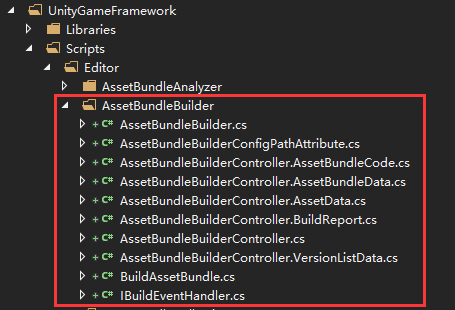
StarForce资源热更模式的实现

**一：实现理由**

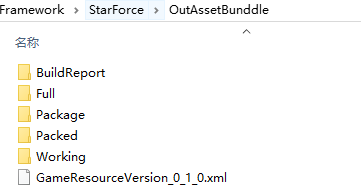
Ellan的UnityGameFramework已经做得非常完整，可用于项目实战了。接触这个框架以来，先了解了一下里面的ObjectPool，Event，FSM，Procedure及Entity系统，不得不佩服Ellan深厚的框架设计能力。当然，在对框架有了基本的了解之后，我的目光逐渐转移到了Resource的管理上，特别是资源热更新，相信不少人都对这块实现比较感兴趣，说直白一点，如果一个框架对资源的管理比较烂的话，其他的都白搭，只能拿来玩玩，不能用于实战。果然，Ellan这块设计再次让我五体投地，不过唯一遗憾的是，我把Resource Mode改成Updatable后，StarForce不但没跑起来，还抛异常了，自己撸了半天代码后，修修补补终于让这个流程跑起来了，其实改动不多，主要是要了解Ellan资源更新这块的用法。然后我也很诧异，为什么StarForce这个Sample中Ellan没有把这块流程实现出来（难道是我代码有什么问题？），所以写了这么一个文档，一是帮助大家了解一下这块流程，二是希望Ellan看看我这样是否有什么问题，最好能在官方版本中把这块流程走通了，增加一个文档帮助大家简单地搭建一个热更新环境。有问题可联系QQ:471812771（李春），再次致敬Ellan，每一个开源者都是伟大的。

**二：热更新资源的准备**

要热更，首先资源的管理上得支持，UGF的AssetBundle Builder给出了自己的解决方案。这块代码在这里：



用AssetBundle Editor编辑好AssetBundle后，用这个工具把资源打包成AssetBundle（不了解这块的可以先看看Ellan的官方教程：http://gameframework.cn/archives/356），会生成这样一个目录结构：



其中Working目录下是打出来的AssetBundle的缓存目录，缓存嘛，这个目录下的东西我们正式发布时是不会用的。

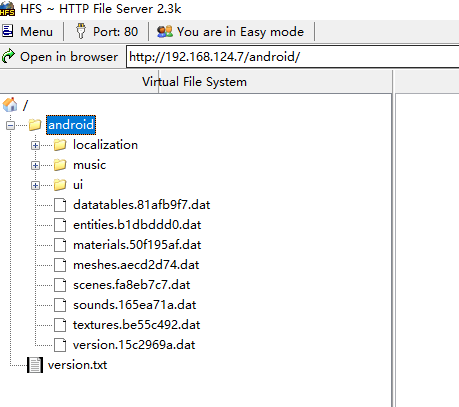
Full, Package和Packed三个目录下的是真正打包好的资源，其中Package目录下的是单机模式用的资源，直接将这个目录下的文件拷贝到Assets/StreamingAssets目录下即可发布单机版本。Packed目录下的是发布Updatable版本时随着APP一起发布的资源，也是需要拷贝到Assets/StreamingAssets目录下发布，哪些资源需要做到Packed包里，在AssetBundle Editor中可以设置：

IMG_258

之后就是Full目录，这下面的是完整资源包，需要拷贝到你的资源服务器上。如果构建AssetBundle时你选择了Zip压缩，你会发现这个目录下的资源名称中间都带了一个数字，比如textures.be55c492.dat，那是资源的HashCode，为什么Full目录下的要用Zip压一次呢，因为是要放服务器上让玩家更新的，当然是为了加快下载速度和省流量了。

最后还有一个GameResourceVersion\_0\_1\_0.xml这样一个文件，这是你本次资源打包后的version信息，这个也很关键，在资源更新时会用到，后面会讲。

