属性动画

# 一. ValueAnimator

这个Animation是针对值的！ValueAnimator不会对控件做任何操作，我们可以给它设定从哪个值运动到哪个值，通过监听这些值的渐变过程来自己操作控件

## 创建实例

调用ValueAnimation类中的ofInt(int...values)、ofFloat(String propertyName,float...values)等静态方法实例化ValueAnimator对象，并设置目标属性的属性名、初始值或结束值等值;

## 设置数值变化监听器

调用addUpdateListener(AnimatorUpdateListener mListener)方法为ValueAnimator对象设置属性变化的监听器;

## 创建自定义的Interpolator（可选）

调用setInterpolator(TimeInterpolator value)为ValueAniamtor设置自定义的Interpolator;(可选，不设置默认为缺省值)

## 创建自定义的TypeEvaluator（可选）

调用setEvaluator(TypeEvaluator value)为ValueAnimator设置自定义的TypeEvaluator数值发生器;(可选，不设置默认为缺省值)。

初始化时指定：ValueAnimator valueanim1 = ValueAnimator.ofObject(new TypeEvaluator<PointF>() {…}并覆写evaluate方法。

## 添加属性值变化监听器

在AnimatorUpdateListener 中的实现方法为目标对象的属性设置计算好的属性值。

## 设置动画的持续时间

## 是否重复及重复次数等属性

## 为ValueAnimator设置目标对象并开始执行动画。

# 二. ObjectAnimator

该类作为ValueAnimator的子类不仅继承了ValueAnimator的所有方法和特性，并且还封装很多实用的方法，方便开发人员快速实现动画。同时，由于属性值会自动更新，使用ObjectAnimator实现动画不需要像ValueAnimator那样必须实现 ValueAnimator.AnimatorUpdateListener ，因此实现任意对象的动画显示就更加容易了。我们在大部分的开发工作中，都会使用ObjectAnimator而非ValueAnimator实现我们所需的动画效果。

View Animation实现View对象动画的特点，即View Animation本身是通过改变View的绘制方式来实现动画的，View对象本身的属性值并没有改变，对象仍然停留在原始位置。那Android为了消除这一弊病，在 Android 3.0 中给 View 增加了一些新的属性以及相应的 getter、setter 方法。Property Animation系统可以通过修改 View 对象实际的属性值来实现屏幕上的动画效果。此外，当属性值发生变化时，Views 也会自动调用 invalidate() 方法来刷新屏幕。

## 常用动画的属性：

### translationX 和 translationY

这两个属性控制着 View 的屏幕位置坐标变化量，以layout 容器的左上角为坐标原点;

### rotation、rotationX 和 rotationY

这三个属性控制着2D旋转角度（rotation属性）和围绕某枢轴点的 3D 旋转角度;

### scaleX、scaleY

这两个属性控制着View 围绕某枢轴点的 2D 缩放比例;

### pivotX 和 pivotY

这两个属性控制着枢轴点的位置，前述的旋转和缩放都是以此点为中心展开的,缺省的枢轴点是 View 对象的中心点;

### x 和 y

这是指 View 在容器内的最终位置，等于View左上角相对于容器的坐标加上 translationX 和 translationY 后的值;

### alpha

表示 View 的 alpha 透明度。缺省值为 1 （不透明），为 0 则表示完全透明（看不见）;

# 三. AnimatorUpdateListener

addUpdateListener(AnimatorUpdateListener mListener)方法为ValueAnimator对象设置属性变化的监听器，可以在计算他们值的时候修改他们。

你可以在ValueAnimator中通过定义监听器来在动画的持续过程中处理重要的事件，例如帧更新。当你实现这些监听器的时候，你可以通过调用getAnimatedValue()来获取当前帧计算后的值。

# 四. AnimatorListenerAdapte

此方法简化了ObjectAnimator实例的addListener方法，addListener方法若传参为AnimationListener则必须实现指定的四个方法：onAnimationStart、onAnimationEnd、onAnimationCancel、OnAnimationRepeat。但这四个方法实际开发中有时不是每个方法都用到，这时可以传参AnimatorListenerAdapter，此接口可以按需选择要覆写的方法。

