Eclipse->Android Studio项目迁移

[1.Cocos2dx v3.2版本项目迁移](#h.xl8p01mnyt32)

[a.检出工程模板，配置环境](#h.akreozb9zsey)

[b.创建工程](#h.53ws38j2fqmf)

[c.复制需要的Android Studio工程](#h.dr8qyyz6wn6b)

[d.调整代码](#h.u1gl8pk0o0vn)

[2.Cocos2dx v2.2.6版本项目迁移](#h.7q82lufncazf)

[a.检出工程模板，配置环境](#h.30unetyyqsvj)

[b.创建工程](#h.6co06qhlgd4w)

[c.复制需要的Android Studio工程](#h.asyvdema507)

[d.复制脚本文件与调整代码](#h.hywrigluqit9)

[3.纯Android项目迁移](#h.9apvxpwwqws4)

[a.更新项目源码](#h.e2mdup8740v5)

[b.转换Eclipse工程为Android Studio工程](#h.n3xgfhjg1lnq)

[c.调整代码](#h.br4724v6cxz7)

[d.重新关联公共库](#h.aciepm6yetnz)

[4.常见问题](#h.tnkva6o2egnf)

[1.报Gradle语法错误，常如下图所示：](#h.esfkk59j4dux)

[2.报找不到aar包的情况，如下图：](#h.dvlgr2d6j9b8)

[3.运行程序，编译时间很长，且最后报错](#h.ftg76dczgu0)

[4.运行程序，加载.so文件报错](#h.8inoe59pzr6n)

[5.报AndroidManifest.xml里一些属性多重定义的错](#h.xu11zbhlyjg0)

[6.Android 65K问题](#h.n2kz6blfh5sv)

[7.程序打包APK失败](#h.tdh1vgrcwkbm)

# 1.Cocos2dx v3.2版本项目迁移

## a.检出工程模板，配置环境

首先检出包含模板工程的引擎到本地目录。

地址：[ssh://git@stash.stm.com:7999/cmf/cocos2d-x.git](mailto:git@stash.stm.com)

该地址下的引擎中，创建模板的脚本，模板工程均有修改，创建工程中的源码部分也是当前源码维护的最新版本。

把引擎克隆到本地后，修改环境变量，令COCOS\_CONSOLE\_ROOT的路径为模板引擎的路径：xxxxx/cocos2d-x-template/tools/cocos2d-console/bin.

切换模板分支为bugfix/Cocos2d-x-3.2-bugfix（用3.2版本的最新tag也可）

## b.创建工程

使用待迁移项目的工程名和包名创建工程

cocos new projName -p projPkg -l cpp

## c.复制需要的Android Studio工程

复制新创建工程中的proj.as\_google文件夹至待迁移工程的根目录下（和之前的proj.android,proj.ios\_mac同级）

## d.调整代码

打开Android Studio工程：

使用脚本关联所需要的公共库包，脚本在在proj.as\_google/工程名/libs目录下执行（没有libs文件夹就自己创建）

脚本地址：[ssh://git@stash.stm.com:7999/cmt/gitsubmodule.git](mailto:git@stash.stm.com) Android Studio分支

公共库的关联方式请详细参考脚本ReadMe文档或工程App Module下的build.gradle文件注释。

根据脚本的文档添加需要关联的三方库（有些三方库并不是由SSC提供的aar包）

在proj.as\_google/工程名/build.gradle文件中修改项目的版本号；

在proj.as\_google/工程名/AndroidManifest.xml文件中修改项目的广告ID等；

mipmap-xxxxx几个文件夹下放入工程icon，尺寸略有不同；

mipmap-hdpi目录下放入自己项目的moregame图片资源；

复制原eclipse项目中的jni代码至proj.as\_google/工程名/目录下

Android.mk文件根据项目需要自行调整，在Android Studio中自带的终端里，可以直接执行./build\_native.py编译C++代码。

模板创建的AppActivity.java文件能满足绝大部分需求，自己项目如有特殊定制，参照原项目修改即可，有额外的java代码需要自己复制过来。

# 2.Cocos2dx v2.2.6版本项目迁移

## a.检出工程模板，配置环境

首先检出包含模板工程的引擎到本地目录。

地址：[ssh://git@stash.stm.com:7999/cmf/cocos2d-x.git](mailto:git@stash.stm.com)

该地址下的引擎中，创建模板的脚本，模板工程均有修改，创建工程中的源码部分也是当前源码维护的最新版本。

把引擎克隆到本地后，修改环境变量，令COCOS\_CONSOLE\_ROOT的路径为模板引擎的路径：xxxxx/cocos2d-x-template/tools/cocos2d-console/bin.

