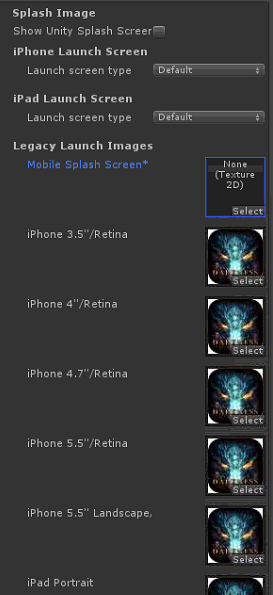
1. **将项目copy到本地。**

svn checkout <http://192.168.11.241/svn/xgame/qjzr/src/client/branch/0.5.0/DevProject> -–username svn账号

1. **Unity**

使用Unity打开项目，删除Assets/Plugins/x86/，Assets/Plugins/x86\_64/文件夹里面的内容。文件夹不删除。打开File/Build Settings， 切换到iOS环境。接着选择Player Settings。检查包名，游戏名是否设置正确；查看PlayerSettings中的Icon栏是否设置正确。查看Splash Image是否设置正确：



最后是Other Settings栏，跟邓总确认Auto Graphics API的设置参数。修改Bundle Identifier，Version，Build版本参数。

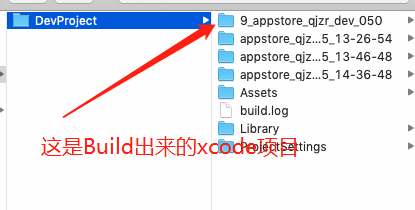
在Assets/Resources/目录下有两个文件需要特殊注意：

system\_config\_ios.xml，其中<cfg url = “”/>配置了生成的ipa连接的服务器cfg文件。发现打出来的包连不上需要跟项目负责人确认正确的cfg路径。

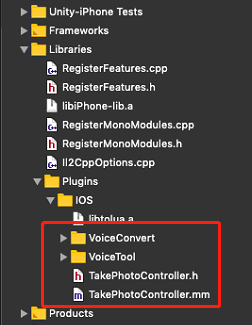
VersionInfo.xml， ResourceVersion，不能小于服务器端补丁的版本，一般查看svn export文件夹下最高的版本，不低于这个版本就可以。GameLoaderVersion，如果没有修改过GameLoader这里可以不改，但是如果改过GameLoader则必须保证这里的版本比服务端的高，否则会下载旧版本覆盖。

这两个文件检查没有问题，则打开File/Build Settings/，点击Build，生成xcode项目，将项目保存在DevProject目录下。

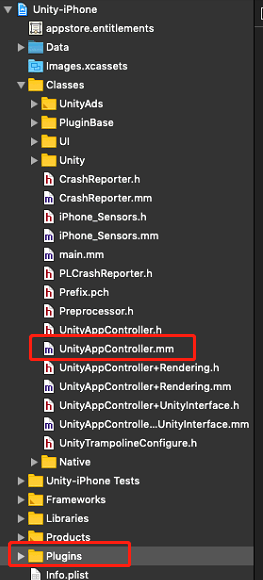
1. **Xcode**



打开xcode项目，删除Libraries/Plugins/IOS下的VoiceConvert，VoiceTool，TakePhotoController.h，TakePhotoController.mm文件。选中右键 >delete >Move To Trash



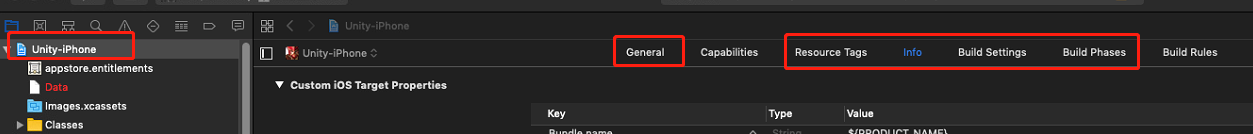
添加SDK文件支持，首先将模板的UnityAppController.mm替换xcode工程的Classes/UnityAppController.mm，将模板的Plugins文件夹整个添加到xcode目录下：说明，这两个东西是接好了SDK的直接复制内容进去即可，如果运营方有新增的内容或者更新sdk，将新增的SDK文件替换到Plugins文件夹中，旧的删除，然后再Plugins文件夹下的appstoreqjzr.mm文件中添加新接口。暴露给C#使用。接口写在extern “C”内部。如果SDK没有改动，这两份东西也不需要改动。



