## android转场动画windowAnimation和ActivityAnimation的区别

显而易见，window与Activity本身从名字上就知道不同，但对于项目开发中 windowAnimation和ActivityAnimation的区别必须心领神会。

区别主要如下

1.windowAnimation包括**windowEnterAnimation**和**windowExitAnimation**；ActivityAnimation包含**android:activityOpenEnterAnimation**，**android:activityOpenExitAnimation**，**android:activityCloseEnterAnimation**和**android:activityCloseExitAnimation**

2.在项目中WindowAnimation的控制权大于Activity的控制权，即在Activity转场过程中，如果同时设置了WindowAnimation和ActivityAnimation,那么**可能（因为这种情况非常多）**只会执行WindowAnimation

3.对于WindowAnimation的定义很简单，在style.xml文件中只需要继承Animation Style即可

   <style name="Animation" parent="@android:style/Animation">

          <!--窗体进入动画-->

        <item name="android:windowEnterAnimation">@anim/slide\_left\_enter</item>

        <!--窗体退出动画-->

        <item name="android:windowExitAnimation">@anim/slide\_right\_exit</item>

    </style>

对于Activity,需要继承Animation Activity Style

<style name="FeelyouWindowAnimTheme" parent="@android:style/Animation.Activity">

    <item name="android:activityOpenEnterAnimation">@android:anim/slide\_in\_left</item>

    <item name="android:activityOpenExitAnimation">@android:anim/slide\_out\_right</item>

    <item name="android:activityCloseEnterAnimation">@anim/push\_right\_in</item>

    <item name="android:activityCloseExitAnimation">@anim/push\_left\_out</item>

</style>

当我们从 A1 启动 A2 时，A1 从屏幕上消失，这个动画叫做 android:activityOpenExitAnimation

当我们从 A1 启动 A2 时，A2 出现在屏幕上，这个动画叫做 android:activityOpenEnterAnimation

当我们从 A2 退出回到 A1 时，A2 从屏幕上消失，这个叫做 android:activityCloseExitAnimation

当我们从 A2 退出回到 A1 时，A1 出现在屏幕上，这个叫做 android:activityCloseEnterAnimation

从上述2中动画的定义上来看，显然ActivityAnimation更为复杂，但这种复杂带来的转场效果非常好，可以同时控制2个Activity的动画，而不像WindowAnimation只能控制下一个Activity的窗体动画。

5.在开发中，窗体动画也可以使用 Activity.overridePendingTransition来设置,也可以定义在主题中，但Activity只能使用在主题中

<application

android:name="test.view.weitop.BaseApplication"

android:allowBackup="false"

android:hardwareAccelerated="false"

android:icon="@drawable/app\_logo"

android:label="@string/app\_name"

android:largeHeap="true"

android:theme="@style/Theme.App" >

</application>

6.对于比较追求界面美感的动画，使用ActivityAnimation要好得多，因此对于一般开发使用windowAnimation即可，但对于追求用户体验，那么直接使用Activity Animation 吧

这里分享了一篇关于Activity动画的研究博客：<http://www.oschina.net/question/97118_34343>

-------------------------------------------------2014-12-27新增内容------------

Android布局动画，所谓布局动画并不是指专门作用在ViewGroup上的动画，而是作用到不居中的Child的有序动画

1.LayoutAnimationController --->第三方控件 WheelView的核心部分

 ScaleAnimation sa =new ScaleAnimation(0.5f, 1f, 0.5f, 1f, Animation.RELATIVE\_TO\_SELF, 0.5f, Animation.RELATIVE\_TO\_SELF, 0.5f);

            sa.setDuration(2000);

            LayoutAnimationController lac = new LayoutAnimationController(sa, 0.5f);

            lac.setOrder(LayoutAnimationController.ORDER\_REVERSE);

            rootView.setLayoutAnimation(lac);

2.android:animateLayoutChanges="true"

表示布局中childView的的大小改变，删除，增加时的 fade in 或者 fade out的动画

3.android:layoutAnimation="@anim/listview\_anim"

android:layoutAnimation 和LayoutAnimationController作用类似，不同点在于，前者定义在xml中，后者使用代码编写

# [window的设置属性](https://www.cnblogs.com/zhangmiao14/p/7491756.html)

## 1.windowAnimationStyle

　　用于设置一组window动画的引用style资源，window的动画属性由R.styleable.WindowAnimation定义。  
　　Winow动画类属性包含以下属性：  
　　　　android:activityCloseEnterAnimation 　　 主要是将R.attr.activityCloseEnterAnimation的属性添加在WindowAnimation数组中。  
　　　　android:activityCloseExitAnimation 　　    主要是将R.attr.activityCloseExitAnimation的属性添加在WindowAnimation数组中。  
　　　　android:activityOpenEnterAnimation 　　 主要是将R.attr.activityOpenEnterAnimation的属性添加在WindowAnimation数组中。  
　　　　android:activityOpenExitAnimation 　　    主要是将R.attr.activityOpenExitAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskCloseEnterAnimation 　　     主要是将R.attr.taskCloseEnterAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskCloseExitAnimation 　　        主要是将R.attr.taskCloseExitAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskOpenEnterAnimation 　　     主要是将R.attr.taskOpenEnterAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskOpenExitAnimation 　　        主要是将R.attr.taskOpenExitAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskToBackExterAnimation 　　   主要是将R.attr.taskToBackEnterAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskToBackExitAnimation 　　     主要是将R.attr.taskToBackExitAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskToFrontEnterAnimation 　　 主要是将R.attr.taskToFrontEnterAnimation的属性添加在WindowAnimation数组中。  
　　　　android:taskToFrontExitAnimation 　　    主要是将R.attr.taskToFrontExitAnimation的属性添加在WindowAnimation数组中。  
　　　　android:windowEnterAnimation 　　         当window被添加时动画被使用。  
　　　　android:windowExitAnimation 　　            当window被移除时动画被使用。  
　　　　android:windowHideAnimation 　　          当window从VISIBLE到INVISIBLE使动画被使用。  
　　　　android:windowShowAnimation　　         当window从INVISIBLE到VISIBLE时动画被使用。

## 2.windowBackground

　　用于全部的window背景。当设置window时有一些特别的情况：  
　　　　当信息是用来推算window的surface的像素值时，如果drwable是非半透明像素，则window是半透明的。  
　　　　如果你想自己画整个背景，需要设置绘制的颜色和关闭的背景颜色相匹配，记得在代码中设置window的background为null，不然不会绘制。

## 3.windowContentOverlay

　　重叠在window内容区域的前景，经常用于在title下放置阴影。

## 4.windowDisablePreview

　　是否禁止window预览动画，默认值为false，如果设置为true，在你的实际实例没有显示给用户之前，系统不能使用window的theme显示预览。

## 5.windowEnterAnimation

　　当window被添加时动画被使用。

## 6.windowExitAnimation

　　当window被移除时动画被使用。

## 7.windowFrame

　　在window中使用框架。

## 8.windowFullscreen

　　标志window是否填充整个屏幕。

## 9.windowHideAnimation

　　当window从VISIBLE到INVISIBLE使动画被使用。

## 10. windowIsFloating

　　标志window是否为浮动窗口。

## 11.windowIsTranslucent

　　表示window是否为半透明窗口。

## 12.windowNoTitle

　　标志window是否没有title。

## 13.windowShowAnimation

　　当window从INVISIBLE到VISIBLE时动画被使用。

## 14.windowSoftInputMode

　　定义当window在显示时默认的输入模式。  
　　必须有以下一个或多个常数值：  
　　Constant                 Vlaue     Description  
　　stateUnspecified 　 0 　　   没有指定，使用系统认为最好的，这个是默认值。  
　　stateUnchanged     1 　　    保留window的输入模式，不管什么状态。  
　　stateHidden            2 　　    不显示输入区域。  
　　stateVisible             3 　　    使输入在正常情况（当用户导航到窗口）下可见。  
　　stateAlwaysVisible  4 　　    当window有输入焦点时总是使输入区域可见。  
　　adjustUnspecified   5 　　    window的resize/pan没有特别调整，系统将自动在resize和pan模式中选择，选择依据依赖于window内容是否含有具有滚动内容的视图。如果有，则window调整大小，在假设调整后的大小能够减少给输入UI让出的地方。  
　　adjustResize 　　   6　　    始终调整窗口的大小，window的内容区域缩小是的输入区域增大。  
　　adjustPan 　　       7 　　   不能调整窗口的大小，以便为输入区腾出空间；相反的，将window的内容作为焦点移动到window的内部，这样是用户可以看到输入的内容。这相比较不是很理想，因为用户也许需要关闭输入区域来与window的部分进行交互。