# 五. PropertyValuesHolder

将多个动画同时作用于View

PropertyValuesHolder holder = PropertyValuesHolder.ofFloat("translationX",0f,200f);

PropertyValuesHolder holder1 = PropertyValuesHolder.ofFloat("translationY",0f,200f);

ObjectAnimator.ofPropertyValuesHolder(imageView,holder,holder1,holder2).setDuration(1500).start();

# 六. AnimatorSet

将多个动画存于Set集合，并指定播放顺序：同时、指定顺序、等

ObjectAnimator anim1 = ObjectAnimator.ofFloat(imageView,"translationX",0f,200f).setDuration(1500);

ObjectAnimator anim2 = ObjectAnimator.ofFloat(imageView,"translationY",0f,200f).setDuration(1500);

AnimatorSet animSet = new AnimatorSet();

animSet.play(anim1).with(anim2);

animSet.play(anim3).before(anim2);

//animSet.playTogether(anim1,anim2,anim3);

//animSet.playSequentially(anim1,anim2,anim3);

animSet.start();

# 七. Interprolators

加速器

http://androidigging.blog.51cto.com/2753843/1427128

AccelerateDecelerateInterpolator 在动画开始与结束的地方速率改变比较慢，在中间的时候加速

AccelerateInterpolator 在动画开始的地方速率改变比较慢，然后开始加速

AnticipateInterpolator 开始的时候向后然后向前甩

AnticipateOvershootInterpolator 开始的时候向后然后向前甩一定值后返回最后的值

BounceInterpolator 动画结束的时候弹起

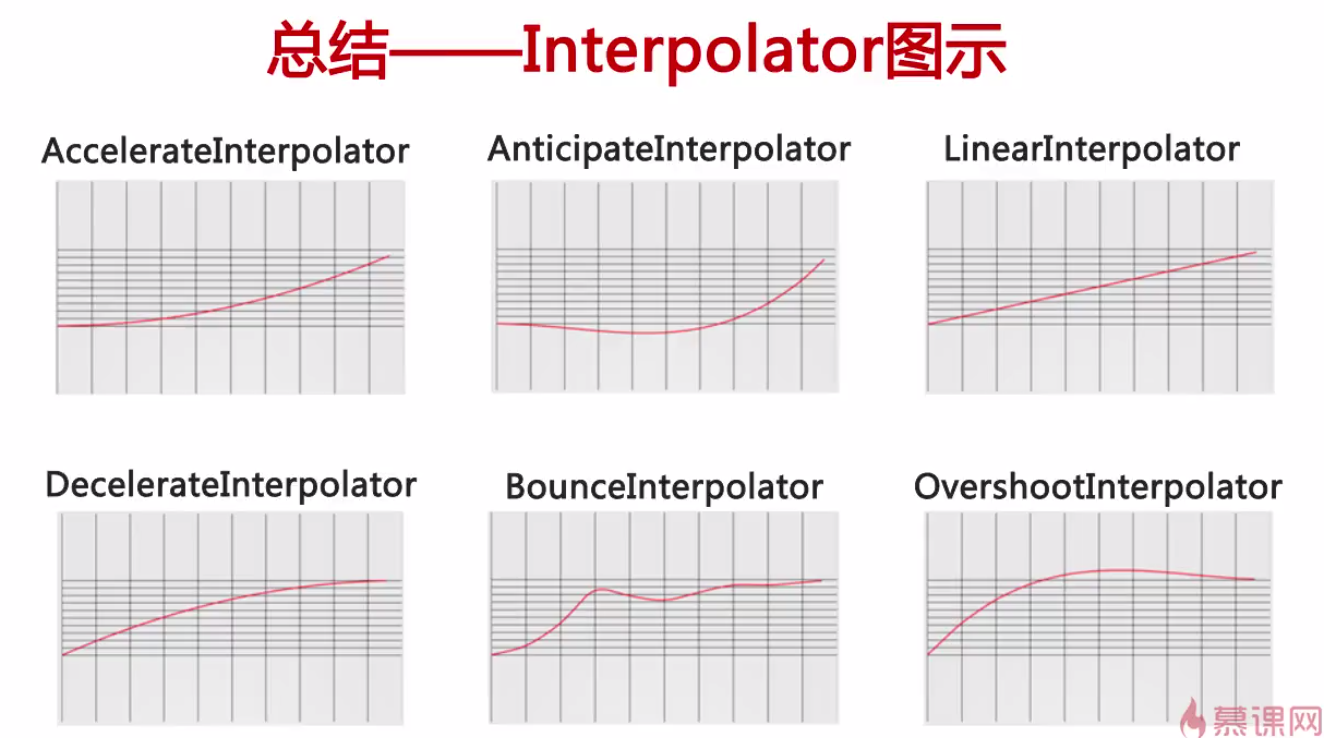
CycleInterpolator 动画循环播放特定的次数，速率改变沿着正弦曲线

