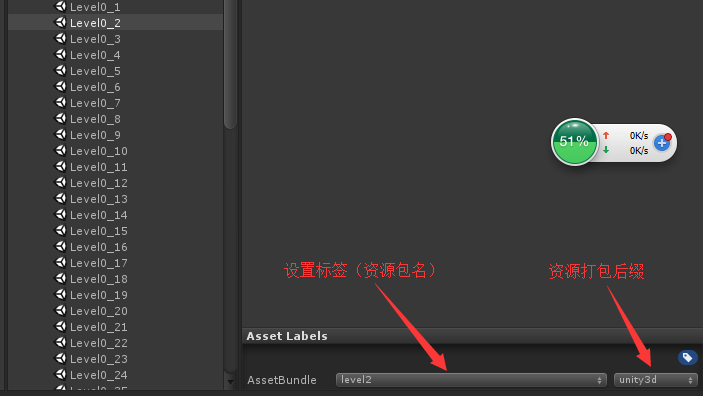
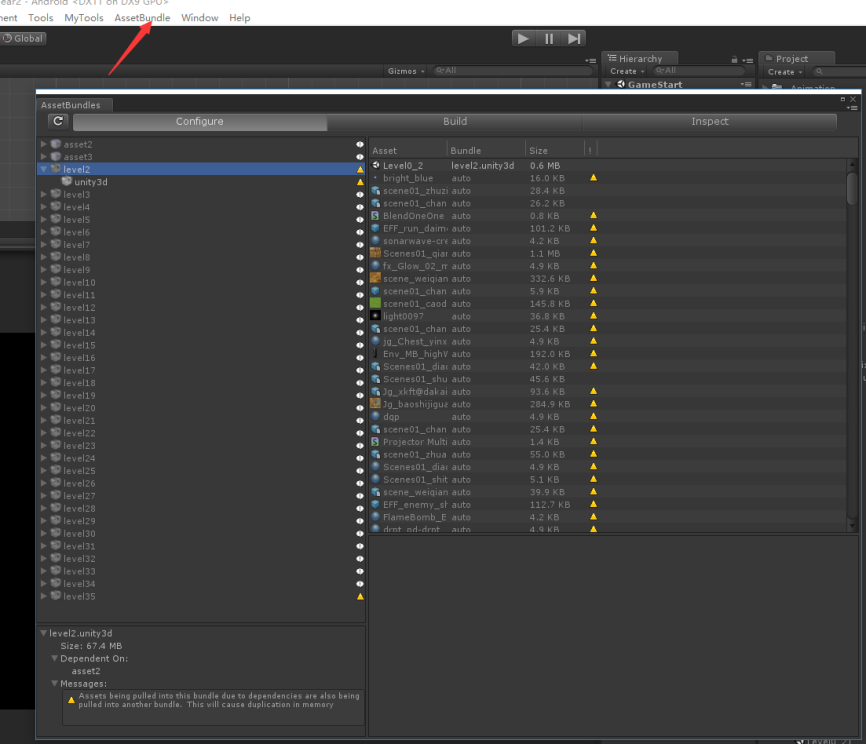
**资源热更新原理：**游戏运行过程中根据一定的策略，通过比对服务器和本地的资源版本号，来更新游戏本地资源。

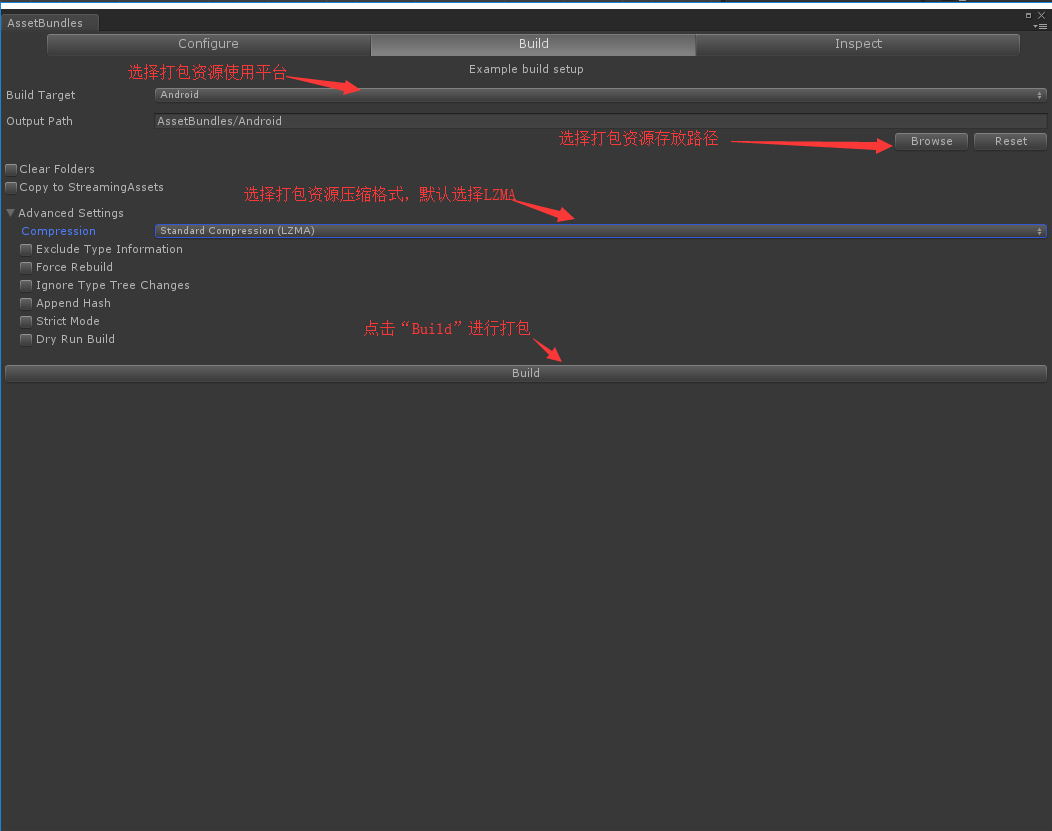
**资源打包原理：**在倒霉熊中采取的是按场景打包，然后对于每一章节对应每一关都用到的资源提出来单独打包，这样等于没个场景资源包都依赖于对应的公共资源包，减小了资源包的大小，也降低了下载量。这样打包完后如果要加载指定关卡的资源只需预加载它所对应的关卡资源以及公共资源包即可。

