将iOS项目部署到Android

相信大部分读者使用的是Android操作系统的手机。既然Cocos2d-x是跨平台的，那么我们就应该可以将自己做的项目安装到自己的手机上。想象一下，自己玩着自己做的游戏，是多么有意思的事情？本章就讲解如何将把我们的项目部署到Android真机上。

17.1 相关概念

在进行移植前，我们首先看看需要那些软件，大家对这些软件应该有所了解。

* Eclipse：支持Java和Android开发的集成开发环境。
* JDK（Java Development Kit）：Java开发工具，没有JDK则无法编译Java程序。
* ADT（Android Development Tools）：Android开发的Eclipse插件，为Android开发提供开发工具的升级或者变更，可简单理解为在Eclipse下开发工具的升级、下载工具。
* CDT（C++ Development Tools）：Eclipse开发C++项目的插件。
* Android SDK：Android开发工具，包括模拟器。
* Android NDK：Android原生态本地开发包，辅助SDK进行编译开发，支持C/C++。
* Apache-ant将软件编译、测试、部署等步骤联系在一起加以自动化的工具，大多数用于Java环境中的软件开发。

明白这些软件的作用后，下面我们就开始将我们的项目部署到Android真机上。

17.2 Mac下的环境搭建

在第2章已经详细讲解了Mac开发环境的搭建，下要要讲解的Mac下的真机测试是在有Mac下Cocos2d-x开发环境的前提下的，如果还没有搭建Mac下Cocos2d-x的开发环境，那么应该回到第2章，先将Mac下Cocos2d-x的开发环境搭建好。如果已经搭建好了Mac下的开发环境，并且已经开发了一个项目，正准备将它安装到自己的Android手机上，那么就可以进行接下来的操作了。

### 17.2.1 准备工作

（1）Android开发环境的下载地址（Eclipse和Android SDK）为：<http://developer.android.com/sdk/index.html#download>。

（2）Android NDK的下载地址为：[http://developer.android.com/tools/sdk/ndk/index. html#Downloads](http://developer.android.com/tools/sdk/ndk/index.html#Downloads)。

（3）apache-ant的下载地址为：<http://ant.apache.org/bindownload.cgi>。

软件下载好之后，其列表如图17-1所示。为了方便后面的操作，我们将压缩包解压到一个目录中。注意路径全部用英文，可以不要有空格和特殊字母，并且对这些目录及命名稍微进行修改，笔者这里的修改如图17-12所示。android-sdk-macosx就是adt-bundle-mac-x86\_64-2013 1030 .zip解压文件夹中的SDK。



图17-1 下载软件列表

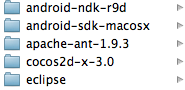


图17-2 软件列表

### 17.2.2 配置环境

下载软件后，就可以配置环境了，打开终端，进入Cocos2d-x 3.0目录，输入“python setup.py”（或者.“/setup.py”）,就会出现如图17-13所示的内容：

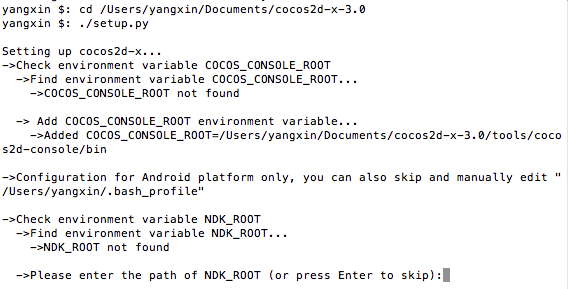


图17-3 运行setup.py

首先需要输入NDK\_ROOT的路径，即文件夹中android-ndk-r9d的路径；然后输入ANDROID\_SDK\_ROOT，即文件夹中android-sdk-macosx的路径；最后需要输入ANT\_ROOT即文件夹中apache-ant-1.9.3的路径。这些路径输入完毕后，其提示我们输入一句命令将设置的环境保存到文件中，笔者这里是“source /Users/yangxin/.bash\_profile”。可以通过下面的方法检查一下环境是否已配置好了。

友情提示：输入路径的时候，可以直接将文件夹拖入到终端中。

我们重新打开一个终端，测试一下环境是否配置成功，在终端输入：

cd $NDK\_ROOT

pwd

如果能显示我们设置的路径，则路径NDK\_ROOT已配置成功。也可以使用同样的方式检查另外两个路径，即$ANDROID\_SDK\_ROOT和$ANT\_ROOT，如果这些路径都没有问题，那么我们的环境就配置好了。