DecelerateInterpolator 在动画开始的地方快然后慢

LinearInterpolator 以常量速率改变

OvershootInterpolator 向前甩一定值后再回到原来位置

package com.example.ai.application.animation;



# 八. 四种方法为View添加动画

### Animation类

TranslateAnimation animation = new TranslateAnimation(0,300,0,0); imageView.startAnimation(animation);

### ObjectAnimator类

ObjectAnimator.ofFloat(imageView,"translationX",0f,200f).setDuration(1500).start();

### PropertyValuesHolder包装

PropertyValuesHolder holder = PropertyValuesHolder.ofFloat("translationX",0f,200f);

PropertyValuesHolder holder1 = PropertyValuesHolder.ofFloat("translationY",0f,200f);

ObjectAnimator.ofPropertyValuesHolder(imageView,holder,holder1,holder2).setDuration(1500).start();

### AnimatorSet包装

ObjectAnimator anim1 = ObjectAnimator.ofFloat(imageView,"translationX",0f,200f).setDuration(1500);

ObjectAnimator anim2 = ObjectAnimator.ofFloat(imageView,"translationY",0f,200f).setDuration(1500);

AnimatorSet animSet = new AnimatorSet();

animSet.play(anim1).with(anim2);

animSet.play(anim3).before(anim2);

//animSet.playTogether(anim1,anim2,anim3);

//animSet.playSequentially(anim1,anim2,anim3);

animSet.start();

# 九. 例子

import android.animation.Animator;

import android.animation.AnimatorListenerAdapter;

import android.animation.AnimatorSet;

import android.animation.ObjectAnimator;

import android.animation.ValueAnimator;

import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.view.animation.OvershootInterpolator;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.Toast;

import com.example.ai.application.R;

import java.util.ArrayList;

/\*\*

\*

\*/

public class AnimActivity extends Activity implements View.OnClickListener{

private boolean flag = true;

private int[] res = {

R.id.image\_01,

R.id.image\_02,

R.id.image\_03,

R.id.image\_04,

R.id.image\_05,

R.id.image\_06,

};

private ArrayList<ImageView> imageViews = new ArrayList<>();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_anim);

ImageView open = (ImageView) findViewById(R.id.open);

open.setOnClickListener(this);

for (int i = 0; i < res.length; i++) {

ImageView imageView = (ImageView) findViewById(res[i]);

imageView.setOnClickListener(this);

imageViews.add(imageView);

}

}

//按钮点击事件——ValueAnimator

public void count(View view) {

final Button button = (Button) view;

ValueAnimator valueanim = ValueAnimator.ofInt(0,100);

valueanim.setDuration(5000);

valueanim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator valueAnimator) {

Integer va = (Integer) valueAnimator.getAnimatedValue();

button.setText(va+"");

}

});

valueanim.start();

// ValueAnimator valueanim1 = ValueAnimator.ofObject(new TypeEvaluator<PointF>() {

//

// @Override

// public PointF evaluate(float v, PointF pointF, PointF t1) {

// return null;

// }

// });

}

//按钮点击事件

public void buttonClick(View view) {

ImageView imageView = (ImageView) findViewById(R.id.imageView);