切换模板分支为bugfix/Cocos2d-x-3.2-bugfix（用3.2版本的最新tag也可）

## b.创建工程

使用待迁移项目的工程名和包名创建工程

cocos new projName -p projPkg -l cpp

## c.复制需要的Android Studio工程

复制新创建工程中的proj.as\_google文件夹至待迁移工程的根目录下（和之前的proj.android同级）

## d.复制脚本文件与调整代码

打开Android Studio工程：

使用脚本关联所需要的公共库包，脚本在在proj.as\_google/工程名/libs目录下执行（没有libs文件夹就自己创建）

脚本地址：[ssh://git@stash.stm.com:7999/cmt/gitsubmodule.git](mailto:git@stash.stm.com) Android Studio分支

公共库的关联方式请详细参考脚本ReadMe文档或工程App Module下的build.gradle文件注释。

根据脚本的文档添加需要关联的三方库（有些三方库并不是由SSC提供的aar包）

在proj.as\_google/工程名/build.gradle文件中修改项目的版本号；

在proj.as\_google/工程名/AndroidManifest.xml文件中修改项目的广告ID等；

mipmap-xxxxx几个文件夹下放入工程icon，尺寸略有不同；

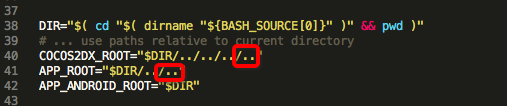
mipmap-hdpi目录下放入自己项目的moregame图片资源；

复制原eclipse项目中的jni代码至proj.as\_google/工程名/目录下

模板创建的AppActivity.java文件能满足绝大部分需求，自己项目如有特殊定制，参照原项目修改即可，有额外的java代码需要自己复制过来。**(特别注意：2dx v2.x的项目，.so文件是在自己的Activity里加载的！请迁移项目时不要忘了手动加载.so文件!)**

Android.mk文件根据项目需要自行调整；

复制原Eclipse工程下的build\_native.sh文件至proj.as\_google/工程名/目录下，打开build\_native.sh文件，修改下图两处，文件目录多添加一级，红框圈住的是已添加的脚本。



v2.2.6项目的build\_native.sh文件可通用，修改一份，其他项目都可以用，在v2.2.6版本的项目中我们使用build\_native.sh编译C++代码而非模板中带的build\_native.py(可删除)。

v2.2.6项目的迁移和v3.2大致相同，特别注意下上方红字部分即可。

# 3.纯Android项目迁移

## a.更新项目源码

在Eclipse的工作环境下，关联最新的广告公共库，根据更新要求，完成项目的内容更新，

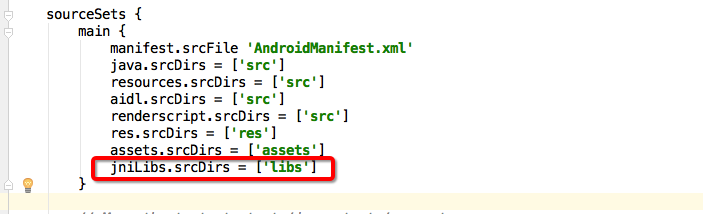
以前用到的框架，现在复制到项目中使用，不再通过关联的形式，复制框架的过程中，注意框架中原有资源与项目资源的合并。

## b.转换Eclipse工程为Android Studio工程

在项目更新完成后，取消项目的公共库关联，转换项目工程为Android Studio工程，因为我们之前所关联的公共库，在Android Studio中都将以aar包，jar包等形式关联。（不取消关联转换项目，会把关联工程一起转换，而且转换后的脚本文件位置也不在项目中（具体原因参见Android Studio使用入门文档或视频））。

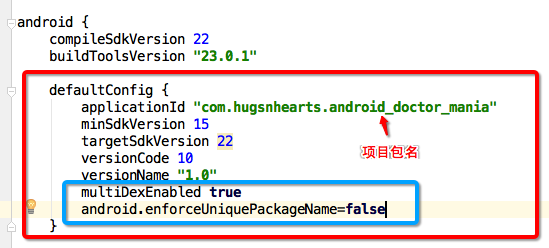
## c.调整代码

打开转换后的Android Stduio工程，因为gradle版本的原因，会报很多错，我们按照Android Studio的提示fix就好。



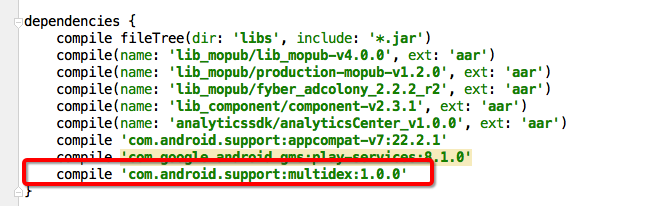
默认转换的Android Studio工程中，项目的build.gradle脚本文件没有上图红框圈住的那句代码，需要我们手动添加上（未添加会导致程序运行找不到libs下的.so文件）。