### 17.2.3 编译

将环境配置好之后，就可以动手将我们在Xcode中已经开发好的项目部署到我们Android真机。首先，使用Xcode打开proj.android /jni/下的Android.mk文件，在“LOCAL\_SRC\_FILES := ”后面添加自己新写的文件（.c或者.cpp文件，不用添加.h文件，并且按照添加AppDelegate.cpp的格式书写），每个文件一行，用“\”续行。我们还要确保代码文件都在class文件夹里面，资源文件都在resource文件里面，并且如果资源文件在某个子文件夹下，那么在Android下文件的路径是需要加上这个文件的名字的，例如”pic/HelloWorld.png”。

最后打开终端，进入到proj.android目录中，其操作如图17-4所示，运行.“/build-native.py”就开始编译。这个过程比较长，请大家耐心等待。编译成功后可以看到如图17-5所示的内容。编译完成后就可以进入17.4节，使用Eclipse对项目进行真机测试或者打包。当然，如果有问题，那么我们还需要修改C++代码，然后再运行build\_native.py编译C++代码，最后在Eclipse刷新项目。

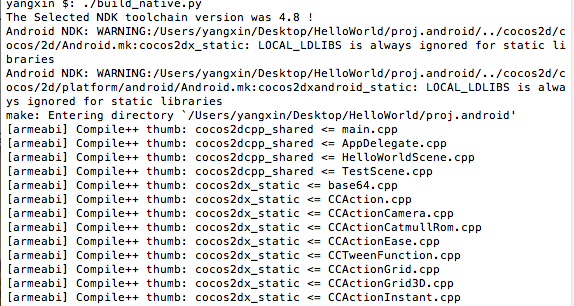


图17-4 编译项目

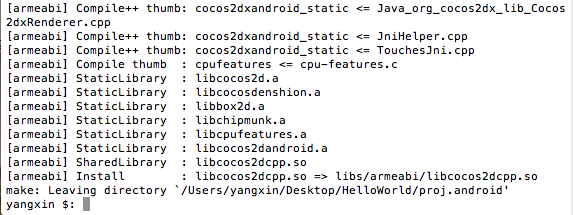


图17-5 编译完成

17.4 Android真机测试和打包

### 17.4.1 导入

打开Eclipse，如果是首次打开，那么需要设置工作目录如图17-10所示。

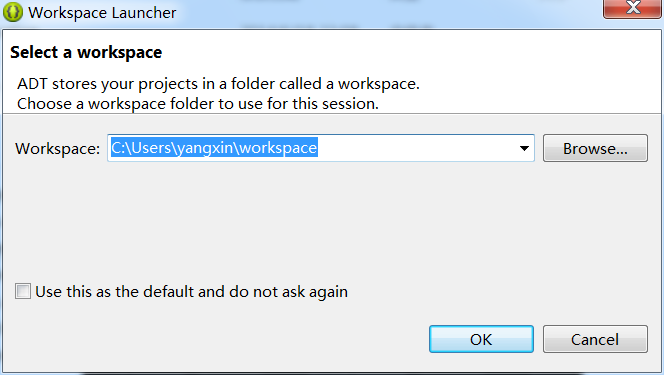


图17-10 指定工作区目录

然后需要配置Android-SDK的路径。这里Mac版本的Eclipse和Windows版本的Eclipse稍微有点区别：在Mac下单击ADT→Preferences→Android。在Windows下单击Windows→Preference。然后在所弹出视图的右边输入SDK路径，最后单击Apply按钮，再单击OK按钮就好了。如图17-11所示。