修改Info.plist文件，这一份也是直接将模板的Info.plist文件复制过去即可，很多权限设置运营方都已经做了要求，也不会再改动了，除非运营方需要新增其他东西。

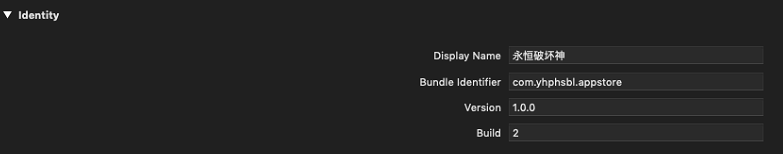
1. **配置**

选择项目Unity-iPhone，下面红框内都有需要配置的地方：

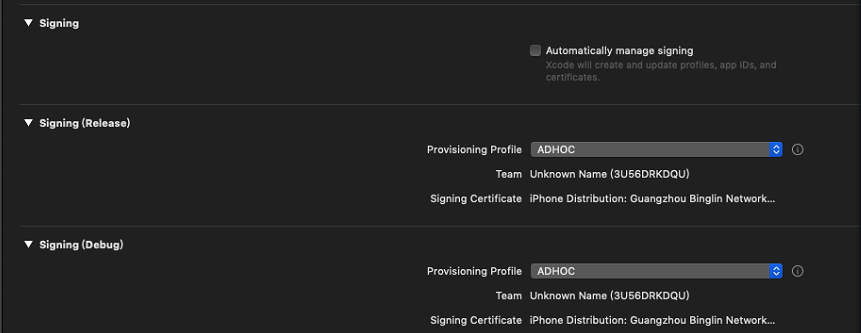


**General：**

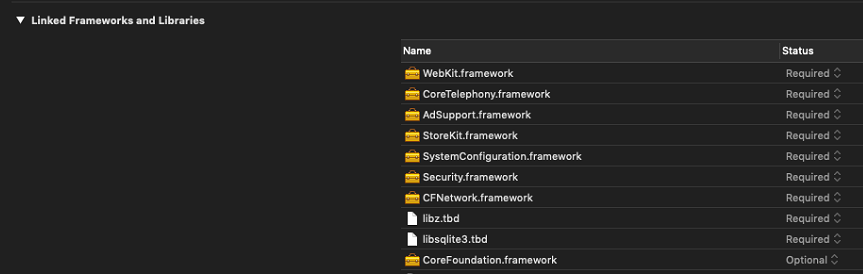
包名，版本号根据运营方给的填：



签名，运营想要什么包就给对应的签名（有dev（开发），dis（正式），adhoc(兼容性测试)三种。

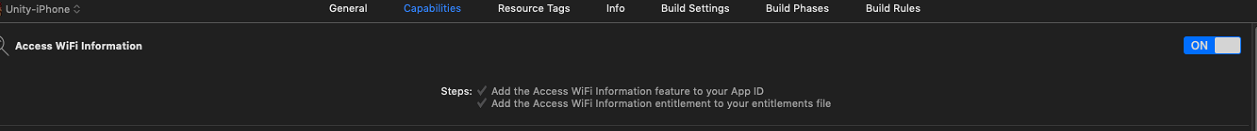


添加依赖包，添加哪些包看SDK对接文档：



**Capabilities:**

打开Access WiFi Information



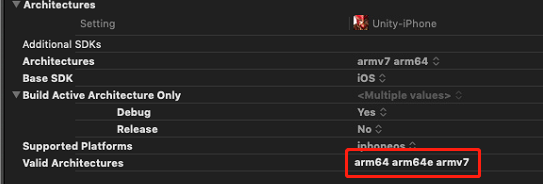
**Info：**

复制模板的Info.plist即可，这里不需要改，但要注意将Application does not run in background设置为NO。

Build Settings：

搜索Enable Bitcode：将其设置为NO。