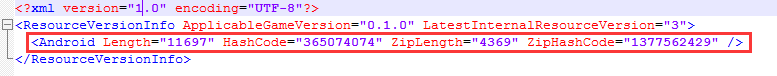
既然是热更新方式，那我们就做一个全部资源都热更的版本吧，就是StreamingAssets这个目录下你什么东西都不要放，之后全靠热更新下来。假设你的资源都已经通过上面的方式打包完成了，首先你需要创建一个自己的HTTP服务器，我这里用HFS软件搭建了一个，把你Full目录下的资源放到里面，咦，version.txt哪里来的，先别管，后面会说到：



**三：热更新的流程**

因为这块在StarForce中并没有跑通，我只能根据自己的经验和Ellan关于AssetBundle Builder的代码来猜。强烈建议把AssetBundleBuilderController中相关代码阅读一遍。首先，上面服务器上的资源（即Full目录下的资源）是Unity打出来的AssetBundle经过加密（如果选择了Load from memory and Decrypt或quick Decrypt）和Zip压缩（如果选择了Zip压缩）后，加上资源的hashcode在加上了一个dat后缀生成的。其中version.dat中记录了每个AssetBundle的原始大小，原始HashCode，压缩成Zip后的大小及Zip包的HashCode，你本地的AssetBundle的HashCode与服务器上的进行对比，如果不一致就需要更新，Zip包的HashCode是干啥的呢，因为你从服务器上下的资源是Zip压缩后的，资源下载下来后，需要计算你下载下来的这个资源的HashCode与服务器上这个资源的ZipHashCode做一个对比，不一致就说明你下载出错了，这个资源不能用。然后解压后，也需要将解压后计算得到的HashCode与服务器上的对比，ZipHashCode和解压后的HashCode都一致这个AssetBundle才能用。

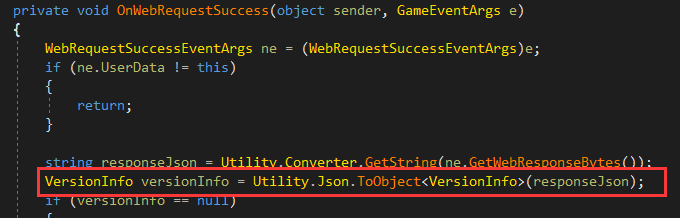
version.hashcode.dat这个文件是放在服务器上的，做资源更新时，你首先就好获取这个文件，那么这个文件你怎么校验呢，这就到了前面提到的GameResourceVersion\_0\_1\_0.xml这个文件，里面记录了最新资源的版本号，以及你的version.hashcode.dat这个文件的特征值。



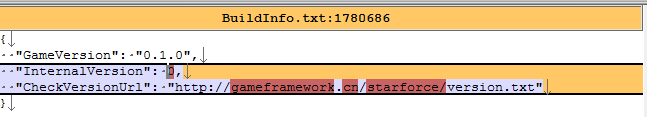
StarForce的资源更新检查是在ProcedureCheckVersion这个流程里面做的，我们可以看到整个工程里面都没提到哪里用到这个GameResourceVersion\_0\_1\_0.xml的地方，但是在这个类的RequestVersion()函数里面，我们看到了这样一句代码：

|  |
| --- |
| GameEntry.WebRequest.AddWebRequest(GameEntry.Config.BuildInfo.CheckVersionUrl, wwwForm, this); |

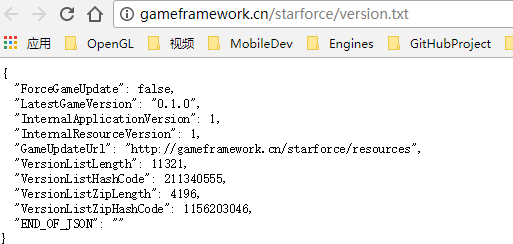
如果这个WebRequest成功后，会回调OnWebRequestSuccess：



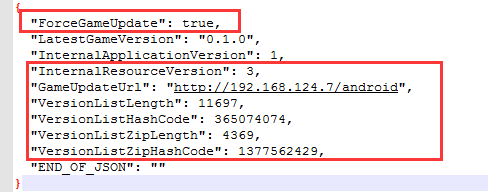
GameEntry.Config.BuildInfo.CheckVersionUrl是在Assets\GameMain\Configs\BuildInfo.txt中设置的，原始的是这样的：