**打包程序设计：**在每次切换关卡时传入该管卡打包时的包命，然后加载Mainfest清单文件，在清单文件中获取该包名及其所依赖的资源包名，组成一个资源队列。然后依次预加载资源队列的文件(先加载依赖，最后加载管卡包)，等资源队列加载完成后切换关卡即可。

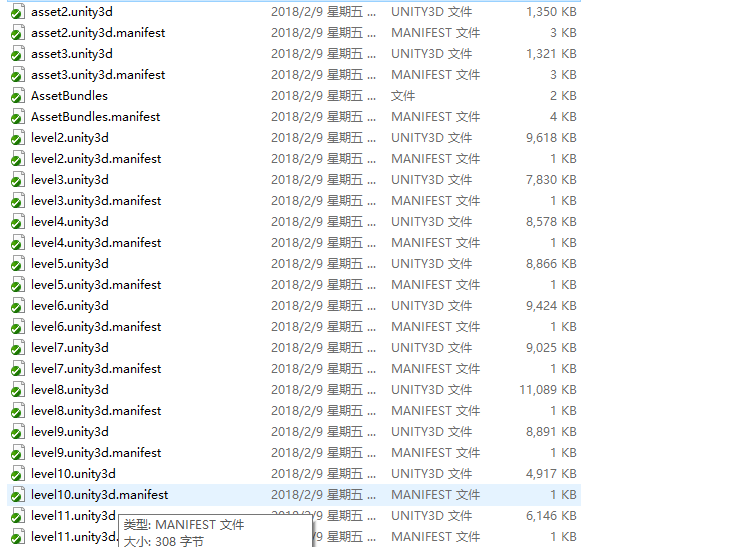
**打包工具使用：**

1：给要打包的资源设置标签（资源包名）和资源格式（打包后缀，无规则）

2：打包设置：点击菜单栏“AssetBundle—>AssetBundleBrowser”，弹出如下界面，里面可以浏览定义的指定标签所包含的资源清单。带黄色三角的表示该资源被自动引用到不同包（更改打包策略可以避免这种冗余，但要把握度）。

3，打包选项

注意：打包时确保同一个标签内的资源不允许重名存在，即使资源类型不一样也不允许，否则会导致打包报错，进行不下去。

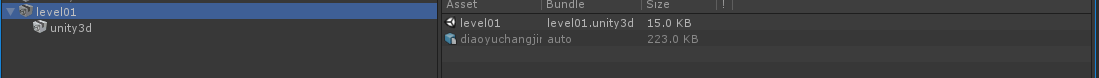
1. 打包资源文件分析：打包后的文件每一个文件会有一个“.unity3d”和“.mainfest”文件，其中“.mainfest”是方便开发人员观察的，后期可以删掉，只保留“.unity3d”做资源更新。另外打包完除了每个标签对应的资源包，还会有一个名为“AssetBundles”的文件，它里边存储这资源包之间的依赖关系，这也是Unity3d新版打包系统的一大优势，方便开发人员后期加载资源。

