Android 画图学习

Android 画图学习总结（一）——类的简介

学习 Android 有一段时间了，看完了 Android SDK 中的大部分文档，但是始终感觉自己还缺少很多，后来发现，Android SDK 中只是介绍了 Google 自己开发的那一部分如何使用，Android 中引用至 Java 的部分如何使用却没有说明。当然这也不是 Google 的职 责，但是这对我们 C++程序员来说的确是缺少了很多，在这里我们将对 Google“缺少的部分”并结合 Android SDK 中 Reference 说明来详细介绍，并不断的补充完善。

首先，如何获取 res 中的资源

数据包 package：android.content.res
主要类：Resources
Android SDK 中的简介：Class for accessing an application’s resources.Class for accessing an application’s resources.

This sits on top of the asset manager of the application (accessible through getAssets()) and provides a higher-level API for getting typed data from the assets.
其主要接口按照功能，划分为以下三部分：

getXXXX() 例如： int getColor(int id) Drawable getDrawable(int id) String getString(int id)	直接获取 res 中存放的资源
InputStream openRawResource(int id)	获取资源的数据流，读取资源数据
void parseBundleExtras(XmlResourceParser parser, Bundle outBundle)	从 XML 文件中获取数据

Resource 为每种资源提供了相应的接口来获取这种资源，除了可以直接获取资源外，还额外提供了以数据流的方式获取资源，这在以后的应用程序开发中会经常使用，那么如何获取 Resources 了，如下：Resources r = this.getContext().getResources();

其次，如何获取资源中的画图对象

数据包 package：android.graphics.drawable
主要类：Drawable
Android SDK 中的简介：A Drawable is a general abstraction for “something that can be drawn.” Most often you will deal with Drawable as the type of resource retrieved for drawing things to the screen; the Drawable class provides a generic

API for dealing with an underlying visual resource that may take a variety of forms.
看了以上简介，发现 Drawable 是个 virtual class，具体如何画图，需要具体分析 Drawable 的子类，例如：BitmapDrawable
Android SDK 中的简介：A Drawable that wraps a bitmap and can be tiled, stretched, or aligned. You can create a BitmapDrawable from a file path, an input stream, through XML inflation, or from a Bitmap object. It can be defined in an XML file with the <bitmap> element.
其主要接口如下：

BitmapDrawable() BitmapDrawable(Bitmap bitmap) BitmapDrawable(String filepath) BitmapDrawable(InputStream is)	
void draw(Canvas canvas)	Draw in its bounds (set via setBounds) respecting optional effects such as alpha (set via setAlpha) and color filter (set via setColorFilter).
final Bitmap getBitmap()	
final Paint getPaint()	

Drawable 是个抽象类，在 BitmapDrawable 中我们就看到位图的具体操作，在仔细看下 BitmapDrawable 的构造函数，我们就会发现与 Resource 中的 openRawResource()接口是相对应的,就可以通过以下方法来获取位图：

```
Resources r = this.getContext().getResources();
Inputstream is = r.openRawResource(R.drawable.my_background_image);
BitmapDrawable  bmpDraw = new BitmapDrawable(is);
```

Bitmap bmp = bmpDraw.getBitmap();
关于 Drawable 深入的学习与理解，请阅读 Android 画图学习总结（三）——Drawable

然后，看几个常用的辅助类

1. Paint
   数据包 package：android.graphics
   Android SDK 中的简介：The Paint class holds the style and color information about how to draw geometries, text and bitmaps. 主要就是定义：画刷的样式，画笔的大小/颜色等。
2. Typeface
   数据包 package：android.graphics
   Android SDK 中的简介：The Typeface class specifies the typeface and intrinsic style of a font. 主要就是定义：字体。

最后，核心类显示资源

数据包 package：android.graphics
主要类：Canvas
Android SDK 中的简介：The Canvas class holds the “draw” calls. To draw something, you need 4 basic components: A Bitmap to hold the pixels, a Canvas to host the draw calls (writing into the bitmap), a drawing primitive (e.g. Rect, Path, text, Bitmap), and a paint (to describe the colors and styles for the drawing).
按照结构的功能，将主要接口分为以下 3 部分：

boolean clipXXXX()	Region 区域操作： DIFFERENCE

	INTERSECT REPLACE REVERSE_DIFFERENCE UNION XOR
void drawXXXX()	画图函数
void rotate() void scale() void skew() void translate()	画布操作函数

Region 在这里需要特殊说明下：Region 就是一个区域，也就是画布（Canvas）中的有效区域，在无效区域上 draw，对画布没有任何改变。

总结说明

在写代码前，必须先仔细看下这几个主要的类，在这里我也只是把 SDK 中的介绍稍微总结下，它代替不了你对 SDK 的详细阅读，毕竟 SDK 是最详细的说明文档，在后续篇幅中再深入详细的介绍。

相关文章

- Activity 、Intent 深入解析
- Android 实现联网（一）——package 说明
- Android 画图学习总结（五）——Paint

- Android 画图学习总结（四）——Animation（下）
- Android 画图学习总结（四）——Animation（中）

Android 画图学习总结（二）——Bitmap

通过前一篇的 学习，对 Android 画图核心部分有了一定的了解，后面篇幅，我们将详细介绍 Android 中的各种画图对象的使用，首先介绍我们最常用的 Bitmap(位图)。位图是我们开 发中最常用的资源，毕竟一个漂亮的界面对用户是最有吸引力的。按照对位图的操作，分为以下几个功能分别介绍：

1. 从资源中获取位图
2. 获取位图的信息
3. 显示位图
4. 位图缩放
5. 位图旋转

1. 从资源中获取位图

在前一篇幅介绍了：先获取 Resource，然后可以通过资源 ID 获取 Drawable，也可以通过资源 ID 获取资源文件的数据流。使用第一种方法 比较容易，下面详细说明第二种方法。通过 Resource 的函数：InputStream openRawResource(int id)获取得到资源文件的数据流后，也可以通过 2 种方法来获取 Bitmap，如下：

使用 BitmapDrawable (A Drawable that wraps a bitmap and can be tiled, stretched, or aligned.)

