**[Android 属性动画（Property Animation） 完全解析 （下）](http://blog.csdn.net/lmj623565791/article/details/38092093)**

###### 1、如何使用xml文件来创建属性动画

大家肯定都清楚，View Animator 、Drawable Animator都可以在anim文件夹下创建动画，然后在程序中使用，甚至在Theme中设置为属性值。当然了，属性动画其实也可以在文件中声明：

首先在res下建立animator文件夹，然后建立res/animator/scalex.xml

**[html]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<objectAnimator** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:duration="1000"
4. android:propertyName="scaleX"
5. android:valueFrom="1.0"
6. android:valueTo="2.0"
7. android:valueType="floatType" **>**
8. **</objectAnimator>**

代码：

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **public** **void** scaleX(View view)
2. {
3. // 加载动画
4. Animator anim = AnimatorInflater.loadAnimator(**this**, R.animator.scalex);
5. anim.setTarget(mMv);
6. anim.start();
7. }

使用AnimatorInflater加载动画的资源文件，然后设置目标，就ok~~是不是很简单，这只是单纯横向的放大一倍~

如果我希望纵向与横向同时缩放呢？则可以怎么定义属性文件：

**[html]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<set** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:ordering="together" **>**
5. **<objectAnimator**
6. android:duration="1000"
7. android:propertyName="scaleX"
8. android:valueFrom="1"
9. android:valueTo="0.5" **>**
10. **</objectAnimator>**
11. **<objectAnimator**
12. android:duration="1000"
13. android:propertyName="scaleY"
14. android:valueFrom="1"
15. android:valueTo="0.5" **>**
16. **</objectAnimator>**
18. **</set>**

使用set标签，有一个orderring属性设置为together,【还有另一个值：sequentially（表示一个接一个执行）】。

上篇博客中忽略了一个效果，就是缩放、反转等都有中心点或者轴，默认中心缩放，和中间对称线为反转线，所以我决定这个横向，纵向缩小以左上角为中心点：

代码：

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. // 加载动画
2. Animator anim = AnimatorInflater.loadAnimator(**this**, R.animator.scale);
3. mMv.setPivotX(0);
4. mMv.setPivotY(0);
5. //显示的调用invalidate
6. mMv.invalidate();
7. anim.setTarget(mMv);
8. anim.start();

很简单，直接给View设置pivotX和pivotY，然后调用一下invalidate，就ok了。

###### 2、布局动画（Layout Animations）

主要使用LayoutTransition为布局的容器设置动画，当容器中的视图层次发生变化时存在过渡的动画效果。

基本代码为：

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. LayoutTransition transition = **new** LayoutTransition();
2. transition.setAnimator(LayoutTransition.CHANGE\_APPEARING,
3. transition.getAnimator(LayoutTransition.CHANGE\_APPEARING));
4. transition.setAnimator(LayoutTransition.APPEARING,
5. **null**);
6. transition.setAnimator(LayoutTransition.DISAPPEARING,
7. **null**);
8. transition.setAnimator(LayoutTransition.CHANGE\_DISAPPEARING,
9. **null**);
10. mGridLayout.setLayoutTransition(transition);

过渡的类型一共有四种：

LayoutTransition.APPEARING 当一个View在ViewGroup中出现时，对**此View**设置的动画

LayoutTransition.CHANGE\_APPEARING 当一个View在ViewGroup中出现时，对此View对其他View位置造成影响，对**其他View**设置的动画

LayoutTransition.DISAPPEARING  当一个View在ViewGroup中消失时，对**此View**设置的动画

LayoutTransition.CHANGE\_DISAPPEARING 当一个View在ViewGroup中消失时，对此View对其他View位置造成影响，对**其他View**设置的动画