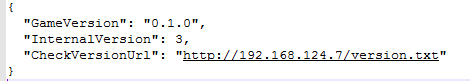
CheckVersionUrl指向了Ellan UGF上的一个地址，这个文件是长这样的，看到没，是不是跟GameResourceVersion\_0\_1\_0.xml这个文件的内容长得很像：



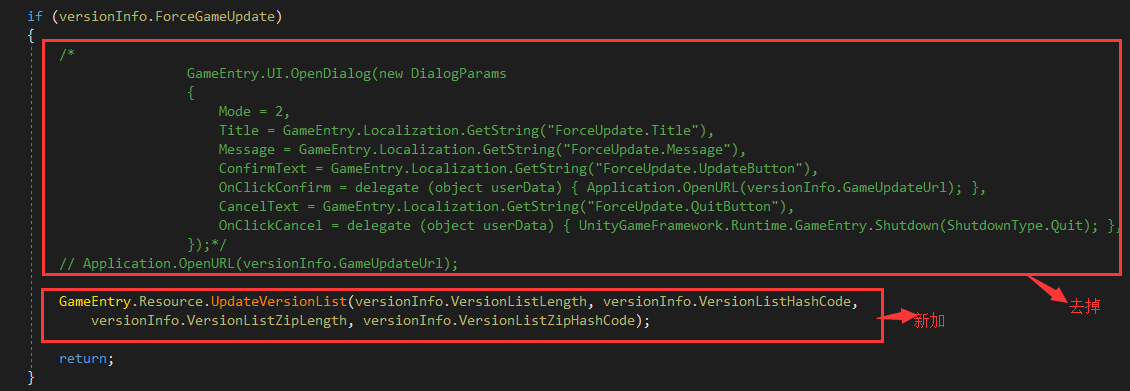
OK，到这里我们已经猜得七七八八了，服务器上的这个version.txt就是由GameResourceVersion\_0\_1\_0.xml来的，于是乎我们也创建这样一个version.txt放到我们自己的服务器上，就是上面提到的version.txt的来源。怎么改呢，GameUpdateUrl改成你自己的资源服务器地址，ApplicationVersion和ResourceVersion也改成你自己最新的版本号，后面的4个version.dat的特征值改成GameResourceVersion\_0\_1\_0.xml文件里的，比如我这里改好了，是这个样子的：



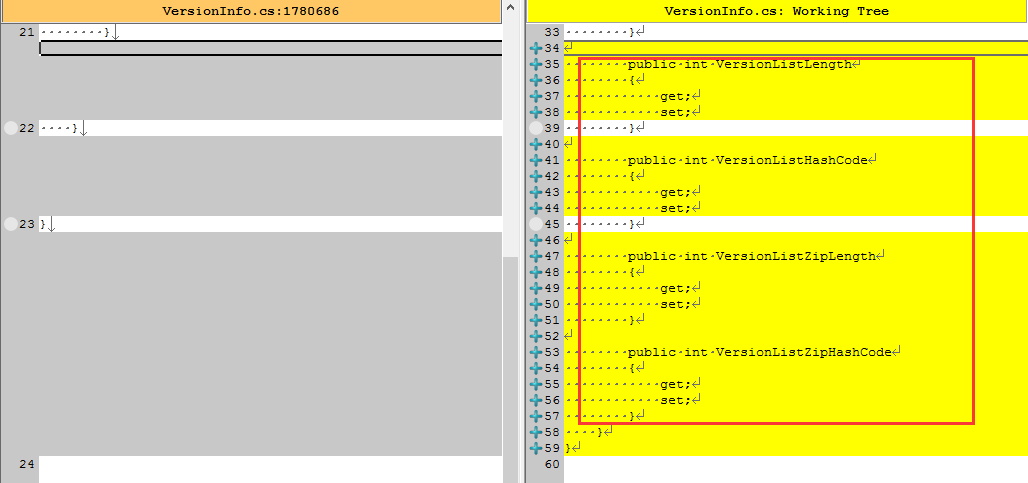
因为这个文件的URL是从Assets\GameMain\Configs\BuildInfo.txt中获取的，所以这个文件也要改，把CheckVersionUrl指向你自己服务器上的version.txt：