1. 使用 BitmapDrawable (InputStream is)构造一个 BitmapDrawable； 2. 使用 BitmapDrawable 类的 getBitmap()获取得到位图； BitmapDrawable 也提供了显示位图等操作
使用 BitmapFactory （Creates Bitmap objects from various sources, including files, streams, and byte-arrays.）
1. 使用 BitmapFactory 类 decodeStream(InputStream is)解码位图资源，获取位图 BitmapFactory 的所有函数都是 static，这个辅助类可以通过资源 ID、路径、文件、数据流等方式来获取位图。

以上方法在编程的时候可以自由选择，在 Android SDK 中说明可以支持的图片格式如下：png (preferred), jpg (acceptable), gif (discouraged)，虽然 bmp 格式没有明确说明，但是在 Android SDK Support Media Format 中是明确说明了。

2. 获取位图的信息

要获取位图信息，比如位图大小、是否包含透明度、颜色格式等，获取得到 Bitmap 就迎刃而解了，这些信息在 Bitmap 的函数中可以轻松获取到。Android SDK 中对 Bitmap 有详细说明，阅读起来也比较容易，不在此详细说明，这里只是辅助说明以下 2 点：

- 在 Bitmap 中对 RGB 颜色格式使用 Bitmap.Config 定义，仅包括 ALPHA_8、ARGB_4444、ARGB_8888、RGB_565，缺少了一些其他的，比如说 RGB_555，在开发中可能需要注意这个小问题；
- Bitmap 还提供了 compress()接口来压缩图片，不过 AndroidSAK 只支持 PNG、JPG 格式的压缩；其他格式的需要 Android 开发人员自己补充了。

3. 显示位图

显示位图需要使用核心类 Canvas，可以直接通过 Canvas 类的 drawBirmap()显示位图，或者借助于 BitmapDrawable 来 将 Bitmap 绘制到 Canvas。具体如何显示位图不是主要的问题，主要问题是如何获取 Canvas，参考 Snake 中的方法，做了个简单的例子 testView，提供给大家下载。

testView 例子介绍：其包含 2 个类 testActivity，testView；testActivity 继承与 Activity，testView 继承与 View。这个例子就是将 testView 直接作为 testActivity 的窗口，这样我们就可以直接在 testView 画图了。具体如何实现的，请大家参考 testActivity 的 onCreate()中的代码，以及 layout\main.xml 中的 设置。在 testView 的 onDraw()直接画图，结果在例子程序运行后就可以直接在界面上显示了。

4. 位图缩放

位图的缩放，在 Android SDK 中提供了 2 种方法：

- 将一个位图按照需求重画一遍，画后的位图就是我们需要的了，与位图的显示几乎一样：
  drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)
- 在原有位图的基础上，缩放原位图，创建一个新的位图：
  createBitmap(Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)

第 2 种方法一看就明白，对于第一种方法，举个简单的例子来说明：

```
int w = 320,h = 240;
String mstrTitle = “感受 Android 带给我们的新体验”;
Bitmap mbmpTest = Bitmap.createBitmap(w,h, Config.ARGB_8888);
Canvas canvasTemp = new Canvas(mbmpTest);
```

```
canvasTemp.drawColor(Color.WHITE);
Paint p = new Paint();
String familyName = “宋体”;
Typeface font = Typeface.create(familyName,Typeface.BOLD);
p.setColor(Color.RED);
p.setTypeface(font);
p.setTextSize(22);
canvasTemp.drawText(mstrTitle,0,100,p);
```

显示位图 mbmpTest，就会发现一张 320×240、白色背景、红色“宋体”文字的图片，如下：

oid 感受 Android 带给我们的新体验

这个例子没有位图缩放的任何操作？的确，但是这是我在想如何写个简单的位图缩放的小程序时，最先想到的。看完这个例子，我想你就应该明白如何实现位图的缩 放了。不要小瞧了这个例子，虽然与位图缩放关系不大，但是却可以让你理解位图缩放的本质：将原始位图按照需求显示出来，就创造了一张新的位图。

5. 位图旋转

位图的旋转，离不开 Matrix。Matrix 在线性代数中都学习过，Android SDK 提供了 Matrix 类，可以通过各种接口来设置矩阵。结合上面的例子程序，将位图缩放例子程序在显示位图的时候前，增加位图旋转功能，修改代码如下：

```
Matrix matrix = new Matrix();
//matrix.postScale(0.5f, 0.5f);
matrix.setRotate(90,120,130);
canvas.drawBitmap(mbmpTest, matrix, mPaint);
```

旋转后的位图显示如下：

除了这种方法之外，我们也可以在使用 Bitmap 提供的函数如下：

public static Bitmap createBitmap (Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)，在原有位图旋转的基础上，创建新位图。

总结说明

对位图的操作，结合 Android SDK 中的类，详细的介绍完了。最后还需要强调的是：这篇文章只是对 Android SDK 中代码阅读分析，它代替不了你阅读 Android SDK，深入的学习还是要仔细的阅读 Android SDK。

相关文章

- Android 画图学习总结（五）——Paint
- Android 画图学习总结（四）——Animation（下）
- Android 画图学习总结（四）——Animation（中）
- Android 画图学习总结（四）——Animation（上）
- Android 画图学习总结（三）——Drawable

