打包示例

# 打包我们的Bundles

## 首先要看一眼Assets目录下的product.json配置文件

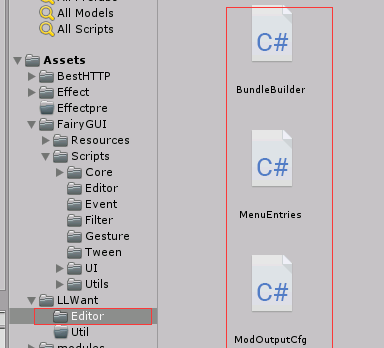
product.json配置文件主要给我们的打包代码提供配置，告诉打包代码，当前有多少个模块，每一个模块有多少个bundle需要打包，以及每一个bundle里面包含什么类型的文件等等。

如果product.json没有配置模块，则打包后就缺少该模块。

如果product.json漏了配置某个bundle，则打包后就缺少该bundle，游戏运行起来后，需要加载该bundle内的资源时，就会报找不到bundle的错误。

随着模块的增删，bundle的增删，都需要及时更新product.json文件。

更详细的打包流程，见下图代码：



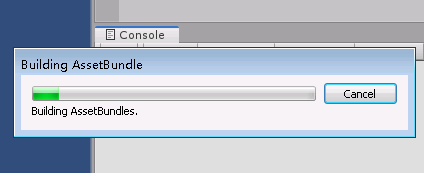
## 打包菜单，输出的位置和结果

首先，打包菜单在这里：

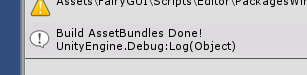


在assets菜单下的Build AssetsBundle。

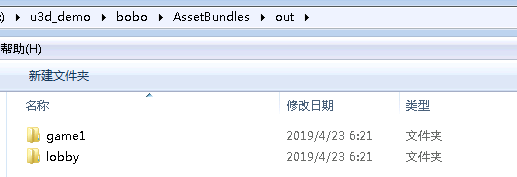
点击后，看似没有反应，等一会就出一个框：



然后再等一会，console窗口就会输出日志，如果没有错误日志，那么就表示成功了，最后一条日志是：



输出结果位于工程根目录下的AssetBundles/out目录下：

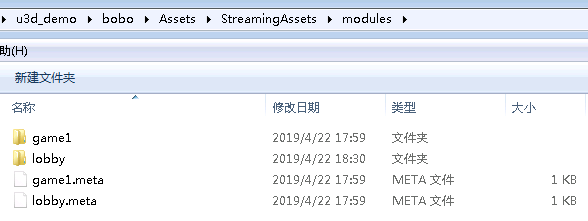


## 打包输出的使用

out目录下有若干子目录，每一个子目录对应一个模块。

可以把子目录单独压缩zip，然后放到服务器做更新用途。

也可以把子目录拷贝到Assets目录下的StreamingAssets/moduels子目录（如果不存在则自己新建一下），然后build 游戏时，Unity会原封不动的拷贝到build 好的安装包中：



# 生成XLUA的包装代码

由于我们需要build游戏成为不依赖于Unity编辑器而运行的独立程序，因此生成XLUA用的包装代码比较有意义：一来是加快运行速度（否则Unity会使用C#的反射来寻找代码，反射是比较慢的）；二来是防止IL2CPP裁剪。

在IOS平台下，由于IOS禁止运行时生成代码，也即是禁止JIT，而C# 是依赖JIT的，因此需要把C#转换为C++代码，Unity是用IL2CPP这个工具链来做这一步的。IL2CPP优化时，如果发现存在没有人用的代码，就会裁剪掉。例如，UnityEngine.SystemInfo这个类的代码，由于咱们并没有在C#代码中使用这个类的代码，只是在LUA里面使用，但是IL2CPP不考虑LUA代码的，因此IL2CPP还是认为UnityEngine.SystemInfo这个类的代码没有人用，于是就裁剪掉。那么当我们的游戏跑起来后，LUA里面需要访问到UnityEngine.SystemInfo这个类的代码时，就会说找不到这个类的代码了（已经被裁剪掉了）。