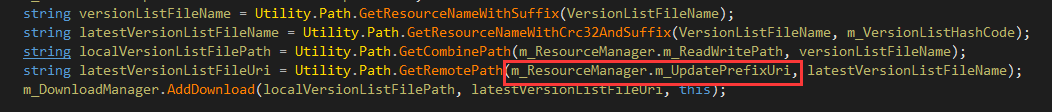
然后在ProcedureCheckVersion的OnWebRequestSuccess函数中做如下的修改：

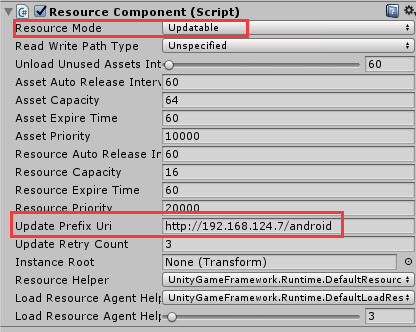


versionInfo的解析也需要增加上面这些数据的支持：



这个修改的目的就是，我已经获取到服务器上version.txt的内容了，知道了服务器上version.hashcode.dat这个文件的特征值，现在我需要从服务器上下载version.hashcode.dat这个文件来检测我本地资源哪些需要更新，这个函数会像服务器发起Download的请求，服务器上文件地址是m\_ResourceManager.m\_UpdatePrefixUri指定的，你可以将version.txt中的GameUpdateUrl设置给它，或者这里我简单地在编辑器中设置了：





顺口提一句，你需要将Resource Mode设置为Updatable才会启用资源热更模式。

之后我们需要做的就是：

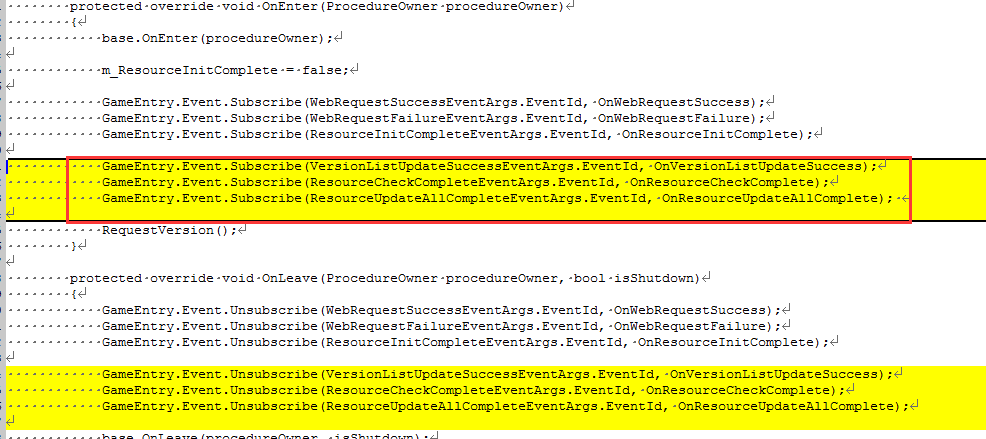
1：等待version.hashcode.dat这个文件下载完成，然后检查我们的资源是否是最新的。

2：资源检查完成后，对于我们不是最新的资源，向服务器发送下载请求。

3：所有资源下载完成后，退出ProcedureCheckVersion流程。

我们所需做的修改也都全部在ProcedureCheckVersion.cs这个文件里面：

首先，添加三个事件监听，对应上面说的三个步骤：一是监听version.hashcode.dat这个文件下载完成，二是监听资源Check完成，三是监听所有资源下载完成：