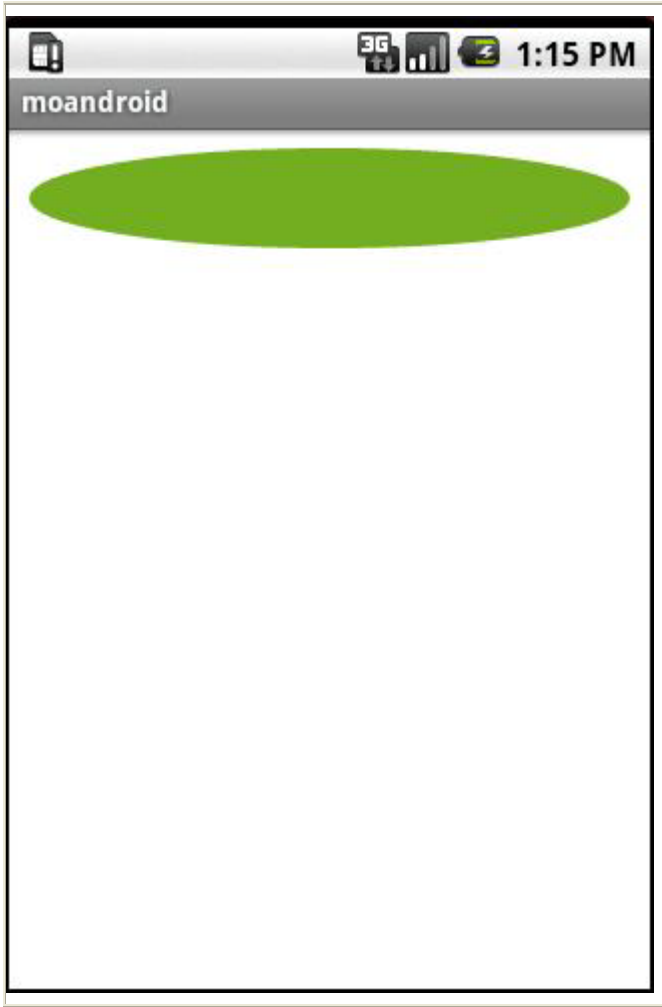
Android 画图学习总结（三）——Drawable

Android SDK 提供了一个强大的类 Drawable，Drawable 这个抽象类到底代表了什么，如何使用？Drawable 是个很抽象的概念，通过简单的例子程 序来学习它，理解它。先看个简单的例子，使用 Drawable 的子类 ShapeDrawable 来画图，如下：

```
public class testView extends View {
private ShapeDrawable mDrawable;
public testView(Context context) {
```

```
super(context);
int x = 10;
int y = 10;
int width = 300;
int height = 50;
mDrawable = new ShapeDrawable(new OvalShape());
mDrawable.getPaint().setColor(0xff74AC23);
mDrawable.setBounds(x, y, x + width, y + height);
}
protected void onDraw(Canvas canvas)
super.onDraw(canvas);
canvas.drawColor(Color.WHITE);//画白色背景
mDrawable.draw(canvas);
}
}
```

程序的运行结果，显示如下：



简要解析：

1. 创建一个 OvalShape（一个椭圆）；
2. 使用刚创建的 OvalShape 构造一个 ShapeDrawable 对象 mDrawable
3. 设置 mDrawable 的颜色；
4. 设置 mDrawable 的大小；
5. 将 mDrawable 画在 testView 的画布上；

这个简单的例子可以帮我们理解什么是 Drawable，Drawable 就是一个可画的对象，其可能是一张位图（BitmapDrawable）， 也可能是一个图形（ShapeDrawable），还有可能是一个图层（LayerDrawable），我们根据画图的需求，创建相应的可画对象，就可以 将这个可画对象当作一块“画布（Canvas）”，在其上面操作可画对象，并最终将这种可画对象显示在画布上，有点类似于“内存画布“。

上面只是一个简单的使用 Drawable 的例子，完全没有体现出 Drawable 的强大功能。Android SDK 中说明了 Drawable 主要的作用是：在 XML 中定义各种动画，然后把 XML 当作 Drawable 资源来读取，通过 Drawable 显示动画。下面举个使用 TransitionDrawable 的例子，创建一个 Android 工程，然后再这个工程的基础上修改，修改过程如下：
1、去掉 layout/main.xml 中的 TextView，增加 ImagView，如下：

```
<ImageView
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”
android:tint=”#55ff0000″
android:src=”@drawable/my_image”/>
```

2、创建一个 XML 文件，命名为 expand_collapse.xml，内容如下：

```
<?xml version=”1.0″ encoding=”UTF-8″?>
<transition xmlns:android=”http://schemas.android.com/apk/res/android”>
<item android:drawable=”@drawable/image_expand”/>
<item android:drawable=”@drawable/image_collapse”/>
</transition>
```

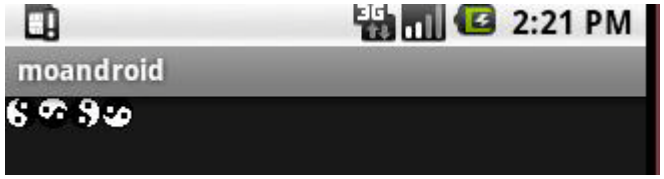
需要 3 张 png 图片，存放到 res\drawable 目录下，3 张图片分别命名为：my_image.png、image_expand.png、image_collapse.png。

3、修改 Activity 中的代码，内容如下：

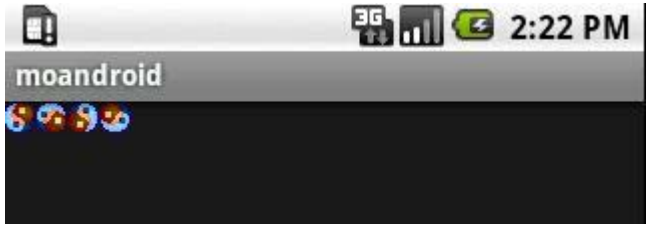
```
LinearLayout mLinearLayout;
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
```

```
    mLinearLayout = new LinearLayout(this);
    ImageView i = new ImageView(this);
    i.setAdjustViewBounds(true);
    i.setLayoutParams(new Gallery.LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
    mLinearLayout.addView(i);
    setContentView(mLinearLayout);
    Resources res = getResources();
    TransitionDrawable transition =
     (TransitionDrawable) res.getDrawable(R.drawable.expand_collapse);
    i.setImageDrawable(transition);
    transition.startTransition(10000);
}
```