调整AndroidManifest.xml文件，把原来项目的versionCode，versionName ,minSdkVersion,targetSdkVersion等信息从AndroidManifest.xml文件中删除，移至build.gradle文件中，如下图：



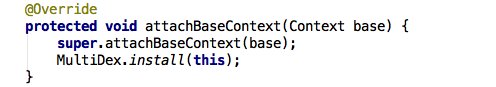
红框圈住的是我们需要添加在build.gradle文件中的信息，蓝框圈住的是解决Android 65k问题的代码，必须要。

为了解决Android 65k问题，我们的Application需要继承一个MultiDexApplication,这个Application来自包 **com.android.support:multidex，**所以，我们还需要在build.gradle中的dependencies里添加上该包的引用，如下图

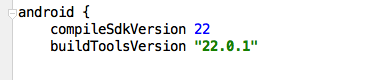


图c-1

添加了包引用，自己的Application类继承了个MultiDexApplication类，还需要覆写个MultiDexApplication中的一个方法如下图：



还需要注意的一点是buildTool的版本要用22的，如下图：



## d.重新关联公共库

重新关联公共库，关联后在build.gradle中的dependencies里如图c-1所示。（图中公共库使用版本基于文档编写时（2015-11-12））

再重新关联公共库后，原AndroidManifest.xml文件里配置的广告Activity全部删除，那些Activity已经配置在广告的aar包里。

此处再强调一次Android Studio中的公共库关联形式。

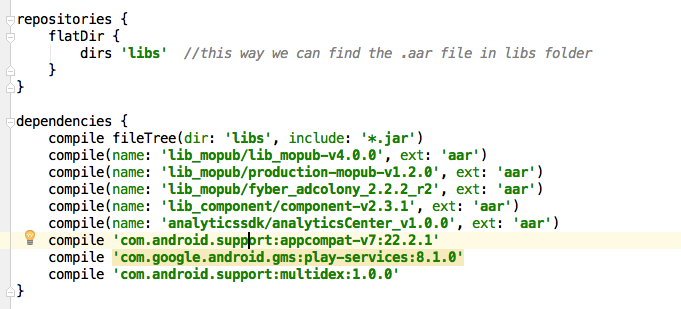
公共库关联的版本以脚本文档为准：

ssh://git@stash.stm.com:7999/cmt/gitsubmodule.git 【android-studio-xxx分支】

请特别注意三方库的几种关联方式：

1.由公司提供的jar包，从stash上检在libs目录下后，可以在Module Setting->dependencies里直接添加file dependency,找到对应目录的jar包进行添加；

2.由公司提供的aar包，从stash上检在libs目录下后，按如下格式添加aar包：



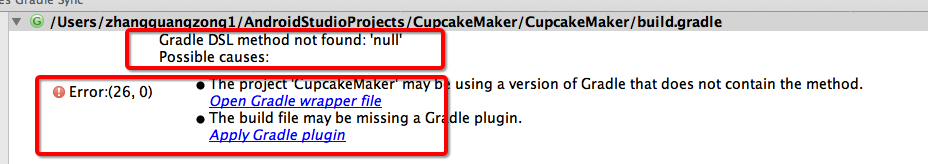
图中顶部的repositories｛……｝是不能少的，图中dependencies中有jar包，有aar包，也有来自中央仓库的包。

arr包格式：compile(name: 'aar包名(不带后缀)', ext: 'aar')

3.由中央仓库提供的三方包，原则上应该在Module Setting->dependencies里选libaray dependency,然后选择对应包，但是由于此种情况下选择的包和当前项目的buildTool版本相关，实际在使用过程中我们需要到的包版本可能不一样,所以按脚本文档里写的，进行手动写代码添加.如上图中倒数二三排添加googleServices和appcompat包的脚本代码，对于第三种包，具体应该使用哪个版本，用哪种方式添加，以公共库关联脚本文档为准！