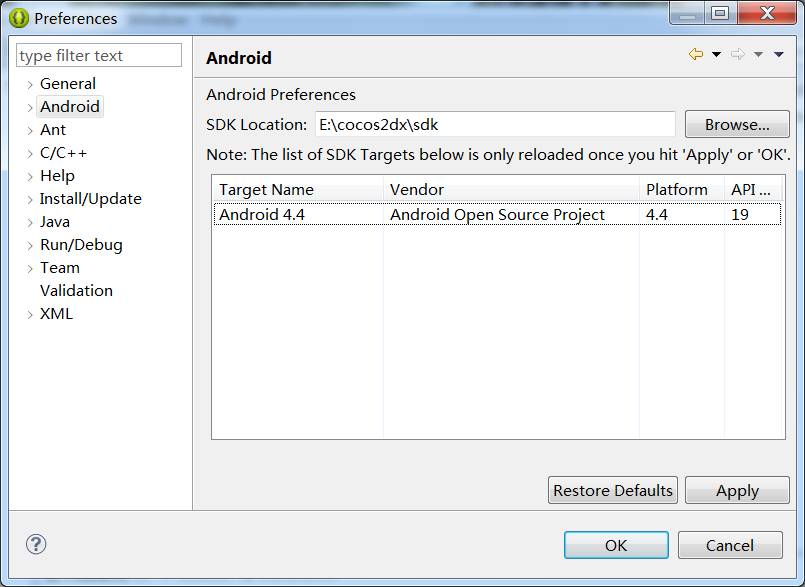


图17-11 指定SDK路径的Windows截图

然后我们就可以将已经编译好的项目导入到Eclipse中。单击File→New→Project→Android，会弹出一个窗口，如图17-12所示。选择Android Project from Existing Code文件夹后单击Next按钮，会弹出如图17-13所示的界面。单击Browse按钮选择我们的项目目录，就会出现如图17-14所示的界面.然后可以看到在Project to Import中有许多选项，它们是插件和我们的项目文件，笔者这个例子不需要其他插件，所以我们只需要选择libcocos2dx和HelloWorld这两项就可以了。这些插件包括多盟广告、友盟等，根据项目的具体需要可以选择响应的文件。

