**[Animation 动画介绍和实现](http://www.cnblogs.com/yc-755909659/p/4290114.html)**

****1.Animation 动画类型****

Android的animation由四种类型组成：

XML中

|  |  |
| --- | --- |
| alph | 渐变透明度动画效果 |
| scale | 渐变尺寸伸缩动画效果 |
| translate | 画面转换位置移动动画效果 |
| rotate | 画面转移旋转动画效果 |

JavaCode中

|  |  |
| --- | --- |
| AlphaAnimation | 渐变透明度动画效果 |
| ScaleAnimation | 渐变尺寸伸缩动画效果 |
| TranslateAnimation | 画面转换位置移动动画效果 |
| RotateAnimation | 画面转移旋转动画效果 |

****2.Android动画模式****

Animation主要有两种动画模式：

一种是tweened animation(渐变动画)

|  |  |
| --- | --- |
| XML中 | JavaCode |
| alpha | AlphaAnimation |
| scale | ScaleAnimation |

一种是frame by frame(画面转换动画)

|  |  |
| --- | --- |
| XML中 | JavaCode |
| translate | TranslateAnimation |
| rotate | RotateAnimation |

**3.**如何在XML文件中定义动画****

步骤如下：

①新建 Android 项目

②在res目录中新建anim文件夹

③在anim目录中新建一个my\_anim.xml(注意文件名小写)

④在 my\_anim.xml 加入动画代码

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

<alpha />

<scale />

<translate />

<rotate /></set>

****4.Android动画解析--XML****  
4.1 alpha 渐变透明度动画效果

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<alpha

android:duration="1000"

android:fromAlpha="0.0"

android:toAlpha="1.0" />

<!--

透明度控制动画效果 alpha

浮点型值：

fromAlpha 属性为动画起始时透明度

toAlpha 属性为动画结束时透明度

说明:

0.0表示完全透明

1.0表示完全不透明

以上值取0.0-1.0之间的float数据类型的数字

长整型值：

duration 属性为动画持续时间

说明:

时间以毫秒为单位

--></set>

 4.2 scale 渐变尺寸伸缩动画效果

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

<scale

android:duration="1000"

android:fillAfter="false"

android:fromXScale="0.0"

android:fromYScale="0.0"

android:interpolator="@android:anim/accelerate\_decelerate\_interpolator"

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="1.4"

android:toYScale="1.4" />

</set><!--

尺寸伸缩动画效果 scale

属性：interpolator 指定一个动画的插入器

在我试验过程中，使用android.res.anim中的资源时候发现

有三种动画插入器:

accelerate\_decelerate\_interpolator 加速-减速 动画插入器

accelerate\_interpolator 加速-动画插入器

decelerate\_interpolator 减速- 动画插入器

其他的属于特定的动画效果

浮点型值：

fromXScale 属性为动画起始时 X坐标上的伸缩尺寸

toXScale 属性为动画结束时 X坐标上的伸缩尺寸

fromYScale 属性为动画起始时Y坐标上的伸缩尺寸

toYScale 属性为动画结束时Y坐标上的伸缩尺寸

说明:

以上四种属性值

0.0表示收缩到没有

1.0表示正常无伸缩

值小于1.0表示收缩

值大于1.0表示放大

pivotX 属性为动画相对于物件的X坐标的开始位置

pivotY 属性为动画相对于物件的Y坐标的开始位置

说明:

以上两个属性值 从0%-100%中取值

50%为物件的X或Y方向坐标上的中点位置

长整型值：

duration 属性为动画持续时间

说明: 时间以毫秒为单位

布尔型值:

fillAfter 属性 当设置为true ，该动画转化在动画结束后被应用

-->

4.3 translate 画面转换位置移动动画效果

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<translate

android:duration="2000"

android:fromXDelta="30"

android:fromYDelta="30"

android:toXDelta="-80"

android:toYDelta="300" />

<!--

translate 位置转移动画效果

整型值:

fromXDelta 属性为动画起始时 X坐标上的位置

toXDelta 属性为动画结束时 X坐标上的位置

fromYDelta 属性为动画起始时 Y坐标上的位置

toYDelta 属性为动画结束时 Y坐标上的位置

注意:

没有指定fromXType toXType fromYType toYType 时候，

默认是以自己为相对参照物

长整型值：

duration 属性为动画持续时间

说明: 时间以毫秒为单位

-->

</set>

4.4 rotate 画面转移旋转动画效果

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<rotate

android:duration="3000"

android:fromDegrees="0"

android:interpolator="@android:anim/accelerate\_decelerate\_interpolator"

android:pivotX="50%"

android:pivotY="50%"

android:toDegrees="+350" />

<!--

rotate 旋转动画效果

属性：interpolator 指定一个动画的插入器

在我试验过程中，使用android.res.anim中的资源时候发现

有三种动画插入器:

accelerate\_decelerate\_interpolator 加速-减速 动画插入器

accelerate\_interpolator 加速-动画插入器

decelerate\_interpolator 减速- 动画插入器

其他的属于特定的动画效果

浮点数型值:

fromDegrees 属性为动画起始时物件的角度

toDegrees 属性为动画结束时物件旋转的角度 可以大于360度

说明:

当角度为负数——表示逆时针旋转

当角度为正数——表示顺时针旋转

(负数from——to正数:顺时针旋转)

(负数from——to负数:逆时针旋转)

(正数from——to正数:顺时针旋转)

(正数from——to负数:逆时针旋转)

pivotX 属性为动画相对于物件的X坐标的开始位置

pivotY 属性为动画相对于物件的Y坐标的开始位置

说明: 以上两个属性值 从0%-100%中取值

50%为物件的X或Y方向坐标上的中点位置

长整型值：

duration 属性为动画持续时间

说明: 时间以毫秒为单位

-->

</set>

5.****如何使用XML中的动画效果****

public static Animation loadAnimation (Context context, int id)

//第一个参数Context为程序的上下文

//第二个参数id为动画XML文件的引用

//例子：

myAnimation= AnimationUtils.loadAnimation(this,R.anim.my\_anim);//使用AnimationUtils类的静态方法loadAnimation()来加载XML中的动画XML文件

6.****如何使用XML中的动画效果****

//在代码中定义 动画实例对象

private Animation myAnimation\_Alpha;

private Animation myAnimation\_Scale;

private Animation myAnimation\_Translate;

private Animation myAnimation\_Rotate;

//根据各自的构造方法来初始化一个实例对象

myAnimation\_Alpha=new AlphaAnimation(0.1f, 1.0f);

myAnimation\_Scale =new ScaleAnimation(0.0f, 1.4f, 0.0f, 1.4f,

Animation.RELATIVE\_TO\_SELF, 0.5f, Animation.RELATIVE\_TO\_SELF, 0.5f);

myAnimation\_Translate=new TranslateAnimation(30.0f, -80.0f, 30.0f, 300.0f);

myAnimation\_Rotate=new RotateAnimation(0.0f, +350.0f,

Animation.RELATIVE\_TO\_SELF,0.5f,Animation.RELATIVE\_TO\_SELF, 0.5f);

****7.Android动画解析--JavaCode****

****7.1 AlphaAnimation****

①　AlphaAnimation类对象定义

private AlphaAnimation myAnimation\_Alpha

②　AlphaAnimation类对象构造

//第一个参数fromAlpha为 动画开始时候透明度//第二个参数toAlpha为 动画结束时候透明度

AlphaAnimation(float fromAlpha, float toAlpha) //说明:0.0表示完全透明,1.0表示完全不透明

myAnimation\_Alpha=new AlphaAnimation(0.1f, 1.0f);

③　设置动画持续时间

//设置时间持续时间为 5000毫秒

myAnimation\_Alpha.setDuration(5000);

****7.2 ScaleAnimation****

①　ScaleAnimation类对象定义

private AlphaAnimation myAnimation\_Alpha;

②　ScaleAnimation类对象构造

ScaleAnimation(float fromX, float toX, float fromY, float toY,

int pivotXType, float pivotXValue, int pivotYType, float pivotYValue)

//第一个参数fromX为动画起始时 X坐标上的伸缩尺寸

//第二个参数toX为动画结束时 X坐标上的伸缩尺寸

//第三个参数fromY为动画起始时Y坐标上的伸缩尺寸

//第四个参数toY为动画结束时Y坐标上的伸缩尺寸

/\*说明:

以上四种属性值

0.0表示收缩到没有

1.0表示正常无伸缩

值小于1.0表示收缩

值大于1.0表示放大\*/

//第五个参数pivotXType为动画在X轴相对于物件位置类型

//第六个参数pivotXValue为动画相对于物件的X坐标的开始位置

//第七个参数pivotXType为动画在Y轴相对于物件位置类型

//第八个参数pivotYValue为动画相对于物件的Y坐标的开始位置

myAnimation\_Scale =new ScaleAnimation(0.0f, 1.4f, 0.0f, 1.4f,

Animation.RELATIVE\_TO\_SELF, 0.5f, Animation.RELATIVE\_TO\_SELF, 0.5f);

③　设置动画持续时间

//设置时间持续时间为 700毫秒

myAnimation\_Scale.setDuration(700);

****7.3 TranslateAnimation****

①　TranslateAnimation类对象定义

private AlphaAnimation myAnimation\_Alpha;