## 15.windowTitleBackgroundStyle

　　window的title区域使用的style资源。

## 16.windowTitleSize

　　window的title bar的高度的style资源。  
　　必须是dimension值，是一个有单位的浮点数如14.5sp，可用的单位是：px,dp,sp,in,mm。

## 17.windowTitleStyle

　　window的title的text的style资源。

WindowManager动画有要求,要求一定要是系统style里的动画组,其他设置是无效的!   
你可以设置R.style.Animation\_Toast试试!这是Toast效果动画!

mParentView = new FrameLayout(mContext);

mWM.addView(mParentView, params);

mParentView.addView(mRootView);

Animation animationUp = AnimationUtils.loadAnimation(mContext, R.anim.grow\_from\_bottom);

mRootView.startAnimation(animationUp);

---------------------

'm not sure about exact requirements for Your task, but there's two ways to provide animation to window:

1. Use WindowManager.LayoutParams.windowAnimations like the following:

params.windowAnimations = android.R.style.Animation\_Translucent;

1. Add additonal 'container' view, because WindowManager is not a real ViewGroup and so normal animation for adding views is not working with it. [This question has been asked already](https://stackoverflow.com/questions/17537672/animate-system-alert-type-view), however it lacks the code. I would apply it the following way:
2. public class BannerLayout extends View {
3. private final Context mContext;
4. private final ViewGroup mPopupLayout;
5. private final ViewGroup mParentView;
6. public BannerLayout(Activity activity, final Context context) {
7. super(context);
8. mContext = context;
9. final WindowManager.LayoutParams params = new WindowManager.LayoutParams(
10. WindowManager.LayoutParams.TYPE\_SYSTEM\_OVERLAY,
11. WindowManager.LayoutParams.FLAG\_LAYOUT\_IN\_SCREEN |
12. WindowManager.LayoutParams.FLAG\_WATCH\_OUTSIDE\_TOUCH,
13. PixelFormat.TRANSLUCENT);
14. final WindowManager mWinManager = (WindowManager) context.getSystemService(Context.WINDOW\_SERVICE);
15. LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);
16. mPopupLayout = (RelativeLayout) inflater.inflate(R.layout.popup\_activity, null);
17. mPopupLayout.setVisibility(GONE);
18. params.width = ActionBar.LayoutParams.WRAP\_CONTENT;
19. params.height = ActionBar.LayoutParams.WRAP\_CONTENT;
20. // Default variant
21. // params.windowAnimations = android.R.style.Animation\_Translucent;
22. mParentView = new FrameLayout(mContext);
23. mWinManager.addView(mParentView, params);
24. mParentView.addView(mPopupLayout);
25. mPopupLayout.setVisibility(GONE);
26. }
27. /\*\*
28. \* Shows view
29. \*/
30. public void show(){
31. final Animation in = AnimationUtils.loadAnimation(this.mContext, android.R.anim.fade\_in);
32. in.setDuration(2000);
33. mPopupLayout.setVisibility(VISIBLE);
34. mPopupLayout.startAnimation(in);
35. }
36. /\*\*
37. \* Hides view
38. \*/
39. public void hide() {
40. mPopupLayout.setVisibility(GONE);
41. }

}

[share](https://stackoverflow.com/a/17758858)[improve this answer](https://stackoverflow.com/posts/17758858/edit)

**10.6k**64760

* sandrstar... worked perfectly! however... I wonder if it's possible using animation translation with these components. I need to make an effect up and down the screen with this component... – [LeandroPortnoy](https://stackoverflow.com/users/2311822/leandroportnoy) [Jul 22 '13 at 12:09](https://stackoverflow.com/questions/17745282/windowmanager-with-animation-is-it-possible#comment25945985_17758858)
* private void show(){ //Animation fadeIn = (Animation) AnimationUtils.loadAnimation(getContext(), android.R.anim.fade\_in); //this.startAnimation(fadeIn); //this.bannerRelativeLayout.setVisibility(VISIBLE); this.setActive(true); mPopupLayout.setVisibility(VISIBLE); final Animation in = new TranslateAnimation(0, 0, -1000, 0 ); in.setDuration(700); AnimationSet animation = new AnimationSet(false); animation.addAnimation(in); mPopupLayout.startAnimation(animation); } – [LeandroPortnoy](https://stackoverflow.com/users/2311822/leandroportnoy) [Jul 22 '13 at 12:28](https://stackoverflow.com/questions/17745282/windowmanager-with-animation-is-it-possible#comment25946671_17758858)
* Sorry for delay. I've tried it and seems it works fine. May be You need to use params.width = ViewGroup.LayoutParams.MATCH\_PARENT; params.height = ViewGroup.LayoutParams.MATCH\_PARENT; for FrameLayout. – [sandrstar](https://stackoverflow.com/users/657487/sandrstar) [Jul 23 '13 at 2:45](https://stackoverflow.com/questions/17745282/windowmanager-with-animation-is-it-possible#comment25970895_17758858)
* usefully.thanks – [Albert.Qing](https://stackoverflow.com/users/770627/albert-qing) [Jan 29 '14 at 7:12](https://stackoverflow.com/questions/17745282/windowmanager-with-animation-is-it-possible#comment32321779_17758858)
* @sandrstar Could you take a look at this question about WindowManager? Thank you [stackoverflow.com/questions/27937250/…](http://stackoverflow.com/questions/27937250/viewpager-on-a-view-added-by-windowmanager-getting-java-lang-illegalargumentexc/27937559) – [Hoa Vu](https://stackoverflow.com/users/2552811/hoa-vu) [Jan 15 '15 at 2:09](https://stackoverflow.com/questions/17745282/windowmanager-with-animation-is-it-possible#comment44306331_17758858)

随笔 - 104  文章 - 0  评论 - 149

# [Android 编程下设置 Activity 切换动画](https://www.cnblogs.com/sunzn/p/3854009.html)

### 为 Activity 设置切换动画

我们知道，我们可以在 AndroidManifest.xml 文件中，通过 android:theme 属性设置 Activity 的主题。主题中定义了关于 Activity 外观的很多特性。同时，主题中还可以定义 Activity 的切换动画。这是应用 Activity 切换动画的一种方法。下面讲解一下如何通过主题来设置 Activity 间的切换动画。

### 定义包含动画的 Activity 主题

res/values/styles.xml，很简单，就是使用 windowAnimationStyle 这个属性，指定切换动画的style即可。

<style name="AnimActivityTheme">

<item name="android:windowAnimationStyle">@style/FeelyouWindowAnimTheme</item>

</style>

### 定义切换动画 style

res/values/styles.xml

<style name="FeelyouWindowAnimTheme" parent="@android:style/Animation.Activity">

<item name="android:activityOpenEnterAnimation">@anim/in\_from\_left</item>

<item name="android:activityOpenExitAnimation">@anim/out\_from\_right</item>

<item name="android:activityCloseEnterAnimation">@anim/in\_from\_right</item>

<item name="android:activityCloseExitAnimation">@anim/out\_from\_left</item>

</style>

注意需要继承自 @android:style/Animation.Activity。具体这4个属性什么意思呢？假设我们有 2 个 Activity，分别是 A1 和 A2：