# 4.常见问题

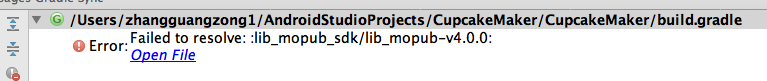
## 1.报Gradle语法错误，常如下图所示：



原因：目前出现的此种问题基本是编码格式引起的（从RTX复制相关代码来使用）。

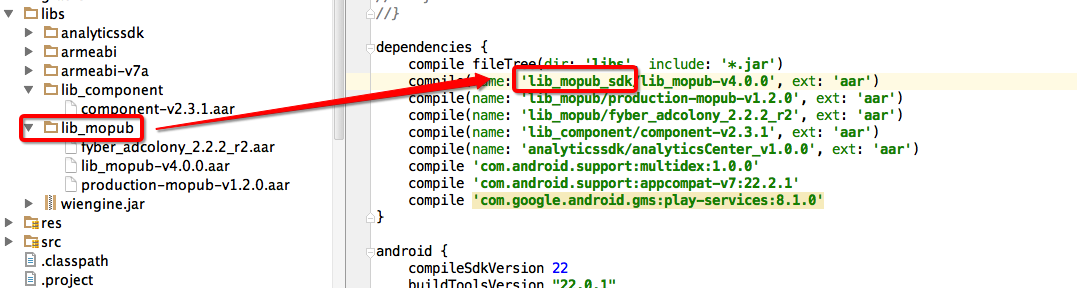
解决：点击下方的Apply Gradle plugin，上方窗口会跳到有问题的脚本代码处，建议把有问题的代码全部删除，重新手打一遍。如果是jar包，可以通过Module Setting重新关联一次。也可以把空格，换行符都删除，自己手动重新换行。

## 2.报找不到aar包的情况，如下图：



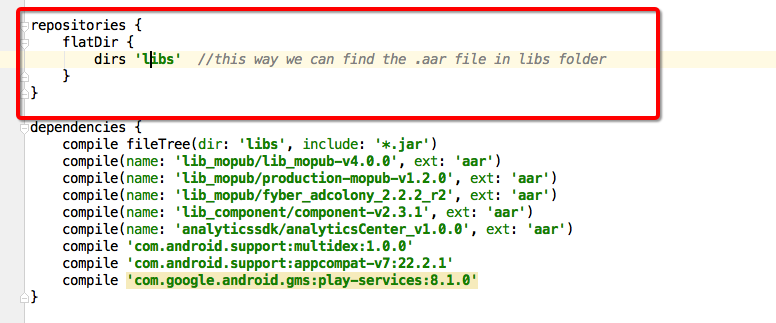
原因一般有两种：

一是因为路径不对，如下图：



修改路径即可。

二是因为少了部分脚本代码，如下图，少了红框圈住的代码：

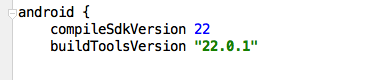


对于第二种情况，会导致该模块下的所有arr包都找不到，对于没用模板创建的项目，需要注意这一点。

## 3.运行程序，编译时间很长，且最后报错

一般报的错是OutOfMemeroy。

原因及解决办法：buildTool使用版本不对，改为22版本，如下图：



如果在修改了buildTool版本后，编译后还是报OutOfMemeroy的错，请检测下当前dependencies里使用的“play-services”版本，如果是8.1.0，请改成8.3.0再试试。

如果以上都不行，最后办法：在android{}内添加如下脚本

android{

…….

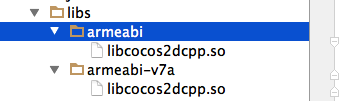
|  |
| --- |
| dexOptions {  incremental true  javaMaxHeapSize "4g"  }  …….  } |

## 4.运行程序，加载.so文件报错

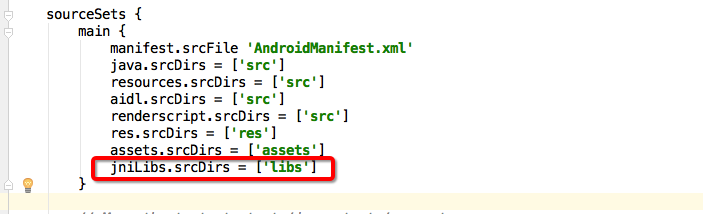
这没有截图，大家意会一下，程序一启动就会崩，看错误日志，要么说是不能加载xxx.so文件，或者说是不安全链接。

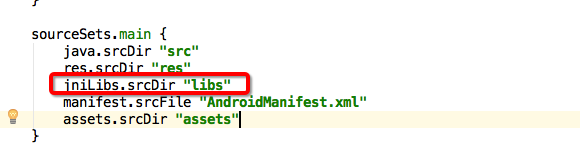
遇到此种情况，在确认.so文件确实成功编译后的前提下，检查两个地方：

1.是否有两个版本的.so文件，如下图，在模板的Application.mk里是有写明会编译出两个版本的，如果未使用模板，自行编译的，编译成功后，复制一份改名为另外一个版本，也是可以的。