将Valid Architectures中的armv7s去掉，假如有。



在Other Linker Flags添加两个标记-ObjC，-fprofile-instr-generate。

C:\Users\ADMINI~1\AppData\Local\Temp\1553512666(1).png

Search Paths一般都不用改，但是很多报错都是这里的问题：

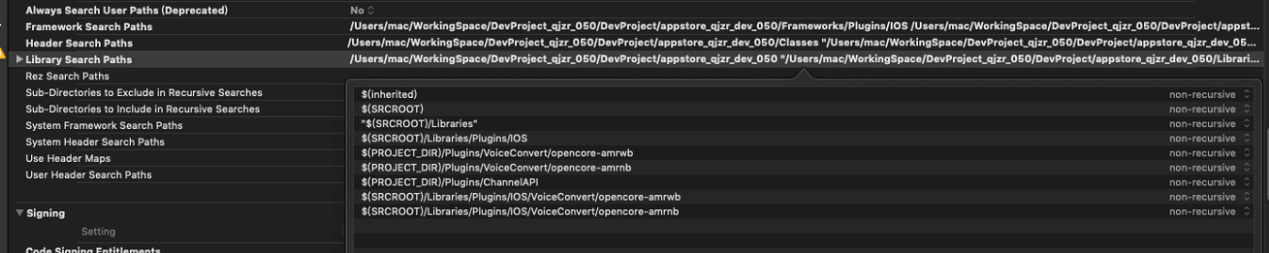
需要注意的是Framework Search Paths要配置下面两个路径，以方便查找到依赖包：

$(PROJECT\_DIR)/Plugins/ChannelAPI

$(PROJECT\_DIR)/Plugins







**Build Phases：**

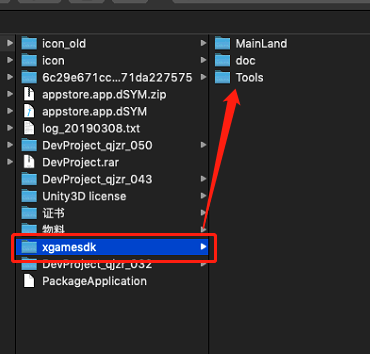
Compile Sources有个TakePhotoController.mm文件，双击，添加-fno-objc-arc

C:\Users\ADMINI~1\AppData\Local\Temp\1553514646(1).png

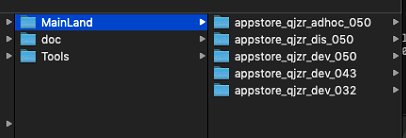
到这里配置就完成了，在xcode选择Product>Build编译，直到编译成功，如果报错则查找原因排除，继续Build直到编译成功。

