVERSE);

**关键帧Keyframe**

　　PropertyValuesHolder的工厂方法里面，除了整形ofInt()、浮点型ofFloat()、Object类型ofObject()之外，还有一种：ofKeyframe()。**Keyframe类型对象由一个time/value对组成**，定义了指定时间点的指定值。每一个keyframe还可以拥有自己的interpolator，控制了前一个关键帧到这一个关键帧之间的时间动画行为。

　　Keyframe 对象的构造也用是工厂方法：ofInt(), ofFloat(), or ofObject()。

　　Keyframe对象构造完之后就可以用 ofKeyframe()工厂方法来构造PropertyValuesHolder对象。

　　代码例子：MultiPropertyAnimation中：

// 第四个小球：利用关键帧实现曲线运动

ball = balls.get(3);

// 属性1：Y坐标运动：下落

pvhY = PropertyValuesHolder.ofFloat("y", ball.getY(),

getHeight() - BALL\_SIZE);

float ballX = ball.getX();

// 三个关键帧

Keyframe kf0 = Keyframe.ofFloat(0f, ballX);

Keyframe kf1 = Keyframe.ofFloat(.5f, ballX + 100f);

Keyframe kf2 = Keyframe.ofFloat(1f, ballX + 50f);

// 属性2：X坐标运动：曲折

// 用三个关键帧构造PropertyValuesHolder对象

PropertyValuesHolder pvhX = PropertyValuesHolder.ofKeyframe(

"x", kf0, kf1, kf2);

// 再用两个PropertyValuesHolder对象构造一个ObjectAnimator对象

ObjectAnimator yxBouncer = ObjectAnimator

.ofPropertyValuesHolder(ball, pvhY, pvhX).setDuration(

DURATION / 2);

yxBouncer.setRepeatCount(1);

yxBouncer.setRepeatMode(ValueAnimator.REVERSE);

**View的多属性动画：使用ViewPropertyAnimator**

　　ViewPropertyAnimator是API Level 12引进的。它是用来做针对View对象的多个属性动画功能。（前面的PropertyValuesHolder对象是针对所有对象的，范围更广）。

　　如果要同时变换一个View的多个属性的话，ViewPropertyAnimator提供了一种更方便和更适合的方法。　　而且由于多个属性的invalidate方法调用统一管理，而不是之前的分别调用，所以还会有一些性能优化。

　　注意 ViewPropertyAnimator 这个类的对象不是由调用者构造的，而是通过View类的animate()方法返回的。

　　比如下面的代码对比：给同一个View实现同一个动画效果：

用多个ObjectAnimator对象：

ObjectAnimator animX = ObjectAnimator.ofFloat(myView, "x", 50f);

ObjectAnimator animY = ObjectAnimator.ofFloat(myView, "y", 100f);

AnimatorSet animSetXY = new AnimatorSet();

animSetXY.playTogether(animX, animY);

animSetXY.start();

用一个ObjectAnimator对象加多个PropertyValuesHolder:

PropertyValuesHolder pvhX = PropertyValuesHolder.ofFloat("x", 50f);

PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("y", 100f);

ObjectAnimator.ofPropertyValuesHolder(myView, pvhX, pvyY).start();

用ViewPropertyAnimator：

myView.animate().x(50f).y(100f);