LayoutTransition.CHANGE 不是由于View出现或消失造成对其他View位置造成影响，然后对其他View设置的动画。

注意动画到底设置在谁身上，此View还是其他View。

好了下面看一个综合的例子：

布局文件：

**[html]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools="http://schemas.android.com/tools"
3. android:id="@+id/id\_container"
4. android:layout\_width="match\_parent"
5. android:layout\_height="match\_parent"
6. android:orientation="vertical" **>**
8. **<Button**
9. android:layout\_width="wrap\_content"
10. android:layout\_height="wrap\_content"
11. android:onClick="addBtn"
12. android:text="addBtns" **/>**
14. **<CheckBox**
15. android:id="@+id/id\_appear"
16. android:layout\_width="wrap\_content"
17. android:layout\_height="wrap\_content"
18. android:checked="true"
19. android:text="APPEARING" **/>**
21. **<CheckBox**
22. android:id="@+id/id\_change\_appear"
23. android:layout\_width="wrap\_content"
24. android:layout\_height="wrap\_content"
25. android:checked="true"
26. android:text="CHANGE\_APPEARING" **/>**
28. **<CheckBox**
29. android:id="@+id/id\_disappear"
30. android:layout\_width="wrap\_content"
31. android:layout\_height="wrap\_content"
32. android:checked="true"
33. android:text="DISAPPEARING" **/>**
35. **<CheckBox**
36. android:id="@+id/id\_change\_disappear"
37. android:layout\_width="wrap\_content"
38. android:layout\_height="wrap\_content"
39. android:checked="true"
40. android:text="CHANGE\_DISAPPEARING " **/>**
42. **</LinearLayout>**

代码：

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **package** com.example.zhy\_property\_animation;
3. **import** android.animation.LayoutTransition;
4. **import** android.app.Activity;
5. **import** android.os.Bundle;
6. **import** android.view.View;
7. **import** android.view.View.OnClickListener;
8. **import** android.view.ViewGroup;
9. **import** android.widget.Button;
10. **import** android.widget.CheckBox;
11. **import** android.widget.CompoundButton;
12. **import** android.widget.CompoundButton.OnCheckedChangeListener;
13. **import** android.widget.GridLayout;
15. **public** **class** LayoutAnimaActivity **extends** Activity **implements**
16. OnCheckedChangeListener
17. {
18. **private** ViewGroup viewGroup;
19. **private** GridLayout mGridLayout;
20. **private** **int** mVal;
21. **private** LayoutTransition mTransition;
23. **private** CheckBox mAppear, mChangeAppear, mDisAppear, mChangeDisAppear;
25. @Override
26. **public** **void** onCreate(Bundle savedInstanceState)
27. {
28. **super**.onCreate(savedInstanceState);
29. setContentView(R.layout.layout\_animator);
30. viewGroup = (ViewGroup) findViewById(R.id.id\_container);
32. mAppear = (CheckBox) findViewById(R.id.id\_appear);
33. mChangeAppear = (CheckBox) findViewById(R.id.id\_change\_appear);
34. mDisAppear = (CheckBox) findViewById(R.id.id\_disappear);
35. mChangeDisAppear = (CheckBox) findViewById(R.id.id\_change\_disappear);
37. mAppear.setOnCheckedChangeListener(**this**);
38. mChangeAppear.setOnCheckedChangeListener(**this**);
39. mDisAppear.setOnCheckedChangeListener(**this**);
40. mChangeDisAppear.setOnCheckedChangeListener(**this**);
42. // 创建一个GridLayout
43. mGridLayout = **new** GridLayout(**this**);
44. // 设置每列5个按钮
45. mGridLayout.setColumnCount(5);
46. // 添加到布局中
47. viewGroup.addView(mGridLayout);
48. //默认动画全部开启
49. mTransition = **new** LayoutTransition();
50. mGridLayout.setLayoutTransition(mTransition);
52. }
54. /\*\*
55. \* 添加按钮
56. \*
57. \* @param view
58. \*/
59. **public** **void** addBtn(View view)
60. {
61. **final** Button button = **new** Button(**this**);
62. button.setText((++mVal) + "");
63. mGridLayout.addView(button, Math.min(1, mGridLayout.getChildCount()));
64. button.setOnClickListener(**new** OnClickListener()
65. {
67. @Override
68. **public** **void** onClick(View v)
69. {
70. mGridLayout.removeView(button);
71. }
72. });
73. }
75. @Override
76. **public** **void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked)
77. {
78. mTransition = **new** LayoutTransition();
79. mTransition.setAnimator(
80. LayoutTransition.APPEARING,
81. (mAppear.isChecked() ? mTransition
82. .getAnimator(LayoutTransition.APPEARING) : **null**));
83. mTransition
84. .setAnimator(
85. LayoutTransition.CHANGE\_APPEARING,
86. (mChangeAppear.isChecked() ? mTransition
87. .getAnimator(LayoutTransition.CHANGE\_APPEARING)
88. : **null**));
89. mTransition.setAnimator(
90. LayoutTransition.DISAPPEARING,
91. (mDisAppear.isChecked() ? mTransition
92. .getAnimator(LayoutTransition.DISAPPEARING) : **null**));
93. mTransition.setAnimator(
94. LayoutTransition.CHANGE\_DISAPPEARING,
95. (mChangeDisAppear.isChecked() ? mTransition
96. .getAnimator(LayoutTransition.CHANGE\_DISAPPEARING)
97. : **null**));
98. mGridLayout.setLayoutTransition(mTransition);
99. }
100. }