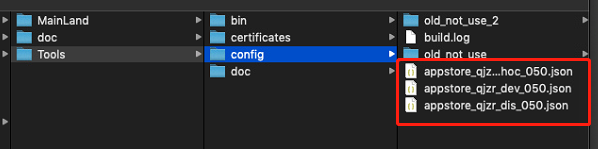
1. **Build成功之后，使用mogo build工具打包。**

相关工具所在目录如下：



MainLand中一个项目文件夹对应一个json文件，如下：





自动打包的代码在bin/iosPyTools/build.py中。

步骤4执行成功之后，将xcode项目的：

-Unity-iPhone

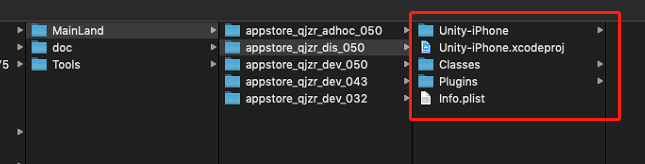
-Unity-iPhone.xcodeproj

-Classes

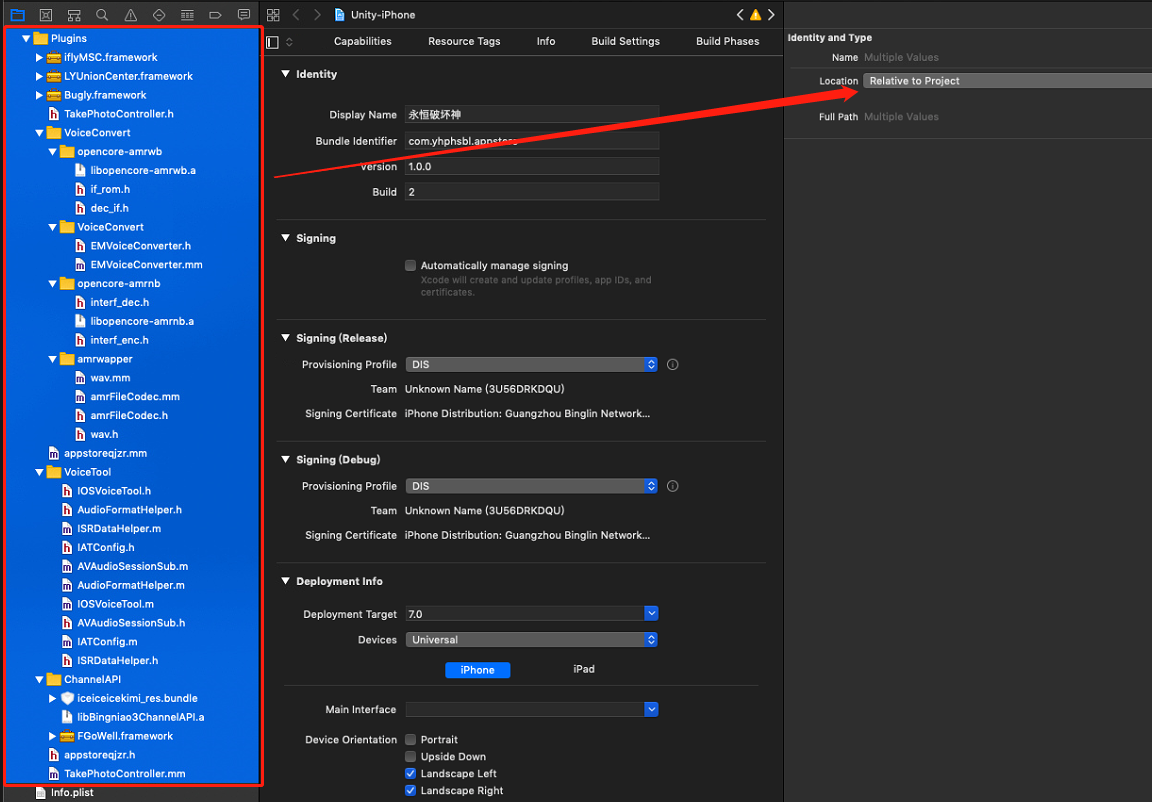
-Plugins

-Info.plist

拷贝到MainLand下的项目文件夹作为自动打包的模板。举个栗子：



双击打开Unity-iPhone.xcodeproj。需要修改Plugins文件夹的一个配置，否则自动打包会生成两份文件，导致打包报错，选中所有Plugins下的文件，将右边的Relative to Group如下：



接着配置json文件。

json文件可以直接从旧的复制一份，然后稍作修改即可用，需要修改的参数如下：

PlatformResourcePath：MainLand下面你拷贝出来的项目模板的路径，例如：MainLand/appstore\_qjzr\_adhoc\_050。

Version：版本号，跟xcode设置的一样即可。

UnityCommand：Unity指令，这里一般不用改，但是如果换了Unity版本需要改下路径。

DisplayName：游戏名，跟xcode项目设置的一样。

Identifier：包名，同理，跟xcode项目设置的一样。

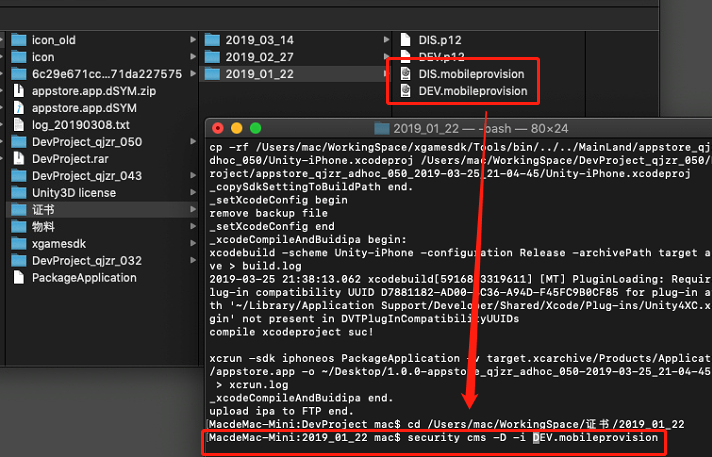
DEBUG\_INFORMATION\_FORMAT：选填“dwarf”“dwarf-with-dsym”，选择” dwarf-with-dsym”将以debug模式进行打包，速度较慢，会生成dsym文件，游戏崩溃的时候可以用这个文件来分析崩溃日志，一般打dev包会用这个选项。“dwarf”则是打dis或者adhoc包用。