②　TranslateAnimation类对象构造

//第一个参数fromXDelta为动画起始时 X坐标上的移动位置

//第二个参数toXDelta为动画结束时 X坐标上的移动位置

//第三个参数fromYDelta为动画起始时Y坐标上的移动位置

//第四个参数toYDelta为动画结束时Y坐标上的移动位置

TranslateAnimation(float fromXDelta, float toXDelta,float fromYDelta, float toYDelta)

③　设置动画持续时间

//设置时间持续时间为 2000毫秒

myAnimation\_Translate.setDuration(2000);

****7.4 RotateAnimation****

①　RotateAnimation类对象定义

private AlphaAnimation myAnimation\_Alpha;

②　RotateAnimation类对象构造

RotateAnimation(float fromDegrees, float toDegrees,int pivotXType, float pivotXValue, int pivotYType, float pivotYValue)

//第一个参数fromDegrees为动画起始时的旋转角度

//第二个参数toDegrees为动画旋转到的角度

//第三个参数pivotXType为动画在X轴相对于物件位置类型

//第四个参数pivotXValue为动画相对于物件的X坐标的开始位置

//第五个参数pivotXType为动画在Y轴相对于物件位置类型

//第六个参数pivotYValue为动画相对于物件的Y坐标的开始位置

myAnimation\_Rotate=new RotateAnimation(0.0f, +350.0f,Animation.RELATIVE\_TO\_SELF,0.5f,Animation.RELATIVE\_TO\_SELF, 0.5f);

②　RotateAnimation类对象构造

//设置时间持续时间为 3000毫秒

myAnimation\_Rotate.setDuration(3000);

8.****如何使用Java代码中的动画效果****

 使用从View父类继承过来的方法startAnimation()来为View或是子类View等等添加一个动画效果

public void startAnimation (Animation animation)

****9.还是来个栗子吧****

****9.1 使用XML文件方式****

package com.yanis.base;

import android.app.Activity;import android.os.Bundle;import android.view.View;import android.view.View.OnClickListener;import android.view.animation.Animation;import android.view.animation.AnimationUtils;import android.widget.Button;import android.widget.ImageView;

public class AnimationActivity extends Activity implements OnClickListener {

private ImageView imgPic;

private Button btnAlpha, btnScale, btnTranslate, btnRotate;

private Animation myAnimation;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_animation);

intiView();

initData();

}

/\*\*

\* 初始化组件

\*/

private void intiView() {

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

btnAlpha = (Button) findViewById(R.id.btnAlpha);

btnScale = (Button) findViewById(R.id.btnScale);

btnTranslate = (Button) findViewById(R.id.btnTranslate);

btnRotate = (Button) findViewById(R.id.btnRotate);

}

/\*\*

\* 初始化数据

\*/

private void initData() {

btnAlpha.setOnClickListener(this);

btnScale.setOnClickListener(this);

btnTranslate.setOnClickListener(this);

btnRotate.setOnClickListener(this);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.btnAlpha:

/\*\*

\* 使用XML中的动画效果 第一个参数Context为程序的上下文 第二个参数id为动画XML文件的引用

\*/

myAnimation = AnimationUtils.loadAnimation(this, R.anim.alpha\_anim);

imgPic.startAnimation(myAnimation);

break;

case R.id.btnScale:

myAnimation = AnimationUtils.loadAnimation(this, R.anim.scale\_anim);

imgPic.startAnimation(myAnimation);

break;

case R.id.btnTranslate:

myAnimation = AnimationUtils.loadAnimation(this,

R.anim.translate\_anim);

imgPic.startAnimation(myAnimation);

break;

case R.id.btnRotate:

myAnimation = AnimationUtils

.loadAnimation(this, R.anim.rotate\_anim);

imgPic.startAnimation(myAnimation);

break;

}

}

}

****9.2 使用Java代码方式****

package com.example.ex4\_10;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.view.KeyEvent;

import android.view.View;

import android.view.animation.AlphaAnimation;

import android.view.animation.Animation;

import android.view.animation.Animation.AnimationListener;

import android.view.animation.RotateAnimation;

import android.view.animation.ScaleAnimation;

import android.view.animation.TranslateAnimation;

public class MyView extends View implements AnimationListener{

private Paint paint;

public MyView(Context context) {

super(context);

paint = new Paint();

paint.setColor(Color.WHITE);

paint.setTextSize(20);

setFocusable(true);

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

Bitmap bmp = BitmapFactory.decodeResource(this.getResources(), R.drawable.pic01);