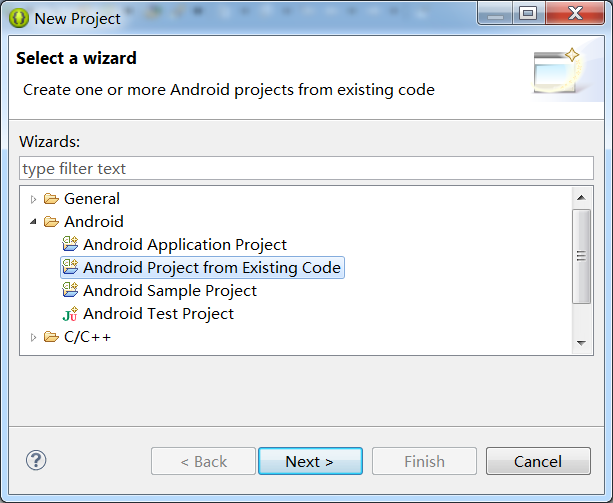


图17-12 选择新建选项

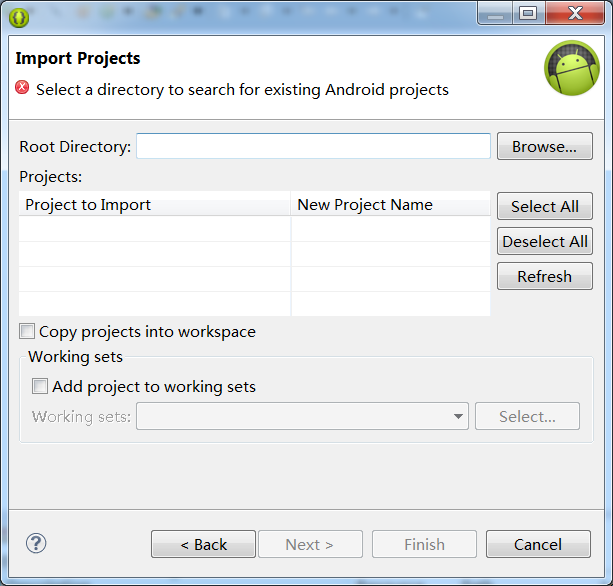


图17-13 导入项目

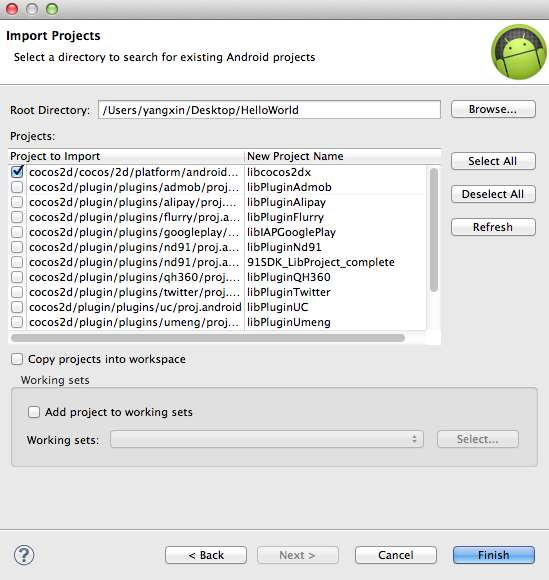


图17-14 选择插件

单击Finish按钮，就将我们的项目导入到了Eclipse中，导入后可以看到我们的项目目录如图17-15所示。

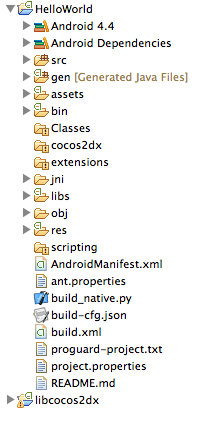


图17-15 项目目录展示

### 17.4.2 真机测试

项目成功导入后我们就可以进行真机测试了。如果需要输出调试的信息，则首先需要在AndroidMainfest.xml文件中将

<application android:label=”@string/app\_name”

android:icon=”@drawable/icon”>

修改为：

<application android:label=”@string/app\_name”

android:icon=”@drawable/icon”

android:debuggable=”true”>

单击工具栏中的debug按钮（就是那个小虫子），就会弹出如图17-16所示的界面，这时如果你的手机使用USB数据线连接电脑，则会看到自己的手机，如果没有看到，则应该是驱动的问题。如果你知道需要安装什么驱动，那么就直接安装这个驱动，如果不知道，那么安装一个豌豆荚，它会帮你安装手机需要的驱动。然后选择真机，再单击ok按钮就会出现如图17-17所示的界面，单击ok按钮，我们的项目就会在真机中运行起来,如图17-18所示在logcat中就可以看到我们的程序输出的内容了。

