[**Android有效解决加载大图片时内存溢出的问题**](http://yueguc.iteye.com/blog/1051966)

尽量不要使用setImageBitmap或setImageResource或BitmapFactory.decodeResource来设置一张大图，因为这些函数在完成decode后，最终都是通过java层的createBitmap来完成的，需要消耗更多内存。因此，改用先通过BitmapFactory.decodeStream方法，创建出一个bitmap，再将其设为ImageView的 source，decodeStream最大的秘密在于其直接调用JNI>>nativeDecodeAsset()来完成decode，无需再使用java层的createBitmap，从而节省了java层的空间。

如果在读取时加上图片的Config参数，可以跟有效减少加载的内存，从而跟有效阻止抛out of Memory异常。另外，decodeStream直接拿的图片来读取字节码了， 不会根据机器的各种分辨率来自动适应， 使用了decodeStream之后，需要在hdpi和mdpi，ldpi中配置相应的图片资源，否则在不同分辨率机器上都是同样大小（像素点数量），显示出来的大小就不对了。

另外，以下方式也大有帮助：

1. InputStream is=

this.getResources().openRawResource(R.drawable.pic1);

BitmapFactory.Options options=new BitmapFactory.Options();

options.inJustDecodeBounds = false;

options.inSampleSize = 10; //width，hight设为原来的十分一

Bitmap btp =BitmapFactory.decodeStream(is,null,options);

2. if(!bmp.isRecycle() ){

bmp.recycle() //回收图片所占的内存

system.gc() //提醒系统及时回收

}

以下奉上一个方法：

Java代码

/\*\*

\* 以最省内存的方式读取本地资源的图片

\* @param context

\* @param resId

\* @return

\*/

public static Bitmap readBitMap(Context context, int resId){

BitmapFactory.Options opt = new BitmapFactory.Options();

opt.inPreferredConfig = Bitmap.Config.RGB\_565;

opt.inPurgeable = true;

opt.inInputShareable = true;

//获取资源图片

InputStream is =

context.getResources().openRawResource(resId);

return BitmapFactory.decodeStream(is,null,opt);

}

=================================================================

***Android内存溢出的解决办法***

昨天在模拟器上给gallery放入图片的时候，出现java.lang.OutOfMemoryError: bitmap size exceeds VM budget 异常，图像大小超过了RAM内存。

模拟器RAM比较小，只有8M内存，当我放入的大量的图片（每个100多K左右），就出现上面的原因。

由于每张图片先前是压缩的情况，放入到Bitmap的时候，大小会变大，导致超出RAM内存，具体解决办法如下：

//解决加载图片 内存溢出的问题

//Options 只保存图片尺寸大小，不保存图片到内存

BitmapFactory.Options opts = new BitmapFactory.Options();

//缩放的比例，缩放是很难按准备的比例进行缩放的，其值表明缩放的倍数，

//SDK中建议其值是2的指数值,值越大会导致图片不清晰

opts.inSampleSize = 4;

Bitmap bmp = null;

bmp = BitmapFactory.decodeResource(getResources(),

mImageIds[position],

opts);

...

bmp.recycle();//回收

通过上面的方式解决了，*但是这并不是最完美*的解决方式。

通过一些了解，得知如下：

优化Dalvik虚拟机的堆内存分配

对 于Android平台来说，其托管层使用的Dalvik Java VM从目前的表现来看还有很多地方可以优化处理，比如我们在开发一些大型游戏或耗资源的应用中可能考虑手动干涉GC处理，使用 dalvik.system.VMRuntime类提供的setTargetHeapUtilization方法可以增强程序堆内存的处理效率。当然具体 原理我们可以参考开源工程，这里我们仅说下使用方法:

private final static float TARGET\_HEAP\_UTILIZATION = 0.75f;