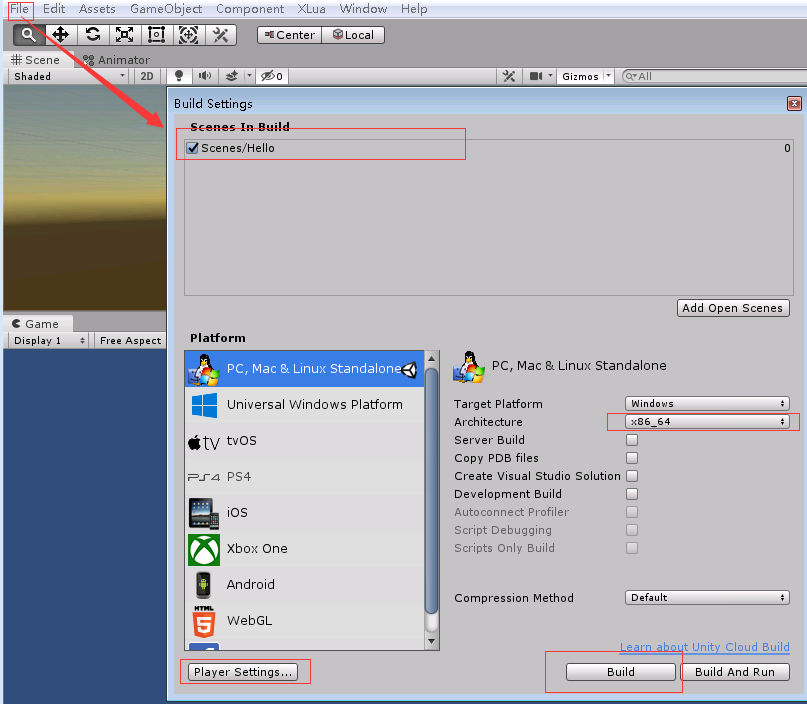
生成XLUA包装代码，就是显式的去引用这些我们仅在LUA脚本中使用的C#类，然后XLUA生成一个配置文件，告诉IL2CPP，不要裁剪任何XLUA生成的包装代码，因此最红，像UnityEngine.SystemInfo这样的代码就不会被IL2CPP裁剪掉。

平时开发的时候，咱们跑Unity Editor，不需要生成XLUA包装代码，因为生成包装代码过程很慢的，就让XLUA使用反射来处理就好了。

# Build For Win64

当走完上面两步后（打包bundles， 以及生成XLUA包装代码），我们就可以生成游戏了。

如下图所示：

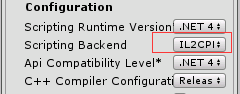


## C# backend的选择

这里需要指出的是，有两种脚本运行时可以选择，正如上面提到的，C#在IOS上需要转换为C++，因此，Windows平台，或者ANDROID平台下，可以跑C#的代码；IOS就需要跑转换后的C++代码。但是也可以任何平台都跑C++代码，也就是任何平台都使用IL2CPP来把C#代码转换为C++代码。

这个选择是位于上图的Player settings按钮，点击后如下图所示（在Other Settings页签下）：





使用IL2CPP生成是比较慢的，毕竟有需要转换C#为C++，然后还要编译C++。由于需要编译C++代码，因此需要Visual Studio 2015版本以上，而且安装了Visual Studio 的 C++编译器，更重要的是，也必须安装 Windows 10 SDK（这个意思是windows sdk 第10版本的意思，不是指windows 10这个系统专用的。 我win7也可以用。）

因此我们平时测试时，可以选择mono（它是一个跨平台的C# 运行时， 跟dot net framework一样，但后者只能运行于windows）作为C#运行时就可以了，不需要选择IL2CPP。IOS下才需要考虑IL2CPP。选择mono作为backend，就不需要安装windows 10 sdk了，同时build也较快。