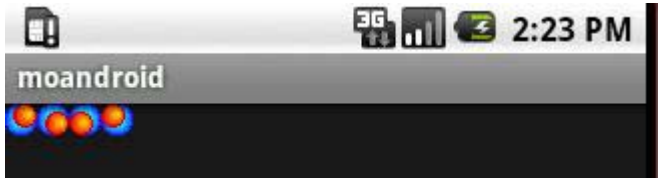
4、如果修改的没有错误，运行程序，结果显示如下：
初始图片



过渡中的图片



最后的图片



屏幕上动画显示的是： 从图片 image_expand.png 过渡到 image_collapse.png，也就是我们在 expand_collapse.xml 中定义的一个 transition 动画。看完这个例子，你对 Drawable 的理解是否又深入些？这里提供这个程序的源代码，供大家下载，可以在这个例子的基础上去体会其他的 Drawable，来加深对 Drawable 的理解。

总结说明

通过以上 2 个例子程序，相信对 Drawable 会有一定的认识了，在以后的篇幅中会介绍更多的例子，更加深入的学习和理解 Drawable。具体还有哪些 Drawable，大家到 Android SDK 去深入学习吧。

相关文章

- Android 画图学习总结（五）——Paint
- Android 画图学习总结（四）——Animation（下）
- Android 画图学习总结（四）——Animation（中）
- Android 画图学习总结（四）——Animation（上）
- Android 画图学习总结（二）——Bitmap

Android 画图学习总结（四）——Animation（上）

随着对 Drewable 的深入了解，发现了 Drawable 更加强大的功能：显示 Animation。Android SDK 介绍了 2 种 Animation：

- Tween Animation：通过对场景里的对象不断做图像变换(平移、缩放、旋转)产生动画效果
- Frame Animation：顺序播放事先做好的图像，跟电影类似

在使用 Animation 前，我们先学习如何定义 Animation，这对我们使用 Animation 会有很大的帮助。Animation 是以 XML 格式定义的，定义好的 XML 文件存放在 res\anim 中。由于 Tween Animation 与 Frame Animation 的定义、使用都有很大的差异，我们将分开介绍，本篇幅中介绍 Tween Animation 的定义与使用，后续篇幅再详细介绍 Frame Animation。按照 XML 文档的结构【父节点，子节点，属性】来介绍 Tween Animation，其由 4 种类型：

- 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：画面转换位置移动动画效果

在介绍以上 4 种类型前，先介绍 Tween Animation 共同的节点属性，关于节点的命名原则，请阅读 AndroidManifest.xml 文件结构说明。

表一		
属性[类型]	功能	
Duration[long]	属性为动画持续时间	时间以毫秒为单位
fillAfter [boolean]	当设置为 true ， 该动画转化在动画结束后被应用	
fillBefore[boolean]	当设置为 true ， 该动画转化在动画开始前被应用	

interpolator	指定一个动画的插入器	有一些常见的插入器 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
repeatCount[int]	动画的重复次数	
RepeatMode[int]	定义重复的行为	1：重新开始 2：plays backward
startOffset[long]	动画之间的时间间隔，从上次动画停多少时间开始执行下个动画	
zAdjustment[int]	定义动画的 Z Order 的改变	0：保持 Z Order 不变 1：保持在最上层 -1：保持在最下层

看了以上节点，大家是不是都想开始定义动画了。下面我们就开始结合具体的例子，介绍 4 种类型各自特有的节点元素。

表二		
XML 节点		功能说明
alpha		渐变透明度动画效果
<alpha android:fromAlpha=”0.1″ android:toAlpha=”1.0″ android:duration=”3000″ />		
fromAlpha	属性为动画起始时透明度	0.0 表示完全透明

toAlpha	属性为动画结束时透明度	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

表三

scale	渐变尺寸伸缩动画效果

```
<scale
android:interpolator= “@android:anim/accelerate_decelerate_interpolator”
android:fromXScale=”0.0″
android:toXScale=”1.4″
android:fromYScale=”0.0″
android:toYScale=”1.4″
android:pivotX=”50%”
android:pivotY=”50%”
android:fillAfter=”false”
android:startOffset=“700”
android:duration=”700″
android:repeatCount=”10″  />
```

fromXScale[float] fromYScale[float]	为动画起始时，X、Y 坐标上的伸缩尺寸	0.0 表示收缩到没有 1.0 表示正常无伸缩
toXScale [float] toYScale[float]	为动画结束时，X、Y 坐标上的伸缩尺寸	值小于 1.0 表示收缩 值大于 1.0 表示放大
pivotX[float] pivotY[float]	为动画相对于物件的 X、Y 坐标的开始位置	属性值说明：从 0%-100%中取值，50%为物件的 X 或 Y 方向坐标上的中点位置