Developer：这里填什么跟xcode设置的Signing有关，打DIS包或者打ADHOC包，这里填的是“iOS Distribution”。打DEV包这里填的是“iOS Developer”。

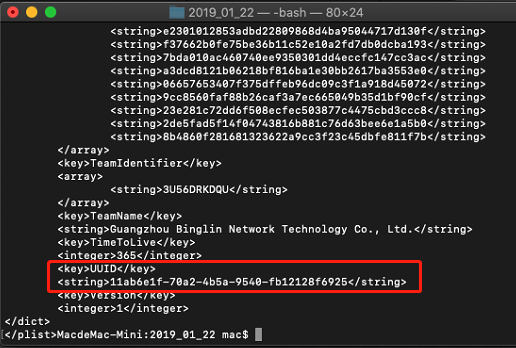
Provision：查询运营发给你的provison文件。查询方法如下：

命令行cd到存放provision文件的目录，然后执行

security cms -D -i XXX.mobileprovision



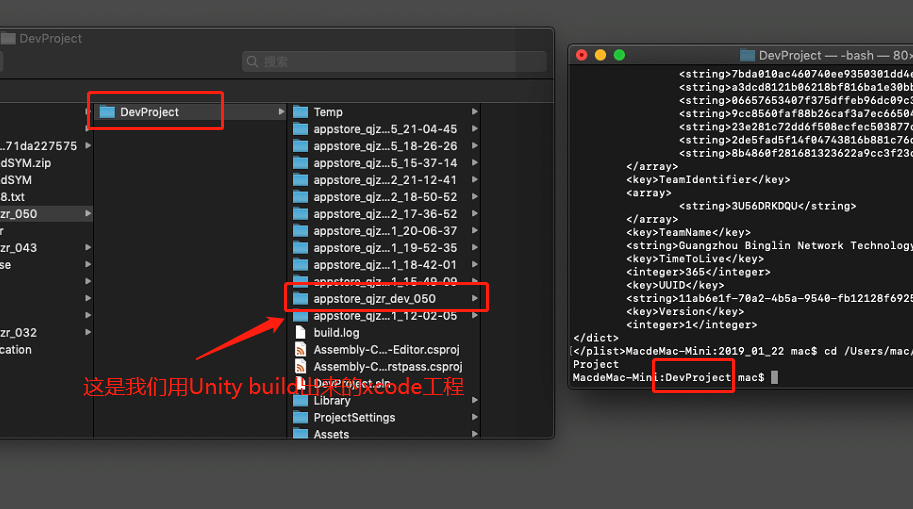
回车，出现的UUID就是要填在这里的。



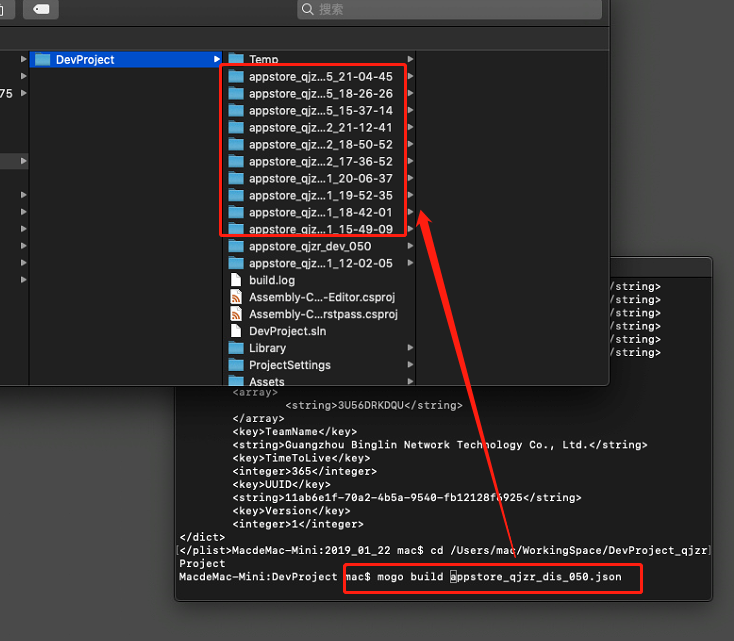
将这串字符拷贝到这里就可以了。

1. **最后一步：mogo build xxx.json**

第五步配置完成就可以开始打包了，首先cd到xcode文件夹所在的目录。



输入mogo build xxx.json。经过漫长的等待，xcode目录下会多生成一份xcode项目，项目名字最后会以你打包的时间作为后缀，如果配置没有问题就正常打出包，如果配置有问题，控制台会报错，这时候你要打开新生成的xcode项目，build一下，看看报什么错，解决问题再重新执行mogo build xxx.json。



1. **接入SDK：**

接SDK需要关注的文件及文件夹：

-Classes/UnityAppController.mm(初始化)

-Plugins/LYUnionCenter.framework(运营方提供的依赖)

-Plugins/ChannelAPI(运营方提供的依赖)

-Plugins/TakePhotoController.h(拍照相关文件)

