**动画类型**Android的animation由四种类型组成   
  
XML中

|  |  |
| --- | --- |
| alpha | 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 |

**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 |

**Android动画解析**

**alpha xml 淡出效果**

1. <?xml version="1.0" encoding="utf-8"?>
2. <set xmlns:android="http://schemas.android.com/apk/res/android">
3. <alpha
4. android:fromAlpha="1.0"
5. android:toAlpha="0.0"
6. android:duration="500" />
7. </set>
8. <!--
9. fromAlpha:开始时透明度
10. toAlpha： 结束时透明度
11. duration：动画持续时间 -->

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

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

<alpha

android:fromAlpha="1.0"

android:toAlpha="0.0"

android:duration="500" />

</set>

<!--

fromAlpha:开始时透明度

toAlpha： 结束时透明度

duration：动画持续时间 -->

**alpha xml 淡入效果**

1. <?xml version="1.0" encoding="utf-8"?>
2. <set xmlns:android="http://schemas.android.com/apk/res/android">
3. <alpha
4. android:fromAlpha="0.0"
5. android:toAlpha="1.0"
6. android:duration="500" />
7. </set>
8. <!--
9. fromAlpha:开始时透明度
10. toAlpha： 结束时透明度
11. duration：动画持续时间 -->

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

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

<alpha

android:fromAlpha="0.0"

android:toAlpha="1.0"

android:duration="500" />

</set>

<!--

fromAlpha:开始时透明度

toAlpha： 结束时透明度

duration：动画持续时间 -->

**rotate.xml 旋转效果：**

1. <?xmlversion="1.0"encoding="utf-8"?>
2. <setxmlns:android="http://schemas.android.com/apk/res/android">
3. <rotate
4. android:interpolator="@android:anim/accelerate\_decelerate\_interpolator"
5. android:fromDegrees="300"
6. android:toDegrees="-360"
7. android:pivotX="10%"
8. android:pivotY="100%"
9. android:duration="10000"/>
10. </set>
11. <!--
12. fromDegrees 动画开始时的角度
13. toDegrees 动画结束时物件的旋转角度,正代表顺时针
14. pivotX 属性为动画相对于物件的X坐标的开始位置
15. pivotY 属性为动画相对于物件的Y坐标的开始位置 -->

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

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

<rotate

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

android:fromDegrees="300"

android:toDegrees="-360"

android:pivotX="10%"

android:pivotY="100%"

android:duration="10000" />

</set>

<!--

fromDegrees 动画开始时的角度

toDegrees 动画结束时物件的旋转角度,正代表顺时针

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 -->

**scale.xml 缩放效果：**

1. <?xml version="1.0" encoding="utf-8"?>
2. <set xmlns:android="http://schemas.android.com/apk/res/android">
3. <scale
4. android:interpolator= "@android:anim/decelerate\_interpolator"
5. android:fromXScale="0.0"
6. android:toXScale="1.5"
7. android:fromYScale="0.0"
8. android:toYScale="1.5"
9. android:pivotX="50%"
10. android:pivotY="50%"
11. android:startOffset="0"
12. android:duration="10000"
13. android:repeatCount="1"
14. android:repeatMode="reverse" />
15. </set>
16. <!--
17. fromXDelta,fromYDelta 起始时X，Y座标,屏幕右下角的座标是X:320,Y:480
18. toXDelta， toYDelta 动画结束时X,Y的座标 --> <!--
19. interpolator 指定动画插入器
20. 常见的有加速减速插入器 accelerate\_decelerate\_interpolator
21. 加速插入器 accelerate\_interpolator，
22. 减速插入器 decelerate\_interpolator。
23. fromXScale,fromYScale， 动画开始前X,Y的缩放，0.0为不显示， 1.0为正常大小
24. toXScale，toYScale， 动画最终缩放的倍数， 1.0为正常大小，大于1.0放大
25. pivotX， pivotY 动画起始位置，相对于屏幕的百分比,两个都为50%表示动画从屏幕中间开始
26. startOffset， 动画多次执行的间隔时间，如果只执行一次，执行前会暂停这段时间，
27. 单位毫秒 duration，一次动画效果消耗的时间，单位毫秒，
28. 值越小动画速度越快 repeatCount，动画重复的计数，动画将会执行该值+1次
29. repeatMode，动画重复的模式，reverse为反向，当第偶次执行时，动画方向会相反。
30. restart为重新执行，方向不变 -->

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

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

<scale

android:interpolator= "@android:anim/decelerate\_interpolator"

android:fromXScale="0.0"

android:toXScale="1.5"

android:fromYScale="0.0"

android:toYScale="1.5"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="0"

android:duration="10000"

android:repeatCount="1"

android:repeatMode="reverse" />

</set>

<!--

fromXDelta,fromYDelta 起始时X，Y座标,屏幕右下角的座标是X:320,Y:480

toXDelta， toYDelta 动画结束时X,Y的座标 --> <!--

interpolator 指定动画插入器

常见的有加速减速插入器 accelerate\_decelerate\_interpolator

加速插入器 accelerate\_interpolator，

减速插入器 decelerate\_interpolator。

fromXScale,fromYScale， 动画开始前X,Y的缩放，0.0为不显示， 1.0为正常大小

toXScale，toYScale， 动画最终缩放的倍数， 1.0为正常大小，大于1.0放大

pivotX， pivotY 动画起始位置，相对于屏幕的百分比,两个都为50%表示动画从屏幕中间开始

startOffset， 动画多次执行的间隔时间，如果只执行一次，执行前会暂停这段时间，

单位毫秒 duration，一次动画效果消耗的时间，单位毫秒，

值越小动画速度越快 repeatCount，动画重复的计数，动画将会执行该值+1次

repeatMode，动画重复的模式，reverse为反向，当第偶次执行时，动画方向会相反。

restart为重新执行，方向不变 -->

**translate.xml 移动效果：**

1. <?xmlversion="1.0"encoding="utf-8"?>
2. <setxmlns:android="http://schemas.android.com/apk/res/android">
3. <translate
4. android:fromXDelta="320"
5. android:toXDelta="0"
6. android:fromYDelta="480"
7. android:toYDelta="0"
8. android:duration="10000"/>
9. </set>
10. <!--
11. fromXDelta,fromYDelta 起始时X，Y座标,屏幕右下角的座标是X:320,Y:480
12. toXDelta， toYDelta 动画结束时X,Y的座标 -->