* 当我们从 A1 启动 A2 时，A1 从屏幕上消失，这个动画叫做 android:activityOpenExitAnimation
* 当我们从 A1 启动 A2 时，A2 出现在屏幕上，这个动画叫做 android:activityOpenEnterAnimation
* 当我们从 A2 退出回到 A1 时，A2 从屏幕上消失，这个叫做 android:activityCloseExitAnimation
* 当我们从 A2 退出回到 A1 时，A1 出现在屏幕上，这个叫做 android:activityCloseEnterAnimation

结合上面我讲的情况，在脑海中想象一下，再结合属性的名字，就很容易理解了！

### 定义具体动画文件

这里的动画可以是透明度、位移、缩放之类的任何动画，我这里以位移来举例。当我们 A1 启动 A2 时，我希望 A1 从右边平移退出屏幕（out\_from\_right)，A2 从左边平移进入屏幕（in\_from\_left)。当我们从 A2 退出回到 A1 时，我希望 A2 从左边平移退出屏幕（out\_from\_left），A1 从右边平移进入屏幕（in\_from\_right）

下面我贴一下4个具体的动画的 xml 文件：

anim/out\_from\_right.xml

[复制代码](javascript:void(0);)

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

<!--

author:Remex Huang

website:feelyou.info

-->

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

android:duration="500"

android:fromXDelta="0"

android:fromYDelta="0"

android:toXDelta="100%p"

android:toYDelta="0" >

</translate>

[复制代码](javascript:void(0);)

anim/in\_from\_left.xml

[复制代码](javascript:void(0);)

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

<!--

author:Remex Huang

website:feelyou.info

-->

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

android:duration="500"

android:fromXDelta="-100%p"

android:fromYDelta="0"

android:toXDelta="0"

android:toYDelta="0" >

</translate>

[复制代码](javascript:void(0);)

anim/out\_from\_left.xml

[复制代码](javascript:void(0);)

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

<!--

author:Remex Huang

website:feelyou.info

-->

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

android:duration="500"

android:fromXDelta="0"

android:fromYDelta="0"

android:toXDelta="-100%p"

android:toYDelta="0" >

</translate>

[复制代码](javascript:void(0);)

anim/in\_from\_right.xml

[复制代码](javascript:void(0);)

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

<!--

author:Remex Huang

website:feelyou.info

-->

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

android:duration="500"

android:fromXDelta="100%p"

android:fromYDelta="0"

android:toXDelta="0"

android:toYDelta="0" >

</translate>

[复制代码](javascript:void(0);)

### 应用到对应 Activity

AndroidMenifest.xml

[复制代码](javascript:void(0);)

<activity

android:name="info.feelyou.demo.A1"

android:theme="@style/AnimActivityTheme" >

</activity>

<activity

android:name="info.feelyou.demo.A2"

android:theme="@style/AnimActivityTheme" >

</activity>

# Android-WindowAnimation和ActivityAnimation的區別和使用

[**android**](https://www.codetw.com/android/) [**ActivityAnimation**](https://www.codetw.com/ActivityAnimation/) [**使用**](https://www.codetw.com/%E4%BD%BF%E7%94%A8/) [**mLayoutParams**](https://www.codetw.com/mLayoutParams/) [**WindowAnimation**](https://www.codetw.com/WindowAnimation/) [**imageView**](https://www.codetw.com/imageView/) [**activity**](https://www.codetw.com/activity/) [**item>**](https://www.codetw.com/item%3E/) • 2019-01-09

**Android-WindowAnimation和ActivityAnimation的區別和使用**

**1. WindowAnimation和ActivityAnimation的區別主要如下：**

* WindowAnimation包括windowEnterAnimation和windowExitAnimation；ActivityAnimation包含android:activityOpenEnterAnimation，android:activityOpenExitAnimation，android:activityCloseEnterAnimation和android:activityCloseExitAnimation。
* WindowAnimation的控制權大於ActivityAnimation的控制權，即在Activity轉場過程中，如果同時設定了WindowAnimation和ActivityAnimation,那麼可能（因為這種情況非常多）只會執行WindowAnimation。
* WindowAnimation在style.xml中繼承Animation style的定義如下：

<style name="WindowAnim" parent="@android:style/Animation">

<!*--窗體進入動畫-->*

<item name="android:windowEnterAnimation">@anim/tweened</item>

<!*--窗體退出動畫-->*

<item name="android:windowExitAnimation">@anim/alpha</item>

</style>

而ActivityAnimation在style.xml中繼承Animation.Activity的定義如下：

<style name="test\_activity\_anim" parent="@android:style/Animation.Activity">

<item name="android:activityOpenEnterAnimation">@anim/alpha</item>

<item name="android:activityOpenExitAnimation">@android:anim/fade\_out</item>

<item name="android:activityCloseEnterAnimation">@android:anim/fade\_in</item>

<item name="android:activityCloseExitAnimation">@android:anim/fade\_out</item>

</style>

當我們從 A1 啟動 A2 時，A1 從螢幕上消失，這個動畫叫做 android:activityOpenExitAnimation

當我們從 A1 啟動 A2 時，A2 出現在螢幕上，這個動畫叫做 android:activityOpenEnterAnimation

當我們從 A2 退出回到 A1 時，A2 從螢幕上消失，這個叫做 android:activityCloseExitAnimation

當我們從 A2 退出回到 A1 時，A1 出現在螢幕上，這個叫做 android:activityCloseEnterAnimation

從上面WindowAnimation和ActivityAnimation的定義可知，WindowAnimation同一時間只可以控制一個activity的窗體動畫，而ActivityAnimation同一時間可以控制兩個activity的動畫。

**2. WindowAnimation的主要使用方式如下：**

<style name="WindowAnim"parent="@android:style/Animation">

<item name="android:windowEnterAnimation">@anim/tweened</item>

<item name="android:windowExitAnimation">@anim/alpha</item>

</style>

**2.1 在主題中使用**

*<!-- Base application theme. -->*

<style name="AppTheme" parent="android:Theme.Holo.Light">

<!-- Customize your theme here. -->

<item name="android:windowAnimationStyle">@style/WindowAnim</item>

</style>

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

上面這種方式會使得該應用程式的所有activity都使用**WindowAnim**樣式。另外也可以像下面只設置指定的activity的動畫。

<activity android:name=".AnimationActivity"

android:theme="@style/WindowAnim"/>

**注意: 上面這樣動畫只在指定的activity上起效果，其他未設定動畫的activity當然就沒有動畫。**

**2.2 程式碼中的使用**

final ImageView imageView = new ImageView(getApplicationContext())*;*

imageView.setImageDrawable(getResources().getDrawable(R.drawable.ic\_launcher))*;*

imageView.setVisibility(View.VISIBLE)*;*

mLayoutParams = new WindowManager.LayoutParams()*;*

mLayoutParams.windowAnimations = R.style.WindowAnim*; // 窗體的動畫，必須是style中類似WindowAnim的定義樣式*

mLayoutParams.x = 0*;*

mLayoutParams.y = 0*;*

mLayoutParams.width = ViewGroup.LayoutParams.WRAP\_CONTENT*;*

mLayoutParams.height = ViewGroup.LayoutParams.WRAP\_CONTENT*;*

mLayoutParams.gravity = Gravity.CENTER*;*

mLayoutParams.flags |= WindowManager.LayoutParams.FLAG\_NOT\_FOCUSABLE*; // 該窗體以外的部分也可以獲取焦點*

getWindowManager().addView(imageView, mLayoutParams)*;*

上述使用方式的動畫效果必須是在imageView的狀態發生改變時才會觸發，比如imageView由不可見變成可見，或者當mLayoutParams改變時呼叫getWindowManager().updateViewLayout(imageView, mLayoutParam)。

