## Android的属性动画ObjectAnimator

网站：

<https://blog.csdn.net/xiaochuanding/article/details/73290917>

使用补间动画会造成控件移动到目的地但是控件的热点（产生事件的范围）仍然在原地

属性动画分为ObjectAnimator和ValueAnimator，前者继承后者

使用方法：

通过：

ObjectAnimator.ofFloat(控件名,动画属性,动画开始变化的初始值，动画结束的值).setDuration(1000).start();//来创建一个动画。

例如:

这是让一个控件只执行一个动作

|  |
| --- |
| 平移动画：  //private Imageview ivBtn;  ObjectAnimator objectAnimator = ObjectAnimator.*ofFloat*(**ivBtn**, **"translationX"**, 0f, 135f); objectAnimator.setDuration(200); objectAnimator.start(); |

|  |
| --- |
| 旋转动画：  //private Imageview ivBtn;  ObjectAnimator objectAnimator = ObjectAnimator.*ofFloat*(**ivBtn**, **"rotation"**, 0f, 135f); objectAnimator.setDuration(200); objectAnimator.start(); |

这是让控件执行一套动作：也就是动画集

操作：

|  |
| --- |
| //封装平移X的动画  PropertyValuesHolder pro = PropertyValuesHolder.ofFloat("translationX",0F,100F);　　　　　　　　　　//封装旋转360的动画  PropertyValuesHolder pro2 = PropertyValuesHolder.ofFloat("rotation",0F,360F);  //最后组合  ObjectAnimator.ofPropertyValuesHolder(**ivBtn**,pro,pro1,pro2).setDuration(200).start();;来为某个控件设置以上动画，其他控件也可 |

对多个控件一起操作

操作:

|  |
| --- |
| 先实例化这个属性动画集合 AnimatorSet set = new AnimatorSet().  　　　　　　通过set.play(<animator1>).with(<animator2>);来表示让Animator1和Animator2一起执行，  　　　　　　通过set.play(<animator1>).after(<animator2>)来表示让Animator1在Animator2后执行  　　　　　　通过set.play(<animator1>).before(<animator2>)来表示让Animator1在Animator2之前执行、 |

### ****自定义ObjectAnimator属性****

操作

|  |  |  |
| --- | --- | --- |
| ObjectAnimator objectAnimator = ObjectAnimator.ofInt(mCircleView,"pointRadius",0,200,100,200,50);  objectAnimator.setDuration(1000);  objectAnimator.start();  然后自定义控件  自定义Point   |  | | --- | | public class Ponit {  private int mRadius;  public Ponit() {  }  public Ponit(int mRadius) {  this.mRadius = mRadius;  }  public int getRadius() {  return mRadius;  }  public void setRadius(int mRadius) {  this.mRadius = mRadius;  }  } |   自定义view   |  | | --- | | public class CircleView extends View {  private Ponit mCurrentPoint = new Ponit();  private Paint mPiant ;  private int mScreenWidth;//屏幕宽度  public CircleView(Context context) {  this(context,null);  }  public CircleView(Context context, @Nullable AttributeSet attrs) {  this(context, attrs,0);  }  public CircleView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {  super(context, attrs, defStyleAttr);  mPiant = new Paint(Paint.ANTI\_ALIAS\_FLAG);  mPiant.setColor(Color.RED);  mPiant.setStyle(Paint.Style.FILL);  mScreenWidth = ((WindowManager)getContext().getSystemService(Context.WINDOW\_SERVICE)).getDefaultDisplay().getWidth();  }  @Override  protected void onDraw(Canvas canvas) {  if (mCurrentPoint != null){  canvas.drawCircle(mScreenWidth/2,getY()+getPaddingTop(),mCurrentPoint.getRadius(),mPiant);  }  }  public int getPointRadius(){ //这个get方法只作为默认值出现在这里的  return 50;  }  public void setPointRadius(int radius){//这里set方法必须和ObjectAnimator中的属性值对应  mCurrentPoint.setRadius(radius);  invalidate();  }  } | |