在程序onCreate时就可以调用

VMRuntime.getRuntime().setTargetHeapUtilization(TARGET\_HEAP\_UTILIZATION); 即可。

Android堆内存也可自己定义大小

对于一些Android项目，影响性能瓶颈的主要是Android自己内存管理机制问题，目前手机厂商对RAM都比较吝啬，对于软件的流畅性来说RAM对 性能的影响十分敏感，除了 优化Dalvik虚拟机的堆内存分配外，我们还可以强制定义自己软件的对内存大小，我们使用Dalvik提供的 dalvik.system.VMRuntime类来设置最小堆内存为例:

private final static int CWJ\_HEAP\_SIZE = 6\* 1024\* 1024 ;

//设置最小heap内存为6MB大小

VMRuntime.getRuntime().setMinimumHeapSize(CWJ\_HEAP\_SIZE);。

当然对于内存吃紧来说还可以通过手动干涉GC去处理

bitmap 设置图片尺寸，避免 内存溢出 OutOfMemoryError的优化方法

★android 中用bitmap 时很容易内存溢出，报如下错误：Java.lang.OutOfMemoryError : bitmap size exceeds VM budget

● 主要是加上这段：

BitmapFactory.Options options = new BitmapFactory.Options();

options.inSampleSize = 2;

● eg1：(通过Uri取图片)

private ImageView preview;

BitmapFactory.Options options = new BitmapFactory.Options();

options.inSampleSize = 2;//图片宽高都为原来的二分之一，即图片为原来的四分之一

Bitmap bitmap = BitmapFactory.decodeStream(cr

.openInputStream(uri), null, options);

preview.setImageBitmap(bitmap);

以上代码可以优化内存溢出，但它只是改变图片大小，并不能彻底解决内存溢出。

eg2:（通过路径去图片）

private ImageView preview;

private String fileName= "/sdcard/DCIM/Camera/2010-05-14 16.01.44.jpg";

BitmapFactory.Options options = new BitmapFactory.Options();

options.inSampleSize = 2;//图片宽高都为原来的二分之一，即图片为原来的四分之一

Bitmap b = BitmapFactory.decodeFile(fileName, options);

preview.setImageBitmap(b);

filePath.setText(fileName);

★Android 还有一些性能优化的方法：

首先内存方面，可以参考 Android堆内存也可自己定义大小 和 优化Dalvik虚拟机的堆内存分配

基础类型上，因为Java没有实际的指针，在敏感运算方面还是要借助NDK来完成。Android123提示游戏开发者，这点比较有意思的是Google 推出NDK可能是帮助游戏开发人员，比如OpenGL ES的支持有明显的改观，本地代码操作图形界面是很必要的。

图形对象优化，这里要说的是Android上的Bitmap对象销毁，可以借助recycle()方法显示让GC回收一个Bitmap对象，通常对一个不用的Bitmap可以使用下面的方式，如

if(bitmapObject.isRecycled()==false) //如果没有回收

bitmapObject.recycle();

目前系统对动画支持比较弱智对于常规应用的补间过渡效果可以，但是对于游戏而言一般的美工可能习惯了GIF方式的统一处理，目前Android系统仅能预览GIF的第一帧，可以借助J2ME中通过线程和自己写解析器的方式来读取GIF89格式的资源。

对于大多数Android手机没有过多的物理按键可能我们需要想象下了做好手势识别 GestureDetector 和重力感应来实现操控。通常我们还要考虑误操作问题的降噪处理。

介绍一下图片占用进程的内存算法吧。

android中处理图片的基础类是Bitmap，顾名思义，就是位图。占用内存的算法如下：

图片的width\*height\*Config。

如果Config设置为ARGB\_8888，那么上面的Config就是4。一张480\*320的图片占用的内存就是480\*320\*4 byte。

前面有人说了一下8M的概念，其实是在默认情况下android进程的内存占用量为16M，因为Bitmap他除了java中持有数据外，底层C++的 skia图形库还会持有一个SKBitmap对象，因此一般图片占用内存推荐大小应该不超过8M。这个可以调整，编译源代码时可以设置参数。