//方法一

// TranslateAnimation animation = new TranslateAnimation(0,300,0,0);

// animation.setDuration(2000);

// animation.setFillAfter(true);

// imageView.startAnimation(animation);

//方法二

// ObjectAnimator.ofFloat(imageView,"translationX",0f,200f).setDuration(1500).start();

// ObjectAnimator.ofFloat(imageView,"translationY",0f,200f).setDuration(1500).start();

// ObjectAnimator.ofFloat(imageView,"rotation",0f,360f).setDuration(1500).start();

//方法三

// PropertyValuesHolder holder = PropertyValuesHolder.ofFloat("translationX",0f,200f);

// PropertyValuesHolder holder1 = PropertyValuesHolder.ofFloat("translationY",0f,200f);

// PropertyValuesHolder holder2 = PropertyValuesHolder.ofFloat("rotation",0f,360f);

// ObjectAnimator.ofPropertyValuesHolder(imageView,holder,holder1,holder2).setDuration(1500).start();

//方法四

ObjectAnimator anim1 = ObjectAnimator.ofFloat(imageView,"translationX",0f,200f).setDuration(1500);

ObjectAnimator anim2 = ObjectAnimator.ofFloat(imageView,"translationY",0f,200f).setDuration(1500);

ObjectAnimator anim3 = ObjectAnimator.ofFloat(imageView,"rotation",0f,360f).setDuration(1500);

AnimatorSet animSet = new AnimatorSet();

animSet.play(anim1).with(anim2);

animSet.play(anim3).before(anim2);

//animSet.playTogether(anim1,anim2,anim3);

//animSet.playSequentially(anim1,anim2,anim3);

animSet.start();

}

public void click(View view) {

ObjectAnimator anim = ObjectAnimator.ofFloat(view,"alpha",0f,1f).setDuration(1500);

anim.addListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

super.onAnimationEnd(animation);

Toast.makeText(AnimActivity.this, "onAnimationEnd", Toast.LENGTH\_SHORT).show();

}

});

// anim.addListener(new Animator.AnimatorListener() {

// @Override

// public void onAnimationStart(Animator animator) {

//

// }

//

// @Override

// public void onAnimationEnd(Animator animator) {

// Toast.makeText(AnimActivity.this, "onAnimationEnd", Toast.LENGTH\_SHORT).show();

// }

//

// @Override

// public void onAnimationCancel(Animator animator) {

//

// }

//

// @Override

// public void onAnimationRepeat(Animator animator) {

//

// }

// });

anim.start();

}

@Override

public void onClick(View view) {

switch (view.getId()){

case R.id.open:

if (flag){

ObjectAnimator.ofFloat(view,"rotation",0f,90f).setDuration(200).start();

openButtonList();

flag = false;

}

else {

ObjectAnimator.ofFloat(view,"rotation",90f,0f).setDuration(200).start();

closeButtonList();

flag = true;

}

break;

default:

Toast.makeText(AnimActivity.this, "click "+view.getId(), Toast.LENGTH\_SHORT).show();

break;

}

}

private void closeButtonList() {

for (int i = 0; i < res.length; i++) {

ObjectAnimator animator = ObjectAnimator.ofFloat(

imageViews.get(i),

"translationX",

i\*-100f,

0f).setDuration(300);

ObjectAnimator animator1 = ObjectAnimator.ofFloat(

imageViews.get(i),

"rotation",

0f,

360f).setDuration(300);

AnimatorSet set = new AnimatorSet();

set.play(animator).with(animator1);

//set.setStartDelay(500);

set.start();

}

}

private void openButtonList() {

for (int i = 0; i < res.length; i++) {

ObjectAnimator animator = ObjectAnimator.ofFloat(

imageViews.get(i),

"translationX",

0f,

i\*-100).setDuration(400);

animator.setInterpolator(new OvershootInterpolator());

ObjectAnimator animator1 = ObjectAnimator.ofFloat(

imageViews.get(i),

"rotation",

0f,

360f).setDuration(400);

AnimatorSet set = new AnimatorSet();

set.play(animator).with(animator1);

//set.setStartDelay(500);

set.start();

}

}

}