//黑色背景 canvas.drawColor(Color.BLACK);

canvas.drawText("方向键 ↑ 渐变透明度动画效果", 80, this.getHeight()-80, paint);

canvas.drawText("方向键 ↓ 渐变尺寸伸缩动画效果", 80, this.getHeight()-60, paint);

canvas.drawText("方向键 → 画面转换位置移动动画效果", 80, this.getHeight()-40, paint);

canvas.drawText("方向键 ← 画面转移旋转动画效果", 80, this.getHeight()-20, paint);

//绘制位图，居中

canvas.drawBitmap(bmp, this.getWidth()/2-bmp.getWidth()/2, this.getHeight()/2-bmp.getHeight()/2, paint);

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if(keyCode==KeyEvent.KEYCODE\_DPAD\_UP)

{

//渐变透明效果

Animation m1 = new AlphaAnimation(0.1f, 1.0f);

//设置动画播放时间为3秒

m1.setDuration(3000);

//启动动画效果

this.startAnimation(m1);

}else if(keyCode==KeyEvent.KEYCODE\_DPAD\_DOWN)

{

//渐变尺寸缩放动画效果

Animation m2 = new ScaleAnimation(0.0f, 2.0f, 1.5f, 1.5f,Animation.RELATIVE\_TO\_PARENT,0.5f,Animation.RELATIVE\_TO\_PARENT,0.0f);

m2.setDuration(2000);

this.startAnimation(m2);

}else if(keyCode==KeyEvent.KEYCODE\_DPAD\_LEFT)

{

//移动动画效果

Animation m3 = new TranslateAnimation(0, 100, 0, 100);

m3.setDuration(2000);

this.startAnimation(m3);

}else if(keyCode==KeyEvent.KEYCODE\_DPAD\_RIGHT)

{

//旋转动画效果，这里是旋转360°

Animation m4 = new RotateAnimation(0.0f, 360.0f, Animation.RELATIVE\_TO\_SELF, 0.5f, Animation.RELATIVE\_TO\_SELF, 0.5f);

m4.setDuration(2000);

this.startAnimation(m4);

}

return super.onKeyDown(keyCode, event);

}

/\*\*

\* 动画开始时响应的函数

\*/

@Override

public void onAnimationStart(Animation animation) {

}

/\*\*

\* 动画结束时响应的函数

\*/

@Override

public void onAnimationEnd(Animation animation) {

}

/\*\*

\* 动画重播时响应的函数

\*/

@Override

public void onAnimationRepeat(Animation animation) {

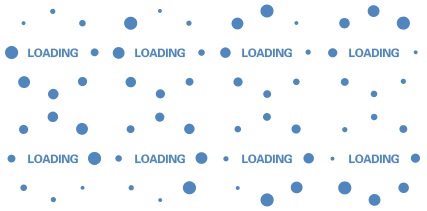
}

}

1. ****用Animation-list实现逐帧动画****

步骤如下：

①在res/drawable目录添加图片素材



②在drawable文件夹中添加动画Animation-list帧布局文件

<?xml version="1.0" encoding="utf-8"?><!--

根标签为animation-list，其中oneshot代表着是否只展示一遍，设置为false会不停的循环播放动画

根标签下，通过item标签对动画中的每一个图片进行声明

android:duration 表示展示所用的该图片的时间长度

--><animation-list xmlns:android="http://schemas.android.com/apk/res/android"

android:oneshot="false" >

<item

android:drawable="@drawable/cmmusic\_progress\_1"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_2"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_3"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_4"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_5"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_6"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_7"

android:duration="150">

</item>

<item

android:drawable="@drawable/cmmusic\_progress\_8"

android:duration="150">

</item></animation-list>

③主界面页面布局设置，太简单，不赘述了

④主界面代码如下：

package com.yanis.base;

import android.app.Activity;

import android.graphics.drawable.AnimationDrawable;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.ImageView;

public class AnimationActivity extends Activity implements OnClickListener {

private ImageView imgPic;

private Button btnStart, btnStop;

private AnimationDrawable animationDrawable;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_animation);

intiView();

initData();

}

/\*\*

\* 初始化组件

\*/

private void intiView() {

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

btnStart = (Button) findViewById(R.id.btnStart);

btnStop = (Button) findViewById(R.id.btnStop);

}

/\*\*

\* 初始化数据

\*/

private void initData() {

btnStart.setOnClickListener(this);

btnStop.setOnClickListener(this);

//Sets a drawable as the content of this ImageView.

imgPic.setImageResource(R.drawable.loading\_anim);

//给动画资源赋值

animationDrawable = (AnimationDrawable) imgPic.getDrawable();

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.btnStart:

animationDrawable.start();//开始

break;

case R.id.btnStop:

animationDrawable.stop(); //停止

break;

}

}

}