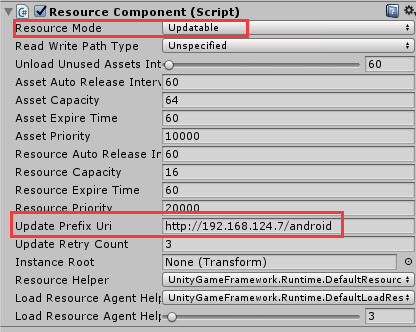
这三个事件处理也非常简单，按步骤进入下一个环节即可：



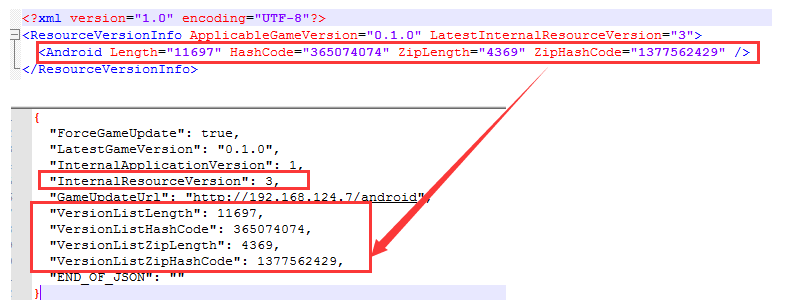
**四：我所做的修改的总体回顾**

前面按流程讲了一下我做的修改，这里总体列举一下前面的修改，其实非常简单，Ellan所有功能都已经实现了，只是没有在StarForce里面写完（个人感觉是，或者是随着更新这块功能缺失了）。

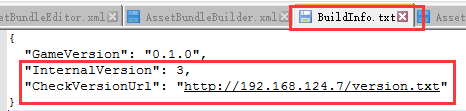
1：在Resource Component中设置Resource Mode为Updatable，设置Update Prefix Uri为你自己Http服务器的资源下载地址：



2：根据你资源的GameResourceVersion\_0\_1\_0.xml文件创建你自己的version.txt：



3：修改BuildInfo.txt，执行你服务器的version.txt

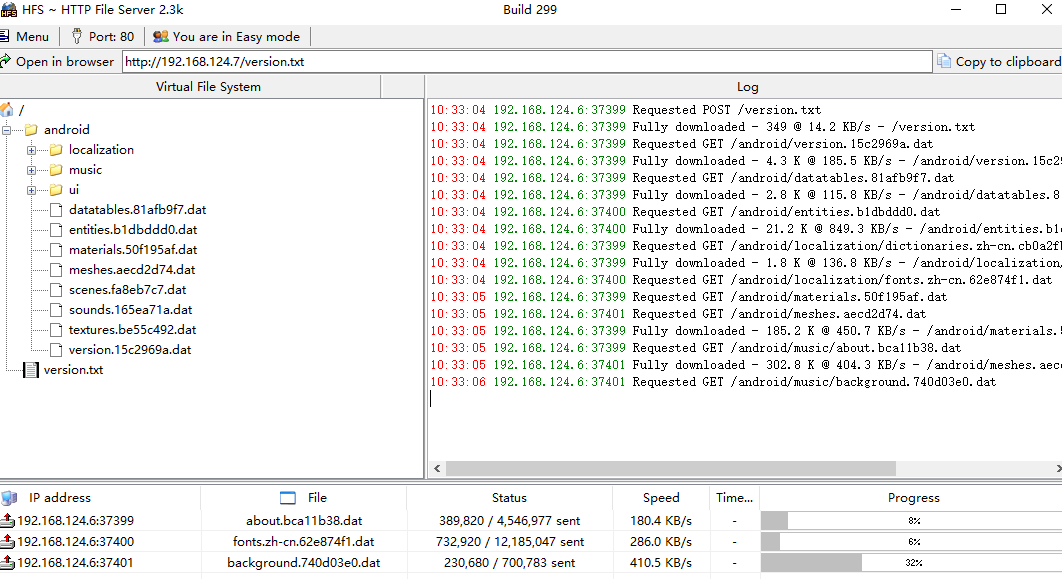


4：修改ProcedureCheckVersion.cs中的处理流程，即上面提到的OnWebRequestSuccess函数中的处理，以及三个事件回调的处理。

That's all, Enjoy it! 打包时，StreamingAssets目录下不要放任何东西，打出来的android包大小比全包小50多M，因为资源都没有：

IMG_275

手机上启动游戏后，你会发现你的Http服务器有资源请求：



等所有资源都下载完成后， 手机上就进入正常的游戏界面了，资源下载过程中你需要等着，可以自己做一个进度条。下载完成后进入你手机的Android/data/com.jiangyin.starforce/files目录，就会发现你下载的资源。再次进入游戏就不会有资源的下载了，但是你的http服务器必须开着，还是会请求version.txt和version.dat文件来检查你资源是否是最新的。