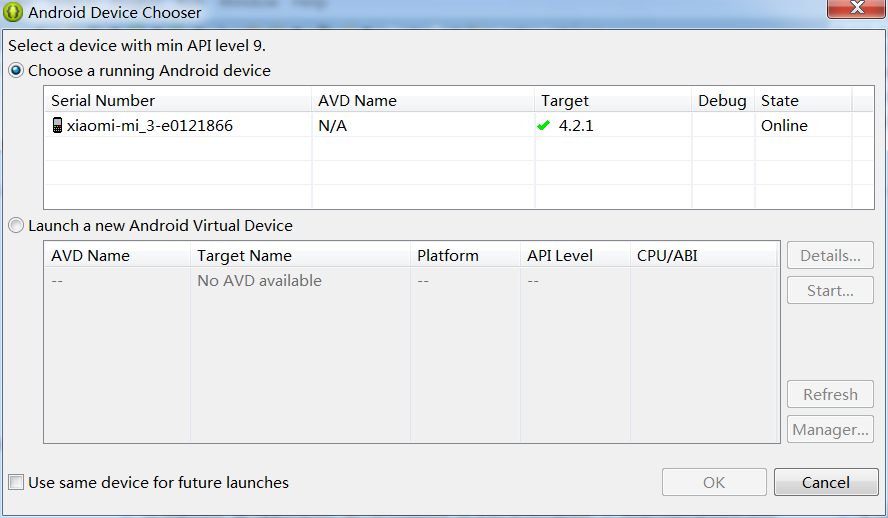


图17-16 选择真机

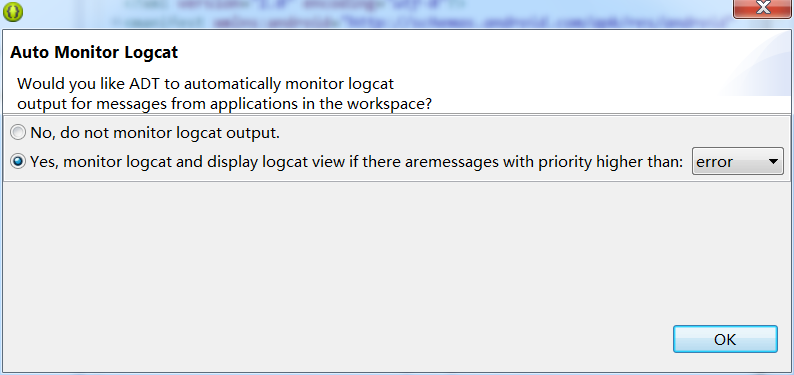


图17-17 logcat

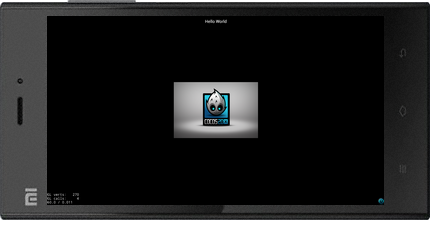


图18-18 真机截图

### 17.4.3 打包

在把程序都调试好了之后，我们就可以使用Eclipse把项目打包成可以安装到Android真机的程序包。下面我们来看看怎样打包APK。

用鼠标右键单击工程名：Android Tools->Export Signed Application Package，如图18-15所示。

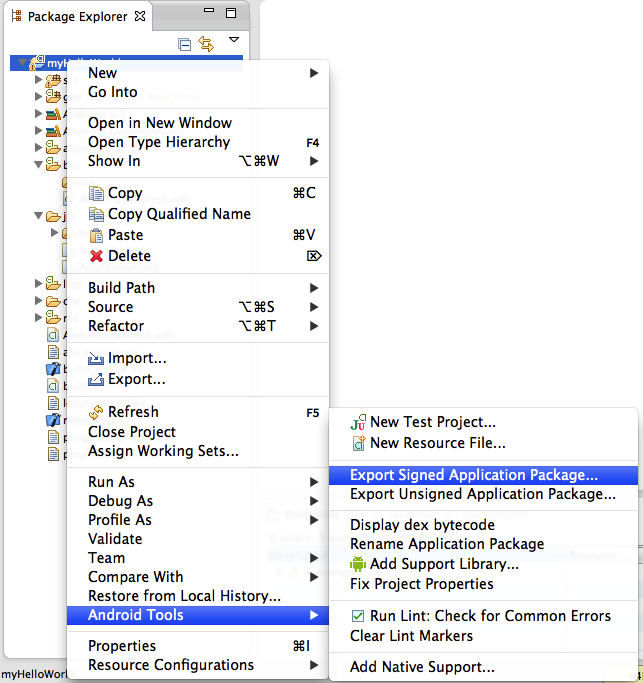


图17-15 导出签名的APK

单击Export Signed Application Package，会出现如图17-16所示的界面。

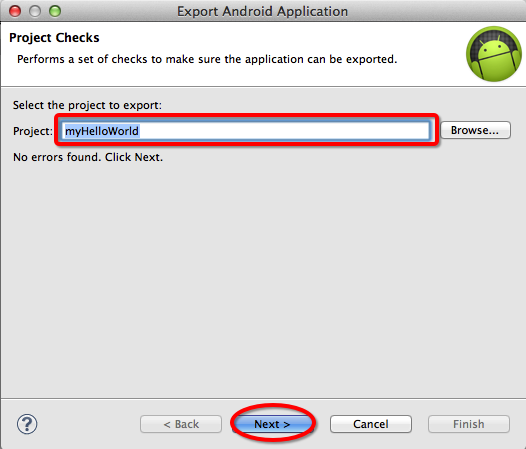


图17-16 需要打包的工程

粗的矩形框圈住的部分默认为工程名，不用修改，直接单击Next按钮，会出现如图17-17所示的界面，选择Create new keystore，创建一个新的Android签名。