表四

translate	画面转换位置移动动画效果

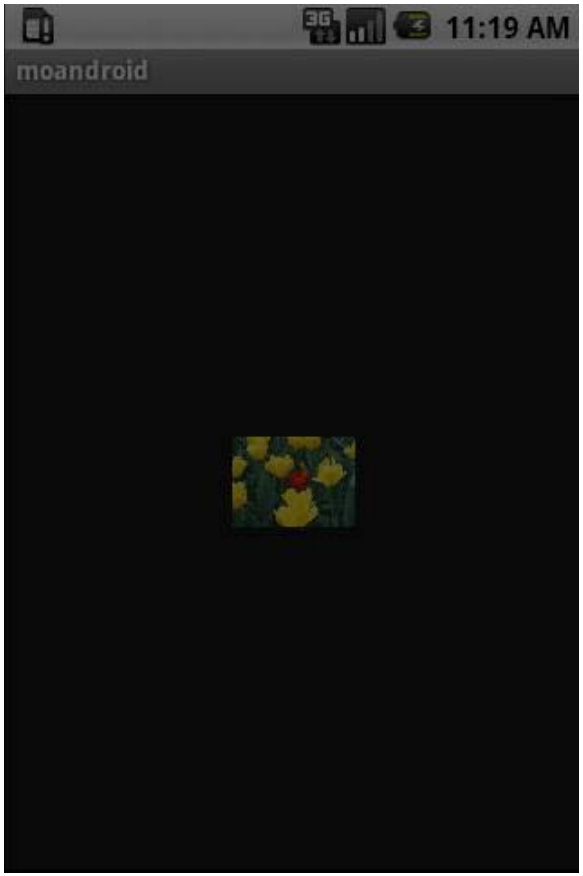
```
<translate
```

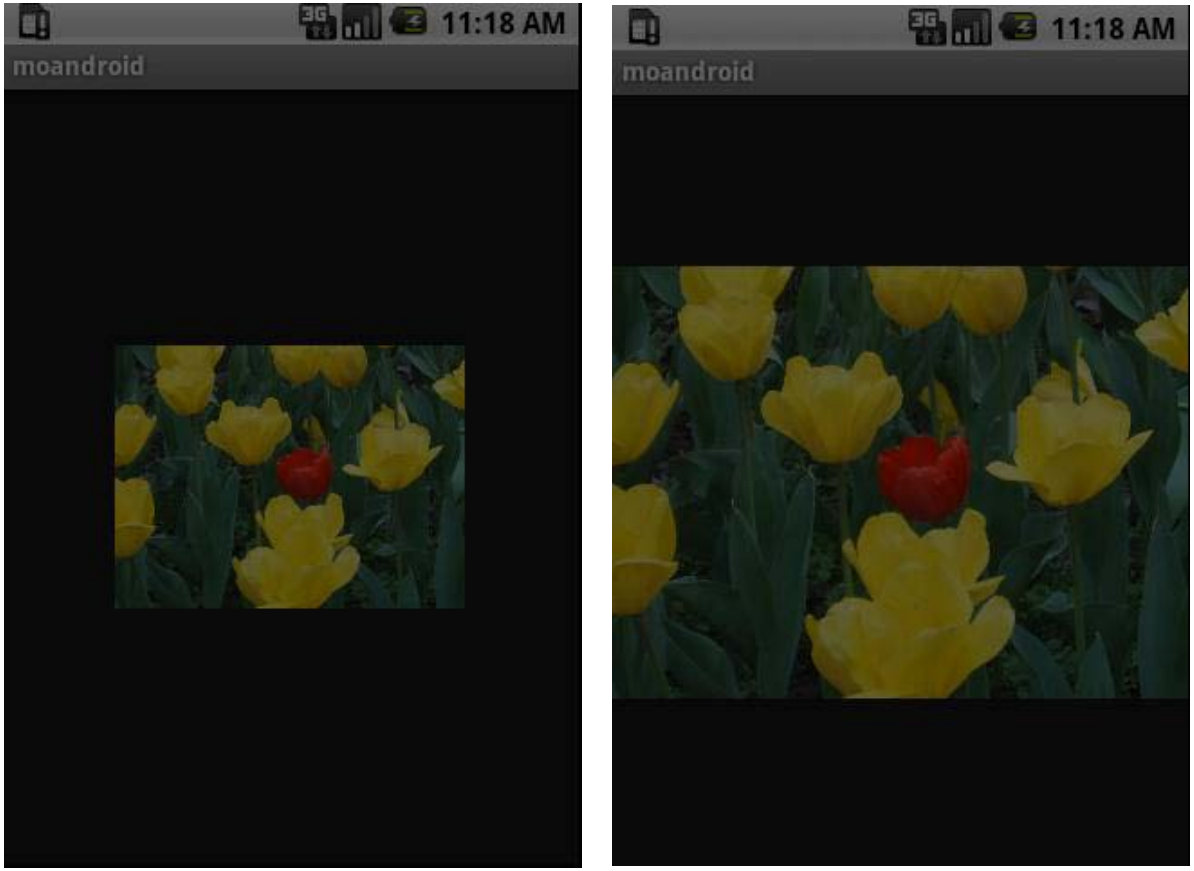
android:fromXDelta=”30″ android:toXDelta=”-80″ android:fromYDelta=”30″ android:toYDelta=”300″ android:duration=”2000″ />		
fromXDelta toXDelta	为动画、结束起始时 X 坐标上的位置	
fromYDelta toYDelta	为动画、结束起始时 Y 坐标上的位置	

表五		
rotate	画面转移旋转动画效果	
<rotate android:interpolator=”@android:anim/accelerate_decelerate_interpolator” android:fromDegrees=”0″ android:toDegrees=”+350″ android:pivotX=”50%” android:pivotY=”50%” android:duration=”3000″ />		
fromDegrees	为动画起始时物件的角度	说明
toDegrees	属性为动画结束时物件旋转的角度 可以大于 360 度	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 pivotY	为动画相对于物件的 X、Y 坐标的开始位	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