-Plugins/TakePhotoController.mm(拍照相关文件)

-Plugins/VoiceTool(语音相关文件，接口都放在这个文件夹下的IOSVoiceTool.h)

-Plugins/VoiceConvert(语音格式转换需要的依赖文件)

-Plugins/appstoreqjzr.mm(接SDK的关键文件，暴露给c#的接口都定义在这里)

如果运营的接口有更新或改动，需要拿他们最新的依赖替换旧的。

接下来以游戏精灵体sdk接口为例，说明sdk对接过程：

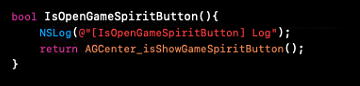
运营那边分别提供了接口：

打开游戏精灵体：AGCenter\_openGameSpiritWebviewWithRole(userInfo)

是否可以打开游戏精灵体：AGCenter\_isShowGameSpiritButton()

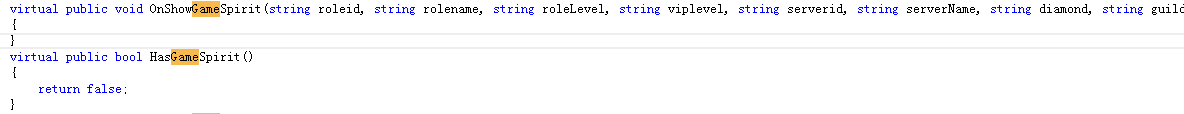
再appstoreqjzr.mm的extern “C”内部添加接口，如下：



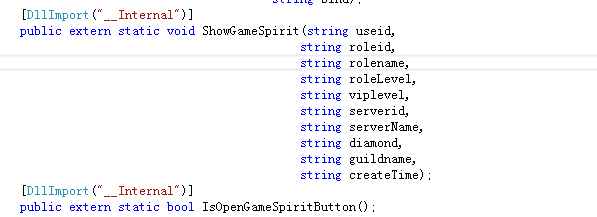


打开GameCode工程，在GameLoader/GameLoader/PlatformSdk/目录下有三个文件，PlatformSdkMgr.cs，IOSSdkMgr.cs，AndroidSdkMgr.cs。PlatformSdkMgr是IOSSdkMgr，AndroidSdkMgr的基类，新增接口需要先在PlatformSdkMgr类中定好接口，子类继承这些接口然后按照平台各自的需求去实现功能。

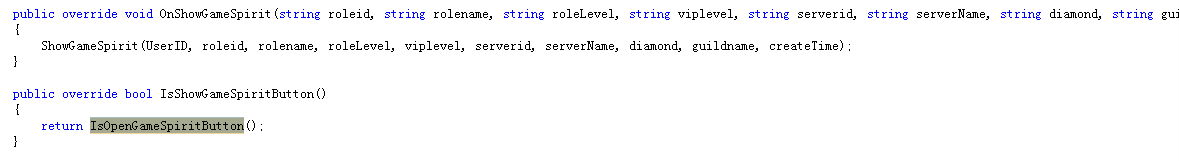
在PlatformSdkMgr定好接口如下，下面的接口通过LuaFacade.cs给Lua层使用，命名你开心就好：