**3. ActivityAnimation的主要使用方式如下：**

<style name="test\_activity\_anim" parent="@android:style/Animation.Activity">

<item name="android:activityOpenEnterAnimation">@anim/alpha</item>

<item name="android:activityOpenExitAnimation">@android:anim/fade\_out</item>

<item name="android:activityCloseEnterAnimation">@android:anim/fade\_in</item>

<item name="android:activityCloseExitAnimation">@android:anim/fade\_out</item>

</style>

**3.1 類似WindowAnimation在主題中使用**

*<!-- Base application theme. -->*

<style name="AppTheme" parent="android:Theme.Holo.Light">

<!-- Customize your theme here. -->

<item name="android:windowAnimationStyle">@style/test\_activity\_anim</item>

</style>

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:supportsRtl="true"

android:theme="@style/test\_activity\_anim">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

上面這種方式會使得該應用程式的所有activity都使用**test\_activity\_anim**樣式。另外也可以像下面只設置指定的activity的動畫。

<activity android:name=".AnimationActivity"

android:theme="@style/test\_activity\_anim"/>

**注意: 上面這樣動畫只指定的activity上起效果，其他未設定動畫的activity當然就沒有動畫。**

**3.2 在程式碼中的使用**

在Activity類中可以使用下面函式來自定義activity的動畫:

*// enterAnim：第二個activity進入的動畫； exitAnim 第一個activity退出的動畫; 引數型別：在anim資料夾下的動畫*

overridePendingTransition(int enterAnim, int exitAnim);

**注意：上面函式的呼叫一般在startActivity()或者finish()之後。**

相關文章

* [關於Cookie和Session的區別及使用](https://www.codetw.com/clpxli.html)
* [「android」 onCreate(),onCreateView()和onActivityCreated()的區別和使用](https://www.codetw.com/hkefhkhk.html)
* [LIB和DLL的區別和使用](https://www.codetw.com/eqpqe.html)
* [「可可 」CGLayer和CALayer的區別和相容性是什麼？](https://www.codetw.com/qqxff.html)
* [C和C++的區別和聯絡（擴充套件知識）](https://www.codetw.com/xhkpxf.html)
* [HTTP和HTTPS的區別和常見的面試題](https://www.codetw.com/hkellqhk.html)
* [【小家java】Session和Cookie的區別和聯絡、分散式session的幾種實現方式](https://www.codetw.com/hkfcyep.html)
* [webservice和socket 的區別和比較!](https://www.codetw.com/hkiiyex.html)
* [Thread和Runnable的區別和聯絡、多次start一個執行緒會怎麼樣](https://www.codetw.com/hkixpchk.html)
* [JSP和Servlet的區別和聯絡（不看後悔，一看必懂）](https://www.codetw.com/celhkey.html)

# Android之基本常见知识 getwindow（） requestWindowFeature（）方法

2017年03月27日 09:00:51 [王东升KevinTech](https://me.csdn.net/sinat_32829963) 阅读数：6837 标签： [android开发](http://so.csdn.net/so/search/s.do?q=android%E5%BC%80%E5%8F%91&t=blog)[androidstudio](http://so.csdn.net/so/search/s.do?q=androidstudio&t=blog)[全屏](http://so.csdn.net/so/search/s.do?q=%E5%85%A8%E5%B1%8F&t=blog)[沉浸状态栏](http://so.csdn.net/so/search/s.do?q=%E6%B2%89%E6%B5%B8%E7%8A%B6%E6%80%81%E6%A0%8F&t=blog)[源码](http://so.csdn.net/so/search/s.do?q=%E6%BA%90%E7%A0%81&t=blog) 更多

个人分类： [Android](https://blog.csdn.net/sinat_32829963/article/category/6857760)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/sinat\_32829963/article/details/66967831

**//能够取得屏幕的信息**

DisplayMetrics dm = new DisplayMetrics();

    getWindowManager().getDefaultDisplay().getMetrics(dm);

dm.widthPixels;                //取得宽像素  
    dm.heightPixels;       //取得高像素

**//获取当地的日历**

Calendar c=Calendar.getInstance();

    mYear=c.get(Calendar.YEAR);//获取年份

    mMonth=c.get(Calendar.MONTH);//获取月份

 mDay=c.get(Calendar.DAY\_OF\_MONTH);//获取号数

    mHour=c.get(Calendar.HOUR\_OF\_DAY);//获取小时

    mMinute=c.get(Calendar.MINUTE); //获取分钟

**//把EditText的内容设为**[**可视**](https://www.baidu.com/s?wd=%E5%8F%AF%E8%A7%86&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)**或隐藏**

/\* 设定EditText的内容为可见的 \*/

     editText.setTransformationMethod(

     HideReturnsTransformationMethod.getInstance());

     /\* 设定EditText的内容为隐藏的 \*/

     editText.setTransformationMethod(

     PasswordTransformationMethod.getInstance());

**//启动activity时不自动弹出软键盘**

        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT\_INPUT\_STATE\_ALWAYS\_HIDDEN);

**//设置全屏(在setContentView之前设置)**

        getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

[Android getWindow().setFlags方法](http://www.cnblogs.com/qingblog/archive/2012/06/01/2529838.html)

            //设置窗体全屏

### getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

            //设置窗体始终点亮

            getWindow().setFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON,

             WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

             //设置窗体背景模糊

             getWindow().setFlags(WindowManager.LayoutParams.FLAG\_BLUR\_BEHIND,  
             WindowManager.LayoutParams.FLAG\_BLUR\_BEHIND);

**//设置取消全屏**

WindowManager.LayoutParams attrs = getWindow().getAttributes();

attrs.flags &= (~WindowManager.LayoutParams.FLAG\_FULLSCREEN); getWindow().setAttributes(attrs); getWindow().clearFlags(WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS);

//设置无标题(在setContentView之前设置)  <http://blog.csdn.net/heng615975867/article/details/8735609>

        requestWindowFeature(Window.FEATURE\_NO\_TITLE);

**//设置禁止手机横屏(在setContentView之前设置)**

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_NOSENSOR);

**//调用手机默认的摄像功能，而且可以设定储存位置**

Intent i = new Intent("[**Android**](http://lib.csdn.net/base/android).media.action.IMAGE\_CAPTURE");

i.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(new File(Environment .getExternalStorageDirectory(),"pic.jpg")));

**//解决中文乱码方法**

A. 使用getBytes("") 来对汉字进行重编码，得到它的字节数组

B. 再使用new String(Bytes[] , "解码方式")  来对字节数组进行相应的解码

**//在Android中轻松实现横竖屏的布局**

竖屏的布局一般在layout下面设置；横屏的布局则在layout的同等级文件夹创建名字layout-land的文件夹。模拟器可以使用Ctrl+F11进行快速切换。

**//Android横竖屏切换不重启Activity**

androidmanifest.xml中的activit元素加入这个属性[android](http://dev.10086.cn/cmdn/wiki/index.php?doc-view-1789):configChanges="orientation|keyboardHidden"

然后在Activity中重载以下方法：

复制代码

          public void onConfigurationChanged(Configuration newConfig) {

     // TODO Auto-generated method stub  
  if (newConfig.orientation==Configuration.ORIENTATION\_LANDSCAPE) {  
            setContentView(R.layout.imageswitch);  
            //横屏  
        } else {  
            setContentView(R.layout.editcontact);//竖屏  
        }  
         super.onConfigurationChanged(newConfig);

}

复制代码

[卸载程序](https://www.baidu.com/s?wd=%E5%8D%B8%E8%BD%BD%E7%A8%8B%E5%BA%8F&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)：  
  
Uri packageURI = Uri.parse("package:com.demo.CanavaCancel");

Intent uninstallIntent = new Intent(Intent.ACTION\_DELETE, packageURI);

startActivity(uninstallIntent);

安装apk：

String str = "/CanavaCancel.apk";

String fileName = Environment.getExternalStorageDirectory() + str;