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

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

<translate

android:fromXDelta="320"

android:toXDelta="0"

android:fromYDelta="480"

android:toYDelta="0"

android:duration="10000" />

</set>

<!--

fromXDelta,fromYDelta 起始时X，Y座标,屏幕右下角的座标是X:320,Y:480

toXDelta， toYDelta 动画结束时X,Y的座标 -->

GridView

1.控件属性介绍

**android:listSelector="@drawable/bg"**

//该属性很重要，如果不设置的话，GridView控件会自带自己的选中样式(黄色边框)，如图所示：



做项目一般是不需要这个的。所以解决办法是将GridView的android:listSelector属性设置为和Activity背景相同即可。  
android:verticalSpacing="10dp" //行宽  
android:horizontalSpacing="10dip" //列宽  
android:numColumns="3" //列数

android:stretchMode="columnWidth" //列可扩展  
以上列数为3，若想在高分辨率中列数设置为4怎么办呢。

首先，需要在属性中设置列为可扩展android:stretchMode="columnWidth"。然后在代码中做个判断：

if(Metrics.heightPixels == 1280 && Metrics.widthPixels == 800){  
gridview.setNumColumns(4);  
}这样GridView的列数是可以随分辨率不同而做更多的设置。

2.例子一：**设置个边框，并且右上角设置个CheckBox来美化选中状态**(左图→右图)。

→→→

这个Item的xml设计中，我是用两个ImageView控件来实现的.

第一个ImageView 要设置一个属性：android:layout\_margin="4dip"。上下左右留出4dip的具体来显示边框的效果。并且在相应的Adapter中做个判断，选中的话背景图片颜色为#87CEFA，未选中时为3FFFFFF：

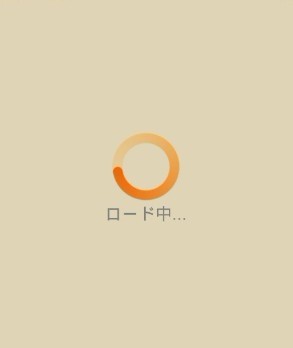
if(bSelect){ //bSelect表示是否选中  
arg1.setBackgroundColor(Color.parseColor("#87CEFA"));  
}else{  
arg1.setBackgroundColor(Color.parseColor("#FFFFFF"));

右上角的看起来像复选框，其实我也是用ImageView控件来实现的。界面设计很重要，距离要对好。我设置的该控件离上边距和右边距分别为2dip的距离。同样的只需要修改该ImageView所绑定的图片就可以。

if(bSelect){  
viewclass.checkboxImage.setImageBitmap(bmpwater\_sel); //bmpwater\_sel表示左图右上角的图片  
}else{  
viewclass.checkboxImage.setImageBitmap(bmpwater\_unsel); //bmpwater\_unsel表示右图右上角的图片  
}

很简单吧。你们可以试试。

3.例子二：**打开一个GridView控件的Activity时，如果打开的图片过多，如何让图片异步加载**。如图所示：

 → 

首先在界面定义中要定义一个全屏显示的加载圈效果，叫做mLoading。程序刚启动时显示图片加载结束后调用函数mLoading.setVisibility(View.GONE);让它不显示。我是用一个函数来调用它，这样更好的体现了封装性，如下：

private void showLoading(boolean bShow){  
if(mLoading == null) return;  
mLoading.setVisibility(bShow ? View.VISIBLE : View.GONE);  
}

xml中该加载框的设计如下：

[View Code](http://www.cnblogs.com/carmanloneliness/archive/2012/05/29/android_%E8%B6%A3%E5%91%B3%E7%9A%84gridView%E6%8E%A7%E4%BB%B6.html)

①为了不卡主线程，实现这一的效果要多开一个线程。在该线程中调用函数InitList(),在该函数中将图片加载到ArrayList中去(花时间最多的就是对多图片的处理，这个放在子线程中可以不卡主线程，如果直接在主线程中做，会让画面出现2-3秒的黑屏停顿)。

像这样，在onCreate方法中开一个子线程：

new Thread(new Runnable() {  
@Override  
public void run() {  
InitList();  
}  
}).start();

②方法InitList()中要做两件事情。一是加载图片，二是通知Handler图片加载完成(因为子线程不可以操作UI控件，只能通过Handler的方式)。

private void InitList()  
{

XXXX; //加载图片代码  
mMainHandler.sendMessage(MainHandler.MSG\_FINISH); //发送消息  
}

③当收到InitList发送的消息时就让上面所提的mLoading不显示。同时GridView绑定adapter的代码也在这个里面写。因为图片加载完成后再绑定adapter,这样图片才能全部显示，否则只能显示一部分。

case MSG\_FINISH:  
showLoading(false); //让加载框不再显示  
gridview.setAdapter(ia); //绑定数据  
break;  
default:  
break;