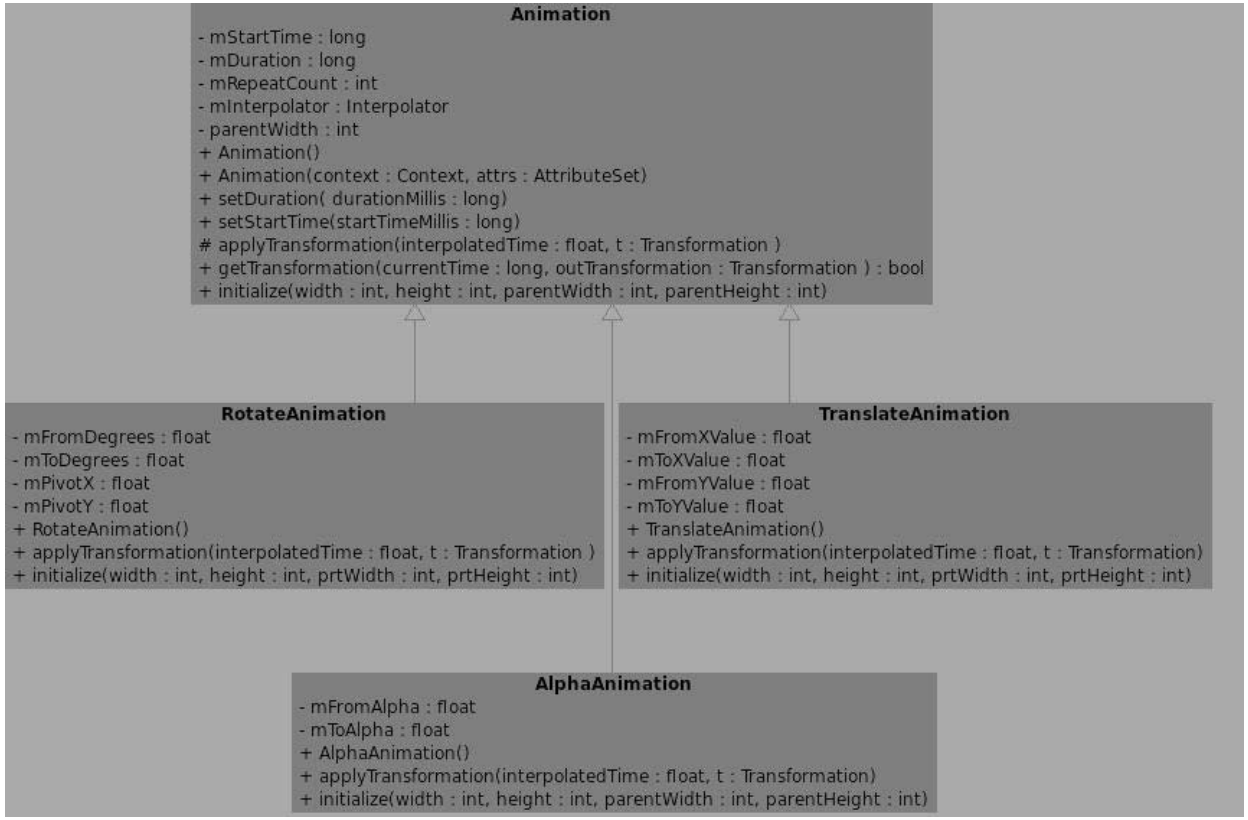
看了上面的例子，想必大家也想看下，这些定义的动画，运行起来是什么样效果，下面运行 Scale 例子中的动画，界面变化如下：





按照上面的讲述学习完了 Tween Animation 的定义,对 Tween Animation 有了详细的了解,再去了解下 Android SDK 的 animation package（android.view.animation）， 其提供了操作 Tween Animation 所有的类。

Android SDK 提供了基类：Animation，包含大量的 set/getXXXX()函数来设置、读取 Animation 的属性，也就是前面表一中显示的各种属 性。Tween Animation 由 4 种类型：alpha、scale、translate、roate，在 Android SDK 中提供了相应的类，Animation 类派生出了 AlphaAnimation、ScaleAnimation、 TranslateAnimation、RotateAnimation 分别实现了平移、旋转、改变 Alpha 值等动画，每个子类都在父类的基础上增加了各自独有的属性。再去看下这几个类的构造函数，是不是与我们在表二、表三、表四、表五种定义的属性完全一样。



在了解了 Tween Animation 的定义，对 android.view.animation 有了一些基本的认识后，开始介绍 Tween Animation 如何使用。Android SDK 提供了 2 种方法：直接从 XML 资源中读取 Animation，使用 Animation 子类的构造函数来初始化 Animation 对象，第二种方法在 看了 Android SDK 中各个类的说明就知道如何使用了，下面简要说明从 XML 资源中读取 Animation，按照应用程序开发的过程，介绍整个使用的过程，如下：

1. 创建 Android 工程；
2. 导入一张图片资源；
3. 将 res\layout\main.xml 中的 TextView 取代为 ImageView；
4. 在 res 下创建新的文件夹 anim，并在此文件夹下面定义 Animation XML 文件；
5. 修改 OnCreate()中的代码，显示动画资源；

关键代码，解析如下：

```
//main.xml 中的 ImageView
ImageView spaceshipImage = (ImageView) findViewById(R.id.spaceshipImage);
//加载动画
Animation hyperspaceJumpAnimation =
AnimationUtils.loadAnimation(this, R.anim.hyperspace_jump);
//使用 ImageView 显示动画
spaceshipImage.startAnimation(hyperspaceJumpAnimation);
```

工程的源代码提供给大家下载，下载地址，这里简要解析如下：

- AnimationUtils 提供了加载动画的函数，除了函数 loadAnimation()，其他的到 Android SDK 中去详细了解吧；
- 所谓的动画，也就是对 view 的内容做一次图形变换；

总结说明

看了这个长篇幅的介绍，详细大家对 Tween Animation 的定义、使用都有了比较深入的了解，由于篇幅有限，这里将 Android SDK 中的内容省略了不少，比如说：Interpolator，需要大家自己去 Android SDK 中仔细阅读。

相关文章

- Android 画图学习总结（四）——Animation（中）
- Android 画图学习总结（四）——Animation（下）
- Android 画图学习总结（五）——Paint
- Android 画图学习总结（三）——Drawable
- Android 画图学习总结（二）——Bitmap

Android 画图学习总结（四）——Animation（中）

在 Android 画图学习总结（四）——Animation（上）中详细介绍了 Tween Animation 的定义、使用，由于篇幅有限，很多中重要的方面没有说明，这篇文章一方面做个完整的总结说明，另外一方面补充说明上一篇幅遗漏的问题，帮助大家更好的理解 Tween Animation。

对 Tween Animation 的本质做个总结：Tween Animation 通过对 View 的内容完成一系列的图形变换（包括平移、缩放、旋转、改变透明度)来实现动画效果。具体来讲，预先定义一组指令，这些指令指定了图形变换的类型、触发时间、持续时间。这些指令可以是 以 XML 文件方式定义，也可以是以源代码方式定义。程序沿着时间线执行这些指令就可以实现动画效果。

在这里，我们需要对 2 个问题进行深入的解析：

- 动画的运行时如何控制的？
- 动画的运行模式。

动画的运行时如何控制的？

这个问题，我们也就也就是上一篇幅中提到的 Tween Animation，估计大家对什么是 Interpolator、到底有什么作用，还是一头雾水，在这里做个详细的说明。按照 Android SDK 中对 interpolator 的说明：interpolator 定义一个动画的变化率（the rate of change）。这使得基本的动画效果(alpha, scale, translate, rotate）得以加速，减速，重复等。