Intent intent = new Intent(Intent.ACTION\_VIEW);

 intent.setDataAndType(Uri.fromFile(new File(fileName)), "application/vnd.android.package-archive");

startActivity(intent);

# [requestWindowFeature()的应用](http://blog.csdn.net/heng615975867/article/details/8735550" \t "_blank)

android开发中经常会在setContentView(R.layout.XXX); 前设置requestWindowFeature(XXXX)。

他的意思是**需要软件全屏显示、自定义标题（使用按钮等控件）和其他的需求**

**首先介绍一个重要方法那就是requestWindowFeature(featrueId),它的功能是启用窗体的扩展特性。参数是Window类中定义的常量。**

**一、枚举常量**

1.DEFAULT\_FEATURES：系统默认状态，一般不需要指定

2.FEATURE\_CONTEXT\_MENU：启用ContextMenu，默认该项已启用，一般无需指定

**3.FEATURE\_CUSTOM\_TITLE：自定义标题。**当需要自定义标题时必须指定。如：标题是一个按钮时

4.FEATURE\_INDETERMINATE\_PROGRESS：不确定的进度

5.FEATURE\_LEFT\_ICON：标题栏左侧的图标

6.FEATURE\_NO\_TITLE：没有标题

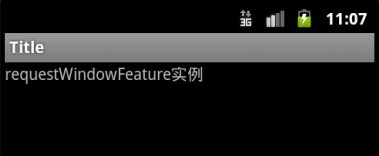
7.FEATURE\_OPTIONS\_PANEL：启用“选项面板”功能，默认已启用。

8.FEATURE\_PROGRESS：进度指示器功能

9.FEATURE\_RIGHT\_ICON:标题栏右侧的图标

**二、详解**

默认显示状态



**1.FEATURE\_CUSTOM\_TITLE详解**

Java代码  收藏代码

1. **this**.requestWindowFeature(Window.FEATURE\_CUSTOM\_TITLE);
2. setContentView(R.layout.main);



这是因为没有设置Featrue

在上面代码后加：getWindow().setFeatureInt(Window.FEATURE\_CUSTOM\_TITLE, R.layout.title);



自定义标题完成，它是一个xml文件布局

Xml代码  收藏代码

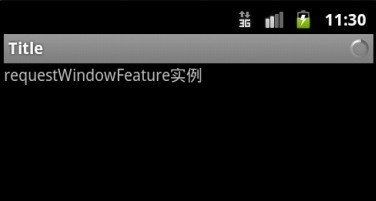
1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="wrap\_content"
4. android:layout\_height="wrap\_content" **>**
6. **<ImageView**
7. android:layout\_width="wrap\_content"
8. android:layout\_height="wrap\_content"
9. android:src="@drawable/ic\_launcher"
10. **/>**
12. **<TextView**
13. android:id="@+id/text"
14. android:layout\_width="wrap\_content"
15. android:layout\_height="wrap\_content"
16. android:layout\_alignParentLeft="true"
17. android:textColor="#000000"
18. android:text="FEATURE\_CUSTOM\_TITLE" **/>**
20. **</LinearLayout>**

**2.FEATURE\_INDETERMINATE\_PROGRESS详解**

可以用来表示一个进程正在运行

Java代码  收藏代码

1. **this**.requestWindowFeature(Window.FEATURE\_INDETERMINATE\_PROGRESS);
2. setContentView(R.layout.main);
3. getWindow().setFeatureInt(Window.FEATURE\_INDETERMINATE\_PROGRESS, R.layout.progress);
4. setProgressBarIndeterminateVisibility(**true**);



Xml代码  收藏代码

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="wrap\_content"
4. android:layout\_height="wrap\_content" **>**
6. **<ProgressBar**
7. android:id="@+id/progress"
8. style="?android:attr/progressBarStyleSmallTitle"
9. android:layout\_width="wrap\_content"
10. android:layout\_height="wrap\_content"
11. android:layout\_gravity="center\_vertical" **>**
12. **</ProgressBar>**
14. **</LinearLayout>**

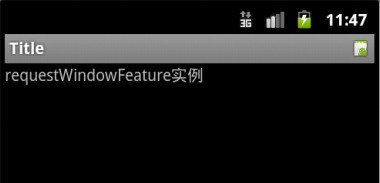
**3.FEATURE\_LEFT\_ICON和FEATURE\_RIGHT\_ICON详解**

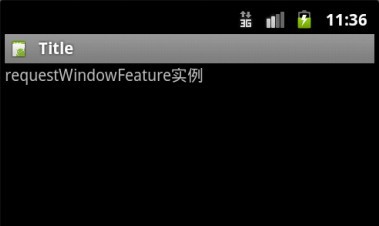
Java代码  收藏代码

1. requestWindowFeature(Window.FEATURE\_RIGHT\_ICON);
2. setContentView(R.layout.main);
3. getWindow().setFeatureDrawableResource(Window.FEATURE\_RIGHT\_ICON,R.drawable.ic\_launcher);

Java代码  收藏代码

1. requestWindowFeature(Window.FEATURE\_LEFT\_ICON);
2. setContentView(R.layout.main);
3. getWindow().setFeatureDrawableResource(Window.FEATURE\_LEFT\_ICON,R.drawable.ic\_launcher);





**4.FEATURE\_NO\_TITLE详解**

Java代码  收藏代码

1. **this**.requestWindowFeature(Window.FEATURE\_NO\_TITLE);
2. setContentView(R.layout.main);



Java代码  收藏代码

1. getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);



# Android Animation完全总结（四）其他

[Android](http://www.paincker.com/category/hacker/android)  1,695℃ [0](http://www.paincker.com/android-animation-4#respond) 3年前 (2016-06-03)

示例代码、思维导图可在此下载   
<https://github.com/jzj1993/AndroidAnimation>

## Dialog View Animation

给Dialog中的View设置动画，和给普通View设置动画的写法一样。

1. Animation anim = AnimationUtils.loadAnimation(this, R.anim.animation);
2. View dialogLayout = LayoutInflater.from(this).inflate(R.layout.dialog, null);
3. View v = dialogLayout.findViewById(R.id.dialog\_text);
4. v.startAnimation(anim);
5. new AlertDialog.Builder(this).setView(dialogLayout).show();

## Dialog Window Animation

给整个Dialog设置动画（即Window对象）

1. <style name="dialogAnimation" parent="@android:style/Animation.Dialog">
2. <item name="android:windowEnterAnimation">@anim/anim\_in</item>
3. <item name="android:windowExitAnimation">@anim/anim\_out</item>
4. </style>
5. Dialog dialog = new AlertDialog.Builder(this)
6. .setMessage("Message")
7. .setPositiveButton("OK", null)
8. .create();
9. dialog.getWindow().setWindowAnimations(R.style.dialogAnimation);
10. dialog.show();

## PopupWindow Animation

XML中的定义和Dialog相同

1. <style name="dialogAnimation" parent="@android:style/Animation.Dialog">
2. <item name="android:windowEnterAnimation">@anim/anim\_in</item>
3. <item name="android:windowExitAnimation">@anim/anim\_out</item>
4. </style>

Java代码中设置动画

1. mPopupWindow.setAnimationStyle(R.style.dialogAnimation);

## Activity Animation From XML

给Activity页面跳转设置动画

1. <!-- 1.定义Activity动画 -->
2. <style name="activityAnimation" parent="@android:style/Animation.Activity">
3. <item name="android:windowEnterAnimation">@null</item>
4. <item name="android:windowExitAnimation">@null</item>
5. <!-- 新Activity启动时，Enter动画 -->
6. <item name="android:activityOpenEnterAnimation">@anim/anim\_in</item>
7. <!-- 新Activity启动时，原有Activity的Exit动画 -->
8. <item name="android:activityOpenExitAnimation">@anim/stay</item>
9. <!-- 新Activity退出时，原有Activity的Enter动画 -->
10. <item name="android:activityCloseEnterAnimation">@anim/stay</item>
11. <!-- 新Activity退出时，Exit动画 -->
12. <item name="android:activityCloseExitAnimation">@anim/anim\_out</item>
13. </style>
14. <!-- 2. AppTheme中引用Activity动画 -->
15. <style name="AppTheme" parent="android:Theme.Holo.Light.NoActionBar">
16. <item name="android:windowAnimationStyle">@style/activityAnimation</item>
17. </style>
18. <!-- 3. Manifest中的Application或Activity标签中，指定theme为AppTheme -->
19. <application
20. android:theme="@style/AppTheme">