动画有点长，耐心点看，一定要注意，是对当前View还是其他Views设置的动画。

当然了动画支持自定义，还支持设置时间，比如我们修改下，添加的动画为：

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. mTransition.setAnimator(LayoutTransition.APPEARING, (mAppear
2. .isChecked() ? ObjectAnimator.ofFloat(**this**, "scaleX", 0, 1)
3. : **null**));

原本的淡入，变成了宽度从中间放大的效果~~是不是还不错~~

###### 3、View的anim方法

在SDK11的时候，给View添加了animate方法，更加方便的实现动画效果。

布局文件：

**[html]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **<RelativeLayout** xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools="http://schemas.android.com/tools"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. **>**
7. **<ImageView**
8. android:id="@+id/id\_ball"
9. android:layout\_width="wrap\_content"
10. android:layout\_height="wrap\_content"
11. android:src="@drawable/bol\_blue" **/>**
13. **<LinearLayout**
14. android:layout\_width="fill\_parent"
15. android:layout\_height="wrap\_content"
16. android:layout\_alignParentBottom="true"
17. android:orientation="horizontal" **>**
19. **<Button**
20. android:layout\_width="wrap\_content"
21. android:layout\_height="wrap\_content"
22. android:onClick="viewAnim"
23. android:text="View Anim" **/>**
25. **<Button**
26. android:layout\_width="wrap\_content"
27. android:layout\_height="wrap\_content"
28. android:onClick="propertyValuesHolder"
29. android:text="PropertyValuesHolder " **/>**

32. **</LinearLayout>**
34. **</RelativeLayout>**

代码：

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. **package** com.example.zhy\_property\_animation;
3. **import** android.animation.ObjectAnimator;
4. **import** android.animation.PropertyValuesHolder;
5. **import** android.app.Activity;
6. **import** android.os.Bundle;
7. **import** android.util.DisplayMetrics;
8. **import** android.util.Log;
9. **import** android.view.View;
10. **import** android.widget.ImageView;
12. **public** **class** ViewAnimateActivity **extends** Activity
13. {
14. **protected** **static** **final** String TAG = "ViewAnimateActivity";
16. **private** ImageView mBlueBall;
17. **private** **float** mScreenHeight;
19. @Override
20. **protected** **void** onCreate(Bundle savedInstanceState)
21. {
22. **super**.onCreate(savedInstanceState);
23. setContentView(R.layout.view\_animator);
25. DisplayMetrics outMetrics = **new** DisplayMetrics();
26. getWindowManager().getDefaultDisplay().getMetrics(outMetrics);
27. mScreenHeight = outMetrics.heightPixels;
28. mBlueBall = (ImageView) findViewById(R.id.id\_ball);
30. }
32. **public** **void** viewAnim(View view)
33. {
34. // need API12
35. mBlueBall.animate()//
36. .alpha(0)//
37. .y(mScreenHeight / 2).setDuration(1000)
38. // need API 12
39. .withStartAction(**new** Runnable()
40. {
41. @Override
42. **public** **void** run()
43. {
44. Log.e(TAG, "START");
45. }
46. // need API 16
47. }).withEndAction(**new** Runnable()
48. {
50. @Override
51. **public** **void** run()
52. {
53. Log.e(TAG, "END");
54. runOnUiThread(**new** Runnable()
55. {
56. @Override
57. **public** **void** run()
58. {
59. mBlueBall.setY(0);
60. mBlueBall.setAlpha(1.0f);
61. }
62. });
63. }
64. }).start();
65. }                                                                                                                                                  }

简单的使用mBlueBall.animate().alpha(0).y(mScreenHeight / 2).setDuration(1000).start()就能实现动画~~不过需要SDK11，此后在SDK12，SDK16又分别添加了withStartAction和withEndAction用于在动画前，和动画后执行一些操作。当然也可以.setListener(listener)等操作。

使用ObjectAnimator实现上面的变化，我们可以使用：PropertyValueHolder

**[java]** [view plain](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "view plain) [copy](http://blog.csdn.net/lmj623565791/article/details/38092093" \o "copy)

1. PropertyValuesHolder pvhX = PropertyValuesHolder.ofFloat("alpha", 1f,
2. 0f, 1f);
3. PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("y", 0,
4. mScreenHeight / 2, 0);
5. ObjectAnimator.ofPropertyValuesHolder(mBlueBall, pvhX, pvhY).setDuration(1000).start();

效果与上面一样。