2.当libs目录和src，res，assets等目录都在app根目录下时（同模块的build.gradle一级），脚本文件里未注明libs目录的路径，即缺少下图红框圈住的代码：

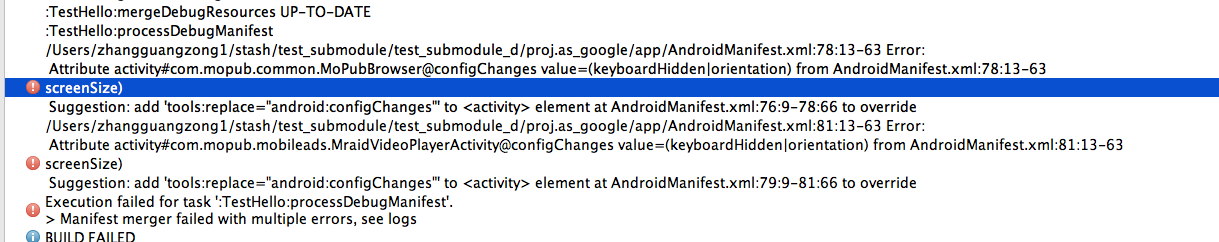




上面两个图的写法不一样，是因为gradle的版本原因，具体看你默认其他配置怎么写的。用模板创建的项目没有这个问题，常见于通过Eclipse工程转换过来的Android Studio项目，默认配置中会缺少libs目录的指定。

## 5.报AndroidManifest.xml里一些属性多重定义的错

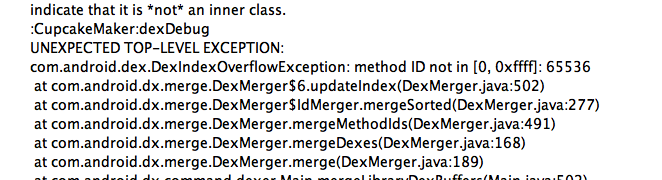
如下图：



原因及解决办法：Androd Studio中使用的广告公共库aar包已经配置了广告的Activity，在我们自己的项目AndroidManifest.xml里不用再配置了，删除自己项目里配置的广告Activity即可。模板创建的项目没有此问题，常见于通过Eclipse工程转换过来的Android Studio项目。

## 6.Android 65K问题

如下图：



解决方法：

1.在build.gradle中的dependencies内添加包multidex的引用，如下：

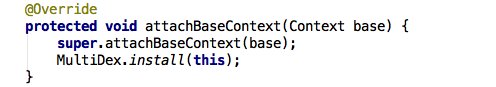
compile **'com.android.support:multidex:1.0.0'**

2.在build.gradle中defaultConfig里加如下两句代码：

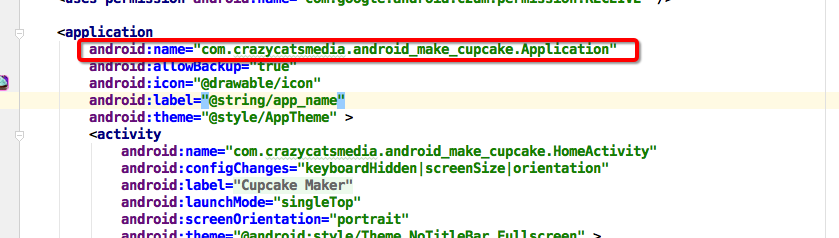
multiDexEnabled **true**

android.enforceUniquePackageName=**false**

3.让自己项目中的Application类(没有自己写个类)继承自MultiDexApplication，并且覆写如下方法：



需要注意的是，Android程序中如果自己写了Application类，需要在AndroidManifest.xml里写明类的地址，如下图：



模板创建的项目没有此问题(对应类和代码，模板已添加)，常见于原Eclipse工程转换过来的Android Studio项目。

## 7.程序打包APK失败

请注意检查项目存放路径，最好不要带空格或点（.）之类的特殊字符，不要以数字打头的路径。

实际项目根目录下的proj.as\_google路径中已经带了点，但是目前有项目因为存在如下路径而导致APK打包失败：

projects/3.x/projectName/proj.as\_google;

解决办法是项目移至3.x路径之外即可成功打包，经验证导致打包失败是因为路径中3.x是以数字打头造成的。