**注意：**

* 如果Activity的launchMode设置为SingleInstance，可能会导致动画失效
* 部分手机需要在设置中开启显示全部动画。
* 某些手机由于定制了ROM，通过XML自定义Activity切换动画无效，只能用Java实现。

## Activity Animation From Java

1. startActivity(new Intent(this, NewActivity.class));
2. overridePendingTransition(R.anim.anim\_in, R.anim.anim\_out); // 设置为0则表示没有动画

在部分机型中，设置为0表示不显示动画可能没效果，可以定义一个没有实际执行效果的动画代替：   
res/anim/stay.xml

1. <?xml version="1.0" encoding="utf-8"?>
2. <alpha
3. xmlns:android="http://schemas.android.com/apk/res/android"
4. android:duration="1000"
5. android:fromAlpha="1.0"
6. android:toAlpha="1.0" />

最后，欢迎扫码关注微信公众号。

# 对WindowManager中的View设置动画

[](https://www.jianshu.com/u/9607292720d8)

[HunterGao35](https://www.jianshu.com/u/9607292720d8) 关注

2017.11.07 16:10\* 字数 261 阅读 837评论 0喜欢 0

# 1.直接对View设置传统动画

传统动画包括帧动画和补间动画。帧动画主要是一帧一帧的播放。可以在xml中使用<animation-list>标签设置，也可以在代码中使用AnimationDrawable设置；补间动画主要包括alpha, translate, scale, rotate。

ScaleAnimation animation = new ScaleAnimation(0.0f, 1f, 0.0f, 1f, Animation.ABSOLUTE, 100, Animation.ABSOLUTE, 100);

animation.setDuration(100);

view.setAnimation(animation);

animation.start();

mWindowManager.addView(defaultSplashLayout, lp);

进行这样的设置，view的动画无法生效。原因是动画执行的条件是不能直接添加到最顶层的Window，而是需要一个容器。  
如果添加一个容器，则只能对容器内的view进行动画设置，还是无法对容器进行动画设置。

# 2.对WindowManager.LayoutParams的windowAnimations设置动画

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.type = WindowManager.LayoutParams.TYPE\_STATUS\_BAR\_PANEL;

lp.flags = WindowManager.LayoutParams.FLAG\_HARDWARE\_ACCELERATED;

lp.format = PixelFormat.RGB\_888;

lp.screenOrientation = ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT;

lp.windowAnimations = R.style.default\_style;

<style name="default\_style">

<item name="@android:windowEnterAnimation">@anim/window\_enter</item>

<item name="@android:windowExitAnimation">@anim/window\_exit</item>

</style>

这样设置以后，这个view都会执行动画，但是动画都是写死在xml文件中的，无法进行动态设置

# 3.对View设置属性动画

属性动画对最顶层的view是可以执行的。

view.setPivotX(100);

view.setPivotY(100);

ValueAnimator valueAnimator = ValueAnimator.ofFloat(0, 1);

valueAnimator.setDuration(200).start();

valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator animation) {

Float value = (Float) animation.getAnimatedValue();

view.setScaleX(value);

view.setScaleY(value);

}

});

mWindowManager.addView(defaultSplashLayout, lp)

# 从源码的角度分析WindowManager的addview与ViewGroup的addview的区别

[](https://www.jianshu.com/u/4a379d9960f9)

[走在路上的日子](https://www.jianshu.com/u/4a379d9960f9) 关注

2018.08.01 15:02\* 字数 981 阅读 271评论 1喜欢 1

首先我们要知道ViewManager接口：

package android.view;

public interface ViewManager

{

//添加View 我们主要看在ViewGroup和WindowManagerImpl中的实现

public void addView(View view, ViewGroup.LayoutParams params);

//更新View的布局

public void updateViewLayout(View view, ViewGroup.LayoutParams params);

//移除View

public void removeView(View view);

}

可以很清楚滴看到这接口是添加、更新和移除View的。其实还有一个很有意思的ViewParent接口，这里不讲了，后面的博客我会说。

### ViewGroup实现ViewManager接口

public abstract class ViewGroup extends View implements ViewParent, ViewManager {

...

public void addView(View child, int index, LayoutParams params) {

if (DBG) {

System.out.println(this + " addView");

}

if (child == null) {

throw new IllegalArgumentException("Cannot add a null child view to a ViewGroup");

}

// addViewInner() will call child.requestLayout() when setting the new LayoutParams

// therefore, we call requestLayout() on ourselves before, so that the child's request

// will be blocked at our level

requestLayout();

invalidate(true);

addViewInner(child, index, params, false);

}

...

}

### WindowManagerImpl实现WindowManager接口:

WindowManager接口继承了ViewManager接口

public final class WindowManagerImpl implements WindowManager {

...

@Override

public void addView(@NonNull View view, @NonNull ViewGroup.LayoutParams params) {

applyDefaultToken(params);

mGlobal.addView(view, params, mContext.getDisplay(), mParentWindow);

}

...

}

这两个地方虽然都用了ViewManager接口，但是作用很不一样，下面我们来一步步验证吧。

# 一、Activity正常写法时页面结构

先了解一下Activity窗口的结构和组成：

Activity界面结构

上面结构图也是展示的正常写法时的样子，一个Activity对应一个PhoneWindow，一个PhoneWindow对应一个DecorView，DecorView其实就是一个FrameLayout控件，这个DecorView才是一个我们看到的界面的根View，一个FrameLayout类型的ViewGroup。其实DecorView中也是有一个xml布局文件的，在给DecorView填充布局文件时回根据对Activity设置不同的style来决定填充哪一个布局文件，下面是大概的源代码：

protected ViewGroup generateLayout(DecorView decor) {

...

// System.out.println("Title!");

} else if ((features & (1 << FEATURE\_ACTION\_MODE\_OVERLAY)) != 0) {

layoutResource = R.layout.screen\_simple\_overlay\_action\_mode;

} else {

// Embedded, so no decoration is needed.

layoutResource = R.layout.screen\_simple;

// System.out.println("Simple!");

}

mDecor.startChanging();

//把需要填充的布局文件addView的方式加入到FrameLayout类型的DecorView

mDecor.onResourcesLoaded(mLayoutInflater, layoutResource);

//ID\_ANDROID\_CONTENT 这个id就是我们Activity setContentView 操作的父布局的id

ViewGroup contentParent = (ViewGroup)findViewById(ID\_ANDROID\_CONTENT);

...

