**AB包**

**什么是AB包？**

AB包全名AssetBundle， 通过将资源分布在不同的AB包中可以最大程度地减少 运行时的内存压力，并且可以有选择地加载内容, 动态下载和加载。

**AB包与Resources的比较**

AB包

1： 资源可以分布在多个包中 2： 储存的位置自定义灵活

3： 压缩的方式灵活

4： 支持后期进行动态更新

**打AB包的时候需要注意什么？**

1：获取文件夹下所有的资源的时候

Resources

所有资源达打成一个打包

必须放在Resources文件夹下 全部压缩成二进制

打包后资源只读无法动态更改

有可能会是

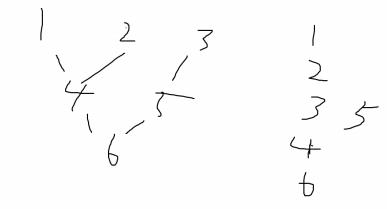
（D:/UnityWork/ab/Assets/Resources\\Capsule.prefab ）这样是获取不到依赖项的

需要更改成（Assets/Resources/Capsule.prefab）

2：不是所有的后缀名文件都可以打包 下面这些就不可以打包 要把这些给去 掉



3：依赖项



把依赖项打一起（ 注意：有的材质球可能会被重复利用 把重复利用的打成一 个包），或者 一个依赖项一个包（注意：有的模型上面会出现名称一样但是后 缀名不一样 如：



) 这样的话还是打一个包会报错 可以把名称一样后缀名不一样的打成一个包 或者 找美术改一下名称

**AB包加载**

通过字典来进行缓存，看看字典里面有没有这个AB包，如果有就让他引用计数+1 如果没有在加载AB包然后放入字典

从AB包加载一个物体的时候 要确保它的依赖项也被加载 如：只加载了一个 Cube没加载依赖项的话会造成材质丢失

**卸载AB包**

加载的资源应该在不再需要时及时卸载 ，以释放内存。使用 AssetBundle.Unload方法可以卸载AB包及其加载的资产。

通过引用计数来判断他是否被人依赖 ，如果引用计数为零的情况下说明该对象没有被 依赖 所有就卸载掉

AssetBundle（false） 卸载指定的AssetBundle，但不会卸载该AssetBundle的依赖项

AssetBundle（true） 卸载指定的AssetBundle以及从该AssetBundle中加载的所有对象，包括它们的依赖项。

1 ：你刚开始第一次就进入项目的时候 ，你需要把 ， S目录给拷到P目录 ， S目录就是一 般放什么?

S目录放所有的资源 刚进入游戏 直接把S目录Copy到P

S目录放版本文件 ，版本文件对比 去服务器下载资源 2:循环依赖

1. ->B B-->C C-->A

在加载资源时，可以使用引用计数的方式来管理资源的加载和卸载。当资源被其他资源引用时，引用计数加1，当资源不再被引用时，引用计数减1。这样可以确保在循环依赖的情况下，资源能够正确地加载和卸载。

死环，在加载时不是有计数嘛，你加载时会判断是否加载过

3:当运行的时候因为内存压力需要卸载一些AB包的时候， 在什么时候可以卸载它，什么时机卸载它

如果当内存达到一个值需要卸载一些AB包的情况下通常我们会卸载那些不太重要的AB包以释放内存空间，使用频率较低，对游戏的核心玩法和贡献较小的AB包

**4：AssetBundle.Unload（true和false）的区别是什么**

AssetBundle.Unload（false）：（不会卸载依赖项）AB包内的资源将保留在内存中。这意味着即使AB包被卸载，任何已经实例化的对象（如通过Instantiate方法创建的游戏对象）仍然有效，并且已加载的资源如纹理、音频文件等仍然会占用内存。这种方式适用于资源仍需使用，但不再需要从AB包中加载新资源的场景。

AssetBundle.Unload（true）：（会卸载依赖项） AB包和其包含的所有资源都将从内存中卸载，这包括任何实例化的对象和资源。这种方式彻底清除了AB包对内存的占用。这种方式适用于确信不再需要AB包中的任何内容，且需要释放尽可能多内存的情`````况。

**5：AB包的打包策略**

1：分类打包---相同的资源打到一个包里面

（场景比较多的情况下）

2：场景打包---一个场景一个包

通过引用计数来判断是否打成公共包

6：如果我AB包加载了一个预制体，预制体依赖于材质球A 我又AB包加载了一个材质球B 把A替换成B 我把预制体给卸载掉那么材质球B怎么办