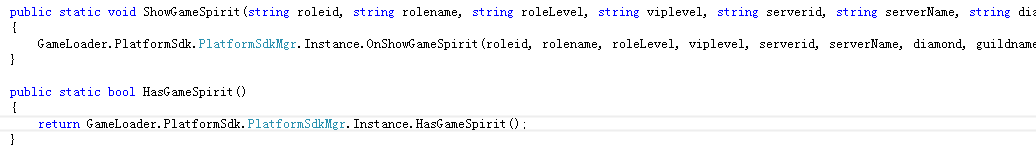
在IOSSdkMgr中定义调用iOS中写好的接口，这里的命名要跟iOS中定义的名字一样：



然后在IOSSdkMgr中实现从PlatformSdkMgr继承的接口，通过这些接口来使用iOS中的SDK接口：



将GameLoader重新生成，LuaFacade就能愉快的暴露接口了：



重新生成GameMain，Lua就能调用SDK的接口了。

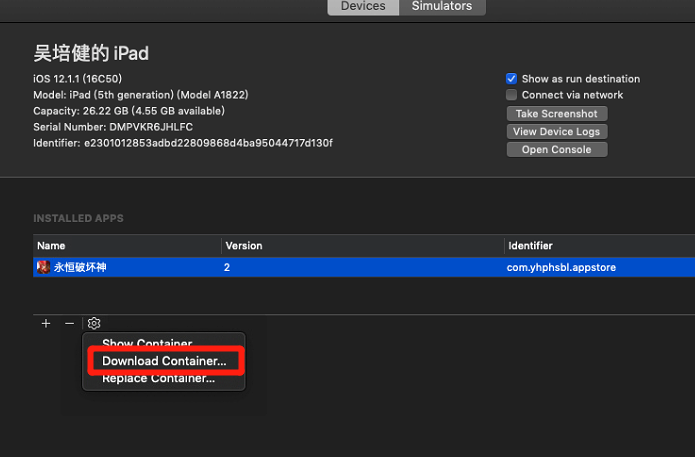
需要注意的是，在PlatformSdkMgr定义接口需要综合考虑好 接口的参数以及返回值，例如：同一个接口，ios可能有五个参数，而在Android可能不需要参数，这时候需要保证PlatformMgr中定义的接口能满足不同平台调用。最好是接之前向运营方确认好。接好之后重新生成GameLoader。

1. **Debug：**

崩溃：拿到崩溃日志进行符号化，然后根据符号化的日志定位问题。符号化教程：

<https://www.jianshu.com/p/c60d9578b656>

游戏中的日志：将装有游戏的设备连接到Mac，打开xcode，window/Devices and Simulators，选中游戏，选择Download Container:



将下载好的文件打开即可访问到游戏中的log。

1. **打包过程中的报错集合：**

1.问题： directory not found for option "..../Libraries",原因是lib找不到。

解决：在Build setting中搜索lib,查看Search Paths/Library Search Paths里配置的路径是否有问题（一般这里的路径不加双引号）

2.问题：libtolua.a报错

解决：Build option Enable Bitcode -> no

3.问题：xcodeConfig配置Developer和Provision

解决：复制Build Settings/Signing 中的Code Signing Identity和Provisioning Profile

4.问题： ARC报错

解决：

target -> Build Phases -> Compile Sources

双击报错的 \*.m 文件

在窗口中输入-fno-objc-arc

如果使用的非 ARC ，则为 ARC 的代码加入 -fobjc-arc

如果使用的是 ARC ，则为非 ARC 代码加入 -fno-objc-arc

判断项目是否使用arc：Build Setting -> Automatic Reference Counting Yes/No

5.问题：linker command failed with exit code 1(use -v to see invocation)

解决：同名文件，删掉重复的即可

1. 问题：游戏运行中切换到后台再回到游戏，游戏闪退。

解决：查看Info.plist文件的key Application does not run in background的值是否为NO。正确应该填NO。