}

上面代码中DecorView有一个简单的填充布局screen\_simple.xml，我们来看一下这个xml文件：

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:fitsSystemWindows="true"

android:orientation="vertical">

<ViewStub android:id="@+id/action\_mode\_bar\_stub"

android:inflatedId="@+id/action\_mode\_bar"

android:layout="@layout/action\_mode\_bar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:theme="?attr/actionBarTheme" />

<FrameLayout

android:id="@android:id/content"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:foregroundInsidePadding="false"

android:foregroundGravity="fill\_horizontal|top"

android:foreground="?android:attr/windowContentOverlay" />

</LinearLayout>

其实Activity 设置窗口无标题栏requestWindowFeature(Window.FEATURE\_NO\_TITLE);就会加screen\_simple.xml这个布局文件。

好了，上面算是大概讲了一下Activity的结构了，看一下我们正常情况下在Activity中setContentView 的页面结构。  
首先是我们的activity\_test.xml布局文件：

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

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

android:id="@+id/mLLayout"

android:gravity="center"

android:orientation="horizontal"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<Button

android:id="@+id/mBtn1"

android:layout\_width="@dimen/d\_180dp"

android:layout\_height="@dimen/d\_120dp"

android:clickable="true"

android:focusable="true"

android:text="mBtn1"

android:focusableInTouchMode="true" />

<Button

android:id="@+id/mBtn2"

android:layout\_width="@dimen/d\_180dp"

android:layout\_height="@dimen/d\_120dp"

android:clickable="true"

android:focusable="true"

android:text="mBtn2"

android:focusableInTouchMode="true" />

</LinearLayout>

布局很简单，就是一个LinearLayout中放了两个按钮而已。  
ActivityTest.java代码：

public class ActivityTest extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

}

}

Activity的代码很简单，上面都没做。  
我们使用Android Studio的layout inspector工具看看页面结构：

image.png

# 二、Activity addContentView后的结构

activity\_test.xml布局文件还是上面的布局文件，没有改动过，这里就不看了，只是在Activity的代码中动态滴加了一个Button:

public class ActivityTest extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

Button button = new Button(this);

button.setText("AddBtn");

LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

addContentView(button, layoutParams);

}

}

看一下布局结构：

image.png

我们加的一个Button直接加入到了DecorView中id为content的FrameLayout中了，跟我们setContent加入的布局文件是平行的关系。  
那么我们看一下源代码：

@Override

public void addContentView(View view, ViewGroup.LayoutParams params) {

//mContentParent 就是DecorView中id为content的FrameLayout

if (mContentParent == null) {

//setContent也要走这一步

installDecor();

}

if (hasFeature(FEATURE\_CONTENT\_TRANSITIONS)) {

// TODO Augment the scenes/transitions API to support this.

Log.v(TAG, "addContentView does not support content transitions");

}

//这里直接将view加入到了DecorView中id为content的FrameLayout中了，

//所以多次调用addContentView添加的View都是在id为content的FrameLayout中为平行关系

mContentParent.addView(view, params);

mContentParent.requestApplyInsets();

final Callback cb = getCallback();

if (cb != null && !isDestroyed()) {

cb.onContentChanged();

}

}

# 三、Activity getWindowManager().addView有什么不同？

我们直接试一下好了，ActivityTest.java代码如下：

public class ActivityTest extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

Button button = new Button(this);

button.setText("addContentView AddBtn");

FrameLayout.LayoutParams layoutParams = new FrameLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

addContentView(button, layoutParams);

LinearLayout linearLayout = new LinearLayout(this);

Button buttonWM = new Button(this);

buttonWM.setText("WM addView AddBtnWM");

LinearLayout.LayoutParams layoutParams1L = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

layoutParams1L.gravity = Gravity.BOTTOM;

linearLayout.addView(buttonWM, layoutParams1L);

WindowManager.LayoutParams layoutParams1WM = getWindow().getAttributes();

getWindowManager().addView(linearLayout,layoutParams1WM);

}

}

运行我们查看布局的时候出现了：

image.png

分别打开两个页面结构：

image.png

image.png

其实出现了跟DecorView平级的一个布局，其实我们getWindowManager().addView的流程是：

getWindowManager().addView—》WindowManagerGlobal.addView—》下面就是代码了：

public void addView(View view, ViewGroup.LayoutParams params,

Display display, Window parentWindow){

...

//创建新的ViewRootImpl

root = new ViewRootImpl(view.getContext(), display);

view.setLayoutParams(wparams);

mViews.add(view);

mRoots.add(root);

mParams.add(wparams);

// do this last because it fires off messages to start doing things

try {

root.setView(view, wparams, panelParentView);

}

...

}

看代码就是生成了一个DecorView级别的布局，存在于PhoneWindow中。看代码我们发现Activity中有一个方法：

void makeVisible() {

if (!mWindowAdded) {

ViewManager wm = getWindowManager();

//将DecorView、ViewRootImpl、PhoneWindow、WindowManagerImpl关联起来

wm.addView(mDecor, getWindow().getAttributes());

mWindowAdded = true;

}

mDecor.setVisibility(View.VISIBLE);

}

这个方法其实就是Activity被创建后ActivityThread类的handleResumeActivity方法中被调用的，就是将DecorView与WindowManagerImpl 关联，WindowManagerImpl的addview方法add的是DecorView。

# 四、扒一扒ViewGroup的addView

直接看源代码：

public void addView(View child, int index, LayoutParams params) {

if (DBG) {

System.out.println(this + " addView");

}

if (child == null) {

throw new IllegalArgumentException("Cannot add a null child view to a ViewGroup");

}

// addViewInner() will call child.requestLayout() when setting the new LayoutParams

// therefore, we call requestLayout() on ourselves before, so that the child's request

// will be blocked at our level

//先重新布局一下

requestLayout();

//刷新一下

invalidate(true);

//正式addView

addViewInner(child, index, params, false);

}

加下来看addViewInner方法：

private void addViewInner(View child, int index, LayoutParams params,

boolean preventRequestLayout) {

...

if (preventRequestLayout) {

child.mLayoutParams = params;

} else {

//设置新的layoutparams会调用requestLayout()方法

child.setLayoutParams(params);

}

if (index < 0) { //计算加入的View在

index = mChildrenCount;

}

//

addInArray(child, index);

...

}

addInArray方法：

private void addInArray(View child, int index) {

View[] children = mChildren;

final int count = mChildrenCount;

final int size = children.length;

if (index == count) {

if (size == count) {

mChildren = new View[size + ARRAY\_CAPACITY\_INCREMENT];

System.arraycopy(children, 0, mChildren, 0, size);

children = mChildren;

}

children[mChildrenCount++] = child;

} else if (index < count) {

if (size == count) {

mChildren = new View[size + ARRAY\_CAPACITY\_INCREMENT];

System.arraycopy(children, 0, mChildren, 0, index);

System.arraycopy(children, index, mChildren, index + 1, count - index);

children = mChildren;

} else {

System.arraycopy(children, index, children, index + 1, count - index);

}

children[index] = child;

mChildrenCount++;

if (mLastTouchDownIndex >= index) {

mLastTouchDownIndex++;

}

} else {

throw new IndexOutOfBoundsException("index=" + index + " count=" + count);

}

}

就是将需要add的view加入到ViewGroup的View[] mChildren数组中去，加入接下来的绘制会用for循环来measureChildren、layoutChildren和dispatchDraw来处理数组中的每一个子view，但是上面看到 requestLayout()和invalidate(true)发生 addViewInner()方法之前，感觉有点没看懂，在我们手动向ViewGroup中addView后习惯性调用invalidate()方法，保证界面及时刷新。

**注意：**  
不管是WindowManagerImpl还是ViewGroup的addView(View view, ViewGroup.LayoutParams params)方法都是来自于ViewManager接口，但是在ViewGroup中是将普通的view或者ViewGroup作为Children加入，但是在WindowManagerImpl是将DecorView作为根布局加入到PhoneWindow中去，虽然都是基于同一个ViewManager接口，但是起作用截然不同.

