测试1：自建shader，一个材质引用他，测试加到always前后材质的引用是否发生了改变，也就是验证是否加到always中的shader就不必打ab包了

结果：前后的meta文件都没有发生任何改变

1. 三种压缩方式： lzma,lz4,不压缩， 第一种每次用的时候解压， 第二种体积大但是解压的时候是只解压用到的，并且第二种无法再进行压缩， 第三种压缩完和lzma体积相同. 这三种压缩方式的时间测试，load了很多图片，lzma平均时间0.9，lz4平均时间0.059， 不压缩平均时间0.044，所以呢， 第一种压缩是绝壁不能用的,第二种和第三种各有优劣吧，第二种其实可以做成不移动到外部存储，这样也就省去解压，并且占得本地内存也少了，第三种就得进游戏解压,但是速度是最快的,包是最小的.

打出的ab大小：不压缩 1.14g，lz4680m

打包的时候准备工作：

1. 写一套依赖查找代码，这个每个项目都不一样，具体问题具体分析
2. 在GraphicsSettings中加入所有内置的shader（自己写工具添加，不然麻烦死）
3. Shader stripping中选择auto，会导致光照贴图有问题，改为手选并勾选所对应选项，注意打ab包和出apK包的时候都要勾选，并且勾选这个后必须把老的ab包删掉重新打
4. 还有一个问题是场景不允许和别的东西一起打包，所以在最终阶段要把unity后缀也特殊处理

4.日了狗，坑死我了，图片不能设置成sprite格式，否则依赖打包无效！！！我以为是我写的依赖有问题，美术把图片不知道为什么设成sprite了

1. 场景里面物体勾选static之后，会在场景的bundle里新加一个combine mesh，所以场景的体积会很大， 刚开始以为是fbx打了两份，一顿查，后来发现还真是两份，但是场景里这份是因为静态批处理，所以这部分fbx就可以省略掉
2. 如果单独打fbx的话，由于fbx默认有个内置shader，Default-Material，用的shader是standrad，由于这个shader比较特殊（据传闻是有很多变种），所以无法单独剥离，加到always中也不行，所以写了个脚本，在导入这个资源的时候修改这个材质，正常情况代码是无法修改的，所以必须导入时候修改
3. 同上原理，如果一个材质用了内置的Default图片，那么也会造成包体变大，也需要替换成项目中已有的
4. Prefab中的默认材质也需要干掉
5. 如果shader加到always中，打ab的时候就不会吧这个shader打到ab中，否则，每个shader内都会存一份这个shader，所以内置的shader必须加到always中
6. 脚本不能有missing， 否则比如missing下面再有脚本，这个脚本打assetbundle打不进去
7. 更改always后，需要重新打assetbundle，否则不生效