用通俗的一点的话理解就是：动画的进度使用 Interpolator 控制。Interpolator 定义了动画的变化速度，可以实现匀速、正加速、负加速、无规则变加速等。Interpolator 是基类，封装了所有 Interpolator 的共同方法，它只有一个方法，即 getInterpolation (float input)，该方法 maps a point on the timeline to a multiplier to be applied to the transformations of an animation。Android 提供了几个 Interpolator 子类，实现了不同的速度曲线，如下：

AccelerateDecelerateInterpolator	在动画开始与介绍的地方速率改变比较慢，在中间的时候加速
AccelerateInterpolator	在动画开始的地方速率改变比较慢，然后开始加速
CycleInterpolator	动画循环播放特定的次数，速率改变沿着正弦曲线
DecelerateInterpolator	在动画开始的地方速率改变比较慢，然后开始减速
LinearInterpolator	在动画的以均匀的速率改变

对于 LinearInterpolator ，变化率是个常数，即 f (x) = x.

```
public float getInterpolation(float input) {
return input;
}
```

Interpolator 其他的几个子类，也都是按照特定的算法，实现了对变化率。还可以定义自己的 Interpolator 子类，实现抛物线、自由落体等物理效果。

动画的运行模式

动画的运行模式有两种：

- 独占模式，即程序主线程进入一个循环，根据动画指令不断刷新屏幕，直到动画结束；
- 中断模式，即有单独一个线程对时间计数，每隔一定的时间向主线程发通知，主线程接到通知后更新屏幕；

额外补充说明：Transformation 类

Transformation 记录了仿射矩阵 Matrix，动画每触发一次，会对原来的矩阵做一次运算， View 的 Bitmap 与这个矩阵相乘就可实现相应的操作(旋转、平移、缩放等)。Transformation 类封装了矩阵和 alpha 值，它有两个重要的成员，一是 mMatrix，二是 mAlpha。Transformation 类图如下所示：

总结说明

图形变换通过仿射矩阵实现。图形变换是图形学中的基本知识，简单来说就是，每种变换都是一次矩阵运算。在 Android 中，Canvas 类中包含当前矩阵，当调用 Canvas.drawBitmap (bmp, x, y, Paint) 绘制时，A ndroid 会先把 bmp 做一次矩阵运算，然后将运算的结果显示在 Canvas 上。这样，编程人员只需不断修改 Canvas 的矩阵并刷新屏幕，View 里的对象就会不停的做图形变换，动画就形成了。

相关文章

- Android 画图学习总结（四）——Animation（上）
- Android 画图学习总结（四）——Animation（下）
- Android 画图学习总结（五）——Paint
- Android 画图学习总结（三）——Drawable
- Android 画图学习总结（二）——Bitmap

Android 画图学习总结（四）——Animation（下）

在 Android 画图学习总结（四）——Animation（上）中， 我们详细介绍了 Tween Aniamation，这里我们将介绍另外一种动画 Frame Animation。在前面已经说过，Frame Animation 是顺序播放事先做好的图像，跟电影类似。不同于 animation package， Android SDK 提供了另外一个类 AnimationDrawable 来定义、使用 Frame Animation。

Frame Animation 可以在 XML Resource 定义（还是存放到 res\anim 文件夹下），也可以使用 AnimationDrawable 中的 API 定义。由于 Tween Animation 与 Frame Animation 有着很大的不同，因此 XML 定义的格式也完全不一样，其格式是：首先是 animation-list 根节点，animation- list 根节点中包含多个 item 子节点，每个 item 节点定义一帧动画：当前帧的 drawable 资源和当前帧持续的时间。下面对节点的元素加以说明：

XML 属性	说明
drawable	当前帧引用的 drawable 资源
duration	当前帧显示的时间（毫秒为单位）
oneshot	如果为 true，表示动画只播放一次停止在最后一帧上，如果设置为 false 表示动画循环播放。
variablePadding	If true, allows the drawable's padding to change based on the current state that is selected.
visible	规定 drawable 的初始可见性，默认为 flase；

下面就给个具体的 XML 例子，来定义一帧一帧的动画：

```
<animation-list xmlns:android=”http://schemas.android.com/apk/res/android”
android:oneshot=”true”>
<item android:drawable=”@drawable/rocket_thrust1″ android:duration=”200″ />
<item android:drawable=”@drawable/rocket_thrust2″ android:duration=”200″ />
<item android:drawable=”@drawable/rocket_thrust3″ android:duration=”200″ />
</animation-list>
```

上面的 XML 就定义了一个 Frame Animation，其包含 3 帧动画，3 帧动画中分别应用了 drawable 中的 3 张图片：rocket_thrust1，rocket_thrust2，rocket_thrust3，每帧动画持续 200 毫秒。

然后我们将以上 XML 保存在 res/anim/文件夹下，命名为 rocket_thrust.xml，显示动画的代码，如下：在 OnCreate()中增加如下代码：

```
ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
rocketImage.setBackgroundResource(R.anim.rocket_thrust); rocketAnimation = (AnimationDrawable)
rocketImage.getBackground();
```

最后还需要增加启动动画的代码：

```
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        rocketAnimation.start();
        return true;
    }
    return super.onTouchEvent(event);
}
```

代码运行的结果想必大家应该就知道了（3 张图片按照顺序的播放一次），不过有一点需要强调的是：启动 Frame Animation 动画的代码 rocketAnimation.start();不能在 OnCreate()中，因为在 OnCreate()中 AnimationDrawable 还没有完全的与 ImageView 绑定，在 OnCreate()中启动动画，就只能看到第一张图片。

下面，阅读 Android SDK 中对 AnimationDrawable 的介绍，有个简单的了解：

AnimationDrawable	
获取、设置动画的属性	
int getDuration()	获取动画的时长
int getNumberOfFrames()	获取动画的帧数
boolean isOneShot() Void setOneShot(boolean oneshot)	获取 oneshot 属性 设置 oneshot 属性
void inflate(Resurce r,XmlPullParser p, AttributeSet attrs)	
增加、获取帧动画	
Drawable getFrame(int index)	获取某帧的 Drawable 资源
void addFrame(Drawable frame,int duration)	为当前动画增加帧（资源，持续时长）