AssetBundle

AB包，本文的AB包就是AssetBundle包，是把资源进行压缩后，生成的一个压缩包。

根据上面文档的步骤，打包出Scene的AB包。

在测试时，打包了一个场景。



# AssetBundleManage说明

## 方法

LoadSceneFromFile 这个方法是用来加载场景AB包的。

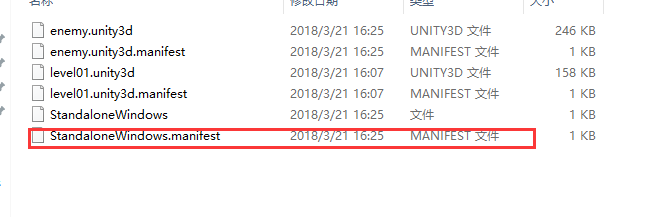
[LoadAsset](#_加载预设AB包) 是用来加载预设AB包的。（后面例子会说明）

我在使用 LoadSceneFromFile 的时候，遇到一个坑，死活加载不出来场景，网上找教程理解AssetBundle原理，分析LoadSceneFromFile 内部代码。

因为这里 AssetBundle bundle = AssetBundle.LoadFromFile(inPath + "AssetBundles");

inPath + "AssetBundles"，这里代码里是写死的。

首先，先加载总包。我在打包场景AB包的时候，有个总包，这个总包的名称是根据当时选择的平台自动命名的。



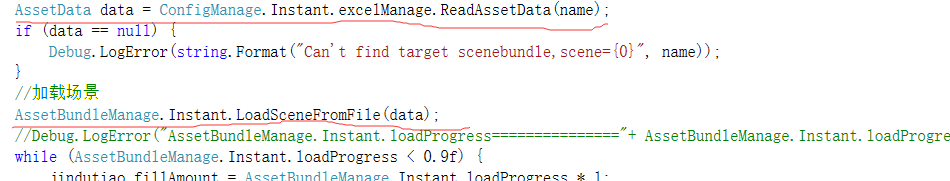
红色的，就是那个总包。然后把这里所有的东西，复制到Asset目录下的StreamingAssets文件夹里，这个文件夹需要手动创建，关于这个文件夹的功能、作用，网上有解释。

我是在UnityEditor测试的，所以，把AssetBundle bundle = AssetBundle.LoadFromFile(inPath + "AssetBundles");里面的outPath改成了inPath。

注意：

本地测试用inPath,从服务器下载到本地，用outPath。

倒霉熊项目中，加载场景的方法：

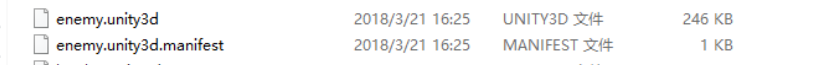


类比倒霉熊项目的方法，我在测试项目里这样加载场景。



## 将预设打包成AB包

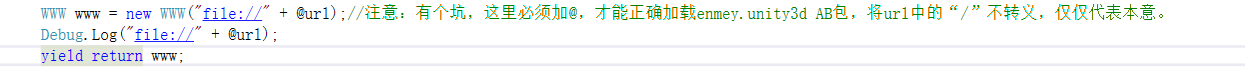
在测试中，把3个怪物制作成预设，然后打到一个AB包里。把包名设置成一样，就可以打进一个包里了。



Enemy.unity3d这个包里，是3个怪物的资源。

## 加载预设AB包

使用LoadAsset方法，在这里说下坑。

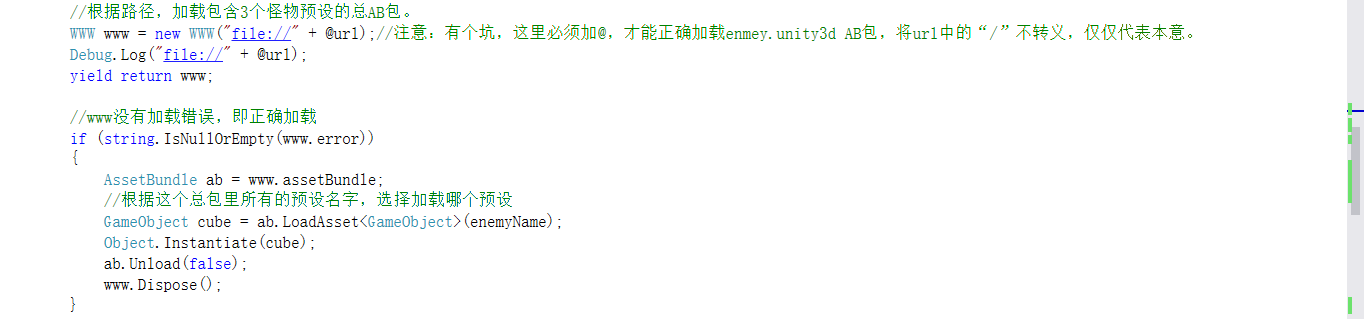


WWW类既可以加载网络路径，也可以加载本地，加载本地，需要在路径前加file://。

用起来，是这样的



说下内部的逻辑



先加载预设的AB包，用WWW来加载，这是一个异步操作，所以，会看到IEnumrator,yield return .

成功加载预设AB包（这里就理解为成功解压预设AB包），用LoadAsset方法直接根据名称加载这个包里的怪物。