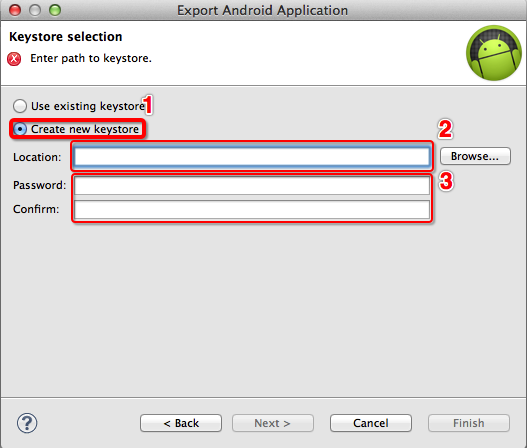


图17-17 创建keystore

输入保存的位置、名称、密码并确认密码，然后继续单击Next按钮，如图17-18所示。

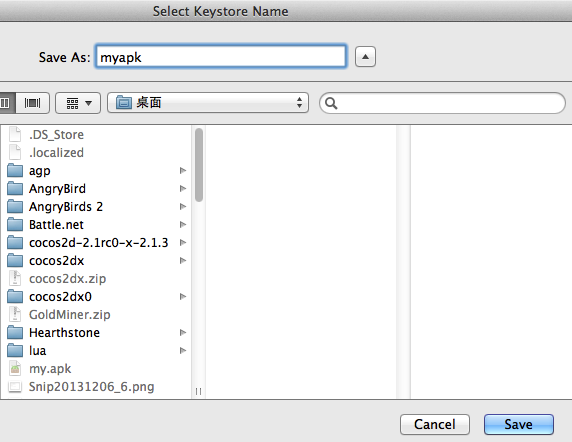


图17-18 保存keystore

如图17-19所示的“1”和“3”区域，可以根据自己的实际情况进行填写。在“2”区域填数字，最好是填写25。单击Next按钮，如图17-20所示。

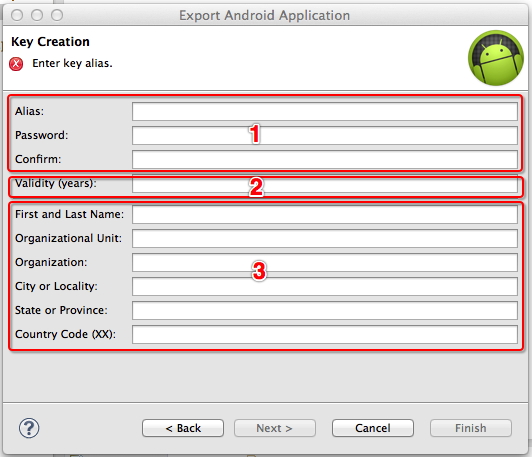


图17-19 输入内容

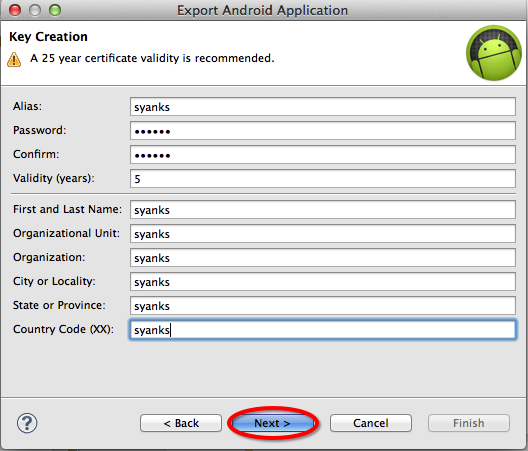


图17-20 内容输入完毕

最后一个Destination APK File就是指定的最终生成APK的存放路径，指定好后，单击Finish完成，如图17-21所示。

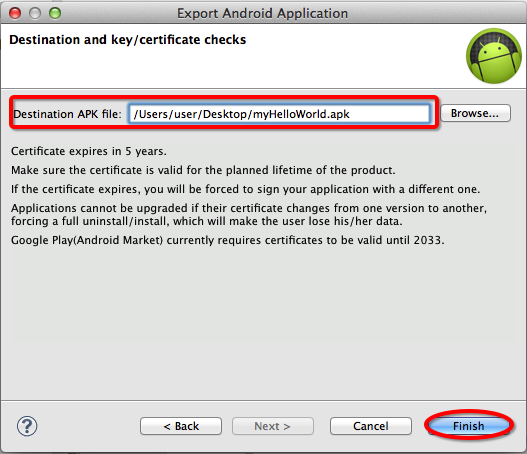


图17-21 指定APK路径

最后，我们从上面指定的路径找到我们打包的APK文件，如图17-22所示将其通过数据线或者蓝牙传到手机上，然后就可以安装和使用了。

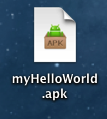


图17-22 打包的APK文件