原文链接：[WindowManager的addview与ViewGroup的addview有什么不同？](https://www.jianshu.com/p/43880dc0baa7)

# [Android 基础】Animation 动画介绍和实现](https://www.cnblogs.com/yc-755909659/p/4290114.html)

在前面[PopupWindow 实现显示仿腾讯新闻底部弹出菜单](http://www.cnblogs.com/yc-755909659/p/4288864.html)有用到Animation动画效果来实现菜单的显示和隐藏，本文就来介绍下吧。

**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

[复制代码](javascript:void(0);)

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

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

<alpha />

<scale />

<translate />

<rotate />

</set>

[复制代码](javascript:void(0);)

**4.Android动画解析--XML**

4.1 alpha 渐变透明度动画效果

[复制代码](javascript:void(0);)

<?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>

[复制代码](javascript:void(0);)

4.2 scale 渐变尺寸伸缩动画效果

[复制代码](javascript:void(0);)

<?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

-->

[复制代码](javascript:void(0);)

4.3 translate 画面转换位置移动动画效果

[复制代码](javascript:void(0);)

<?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>

[复制代码](javascript:void(0);)

4.4 rotate 画面转移旋转动画效果

[复制代码](javascript:void(0);)

<?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>

[复制代码](javascript:void(0);)

5.**如何使用XML中的动画效果**

[复制代码](javascript:void(0);)

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

[复制代码](javascript:void(0);)

6.**如何使用XML中的动画效果**

[复制代码](javascript:void(0);)

//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);

[复制代码](javascript:void(0);)

**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

[复制代码](javascript:void(0);)

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);

[复制代码](javascript:void(0);)

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

[复制代码](javascript:void(0);)

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);

[复制代码](javascript:void(0);)

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**

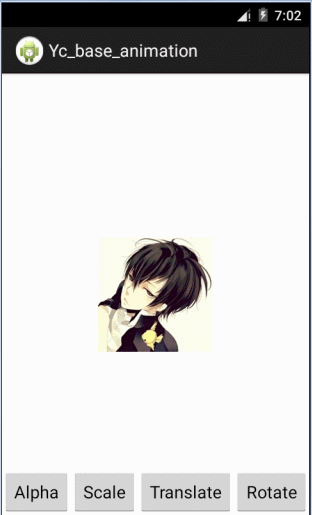
 使用从View父类继承过来的方法startAnimation()来为View或是子类View等等添加一个动画效果

public void startAnimation (Animation animation)

**9.还是来个栗子吧**

**9.1 使用XML文件方式**

①效果图如下：



②在XML文件中定义动画，前面已提及

③主界面布局，这没啥好说的，很简单 o(∩\_∩)o

④主界面逻辑代码，主要就是这个了，控制动画显示

[复制代码](javascript:void(0);)

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;

}

}

}

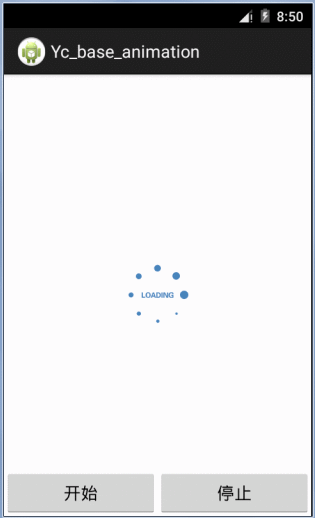
[复制代码](javascript:void(0);)

**9.2 使用Java代码方式**

博文 [游戏开发基础（动画）](http://www.cnblogs.com/yc-755909659/p/4171732.html) 中有实例说明，此处不再赘述。

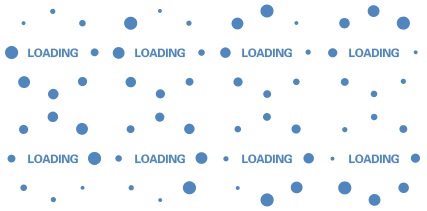
**10. 用Animation-list实现逐帧动画**

栗子效果图如下：



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

[复制代码](javascript:void(0);)

<?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>

[复制代码](javascript:void(0);)

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：

[复制代码](javascript:void(0);)

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;

}

}

}

[复制代码](javascript:void(0);)

本文地址：<http://www.cnblogs.com/yc-755909659/p/4290114.html>

* [Linux](https://codeday.me/tag/linux)
* [Python](https://codeday.me/tag/python)
* [Java](https://codeday.me/tag/java)
* [C#](https://codeday.me/tag/c)
* [Docker](https://codeday.me/tag/docker)
* [Nginx](https://codeday.me/tag/nginx)
* [Mysql](https://codeday.me/tag/mysql)
* [Spring](https://codeday.me/tag/spring)
* [Android](https://codeday.me/tag/android)
* [开源项目](https://codeday.me/category/news)
* [图库](https://codeday.me/category/process)

窗体顶端



窗体底端

#### [java – WindowManager with Animation(可以吗？)](https://codeday.me/bug/20181102/350724.html)

时间：2018-11-02 标签：[android](https://codeday.me/tag/android)[animation](https://codeday.me/tag/animation)[java](https://codeday.me/tag/java)[view](https://codeday.me/tag/view)

有没有办法用WindowManager使用动画(在[android](https://codeday.me/tag/android)的项目)来夸张一个视图？我甚至不能使用网站上的示例！我使用了很多例子,但没有工作！

public BannerLayout(Activity activity, final Context context) {

super(context);

this.context = context;

final WindowManager.LayoutParams params = new WindowManager.LayoutParams(

WindowManager.LayoutParams.TYPE\_SYSTEM\_OVERLAY,

WindowManager.LayoutParams.FLAG\_LAYOUT\_IN\_SCREEN |

WindowManager.LayoutParams.FLAG\_WATCH\_OUTSIDE\_TOUCH,

PixelFormat.TRANSLUCENT);

wm = (WindowManager) context.getSystemService(Context.WINDOW\_SERVICE);

LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

this.popupLayout = (RelativeLayout) inflater.inflate(R.layout.popup\_activity, null);

this.popupLayout.setVisibility(GONE);

this.setActive(false);

wm.addView(this.popupLayout, params);

context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

}

private void show(){

Animation in = AnimationUtils.loadAnimation(this.context, android.R.anim.fade\_in);

this.popupLayout.setAnimation(in);

this.setActive(true);

this.popupLayout.setVisibility(VISIBLE);

}

我不确定您的任务的确切要求,但有两种方法可以向窗口提供动画：

>使用WindowManager.LayoutParams.windowAnimations,如下所示：

params.windowAnimations = android.R.style.Animation\_Translucent;

>添加附加的“容器”视图,因为WindowManager不是一个真正的ViewGroup,所以正常的动画添加视图不能使用它. [This question has been asked already](https://stackoverflow.com/questions/17537672/animate-system-alert-type-view),但是它缺少代码.我将以下列方式申请：

public class BannerLayout extends View {

private final Context mContext;

private final ViewGroup mPopupLayout;

private final ViewGroup mParentView;

public BannerLayout(Activity activity, final Context context) {

super(context);

mContext = context;

final WindowManager.LayoutParams params = new WindowManager.LayoutParams(

WindowManager.LayoutParams.TYPE\_SYSTEM\_OVERLAY,

WindowManager.LayoutParams.FLAG\_LAYOUT\_IN\_SCREEN |

WindowManager.LayoutParams.FLAG\_WATCH\_OUTSIDE\_TOUCH,

PixelFormat.TRANSLUCENT);

final WindowManager mWinManager = (WindowManager) context.getSystemService(Context.WINDOW\_SERVICE);

LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

mPopupLayout = (RelativeLayout) inflater.inflate(R.layout.popup\_activity, null);

mPopupLayout.setVisibility(GONE);

params.width = ActionBar.LayoutParams.WRAP\_CONTENT;

params.height = ActionBar.LayoutParams.WRAP\_CONTENT;

// Default variant

// params.windowAnimations = android.R.style.Animation\_Translucent;

mParentView = new FrameLayout(mContext);

mWinManager.addView(mParentView, params);

mParentView.addView(mPopupLayout);

mPopupLayout.setVisibility(GONE);

}

/\*\*

\* Shows view

\*/

public void show(){

final Animation in = AnimationUtils.loadAnimation(this.mContext, android.R.anim.fade\_in);

in.setDuration(2000);

mPopupLayout.setVisibility(VISIBLE);

mPopupLayout.startAnimation(in);

}

/\*\*

\* Hides view

\*/

public void hide() {

mPopupLayout.setVisibility(GONE);

}

}

相关文章

* [android - Windowmanager with Animation](https://codeday.me/bug/20181213/444342.html)