动画控制	
void start()	开始动画
void run()	外界不能直接掉调用，使用 start()替代
boolean isRunning()	当前动画是否在运行
void stop()	停止当前动画

总结说明

Frame Animation 的定义、使用比较简单，在这里已经详细介绍完了，更加深入的学习还是到 Android SDK 去仔细了解吧，在 Android SDK 中也包含很多这方面的例子程序。

相关文章

- Android 画图学习总结（四）——Animation（中）
- Android 画图学习总结（四）——Animation（上）
- Android 画图学习总结（五）——Paint
- Android 画图学习总结（三）——Drawable
- Android 画图学习总结（二）——Bitmap

Android 画图学习总结（五）——Paint

前面的 Android 画图学习总结的系列中，我们分别学习了 Bitmap、Drawable、Aniamtion，除了这些画图元素之外，开发应用程序使用最多的还是 String（字符串），下面我们就如何显示 String 详细的说明。

引用 Android SDK 中显示 String 的函数，列举如下：

drawText(String text, int start, int end, float x, float y, Paint paint)	Draw the text, with origin at (x,y), using the specified paint.
void drawText(char[] text, int index, int count, float x, float y, Paint paint)	Draw the text, with origin at (x,y), using the specified paint.
void drawText(String text, float x, float y, Paint paint)	Draw the text, with origin at (x,y), using the specified paint.
void drawText(CharSequence text, int start, int end, float x, float y, Paint paint)	Draw the specified range of text, specified by start/end, with its origin at (x,y), in the specified Paint.
void drawTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint)	Draw the text, with origin at (x,y), using the specified paint, along the specified path.
void drawTextOnPath(char[] text, int index, int count, Path path, float hOffset, float vOffset, Paint paint)	Draw the text, with origin at (x,y), using the specified paint, along the specified path.

在所有的函数中，参数主要分为 3 部分：字符串（String、char、CharSequence），长度（start—end、index—count），如何显示 String（paint）。前 2 个参数一看就明白，这里我们主要介绍第 3 个参数 Paint paint。

首先，什么是 Paint？

引用 Android SDK 中的说明，**Paint 类包含样式和颜色有关如何绘制几何形状，文本和位图的信息**。Canvas 是一块画布，具体的文本和位图如何显示，这就是在 Paint 类中定义了。

然后，Paint 有哪些功能？

在了解 Paint 的功能前，我们按照 Word 文档的功能，说明下对 String 的显示，有影响的因素有哪些？字体、大小（TextSize）、颜色 （TextColor）、对齐方式(TextAlign)、粗体（Bold）、斜体（Italic）、下划线（Underline）等，下面我们就按照这 些影响 String 显示的因素，结合 Android SDK 中 Paint 类的介绍，详细说明 Paint 类有哪些功能。

在 Android SDK 中使用 Typeface 类来定义字体,Typeface 类:指定字体和字体的固有风格,用于 Paint,类似于 Paint 的其他 textSize，textSkewX，textScaleX 一样，来说明如何绘制文本。归纳起来，Typeface 类主要包括以下 3 个方面：

1. 一些常量的定义（BOLD，BOLD_ITALIC，ITALIC，NORMAL）
2. 常量字体的定义：

字体（Typeface）	说明
DEFAULT	The default NORMAL typeface object
DEFAULT_BOLD	The default BOLD typeface object.
MONOSPACE	The NORMAL style of the default monospace typeface.
SANS_SERIF	The NORMAL style of the default sans serif typeface.
SERIF	The NORMAL style of the default serif typeface.

3. 这些常量字体，在程序中是可以直接使用的，例如：Typeface. SERIF
4. 函数：创建字体(Create())，获取字体属性(getStyle()、isBold()、isItalic())；
   Typeface 类不仅定义了字体，还包括粗体（Bold）、斜体（Italic）。

其它对显示 String 有影响的因素，我们都可以在 Paint 类中找到它们的影子，如下：

类型	功能	Paint 中的相关操作
Typeface	字体	Typeface setTypeface(Typeface typeface) Typeface getTypeface()
class Paint.Align	对齐方式	setTextAlign(Paint.Align align) Paint.Align getTextAlign()
	字体大小	int getTextSize() setTextSize(float textSize)
	颜色	setColor(int color) int getColor()
	下划线	boolean isUnderlineText() setUnderlineText(boolean underlineText)

看了这些，想必大家对 Paint 类也有些基本的了解，实际上在 Paint 类中还有其他一些功能，比如说 Alpha、Dither 等，这些也只有大家去 Android SDK 中仔细阅读了，由于篇幅有限，就不在此详细说明。

最后，如何使用 Paint 显示 String?

实际上，在前面的一些篇幅中的例子程序中都使用了 Paint 类，在 Android 画图学习总结（二）——Bitmap 例子的基础上修改下，说明如何使用 Paint，如下：

```
public void onDraw(Canvas canvas)
{
    super.onDraw(canvas);
    Paint p = new Paint();
    String familyName = “宋体”;
    Typeface font = Typeface.create(familyName,Typeface.BOLD);
    p.setColor(Color.RED);
    p.setTypeface(font);
    p.setTextSize(22);
    canvas.drawText(mstrTitle,0,100,p);
}
```

程序运行后，界面显示如下：

总结说明

String 是我们开发应用程序最经常处理的数据，如何显示 String 是开发应用程序最基本的要求，在这里我只是抛砖引玉下，简要介绍了 Paint 类，更加深入的学习请大家到 Android SDK 中去详细阅读吧！

相关文章

- Android 画图学习总结（四）——Animation（下）
- Android 画图学习总结（四）——Animation（中）
- Android 画图学习总结（四）——Animation（上）
- Android 画图学习总结（三）——Drawable
- Android 画图学习总结（二）——Bitmap