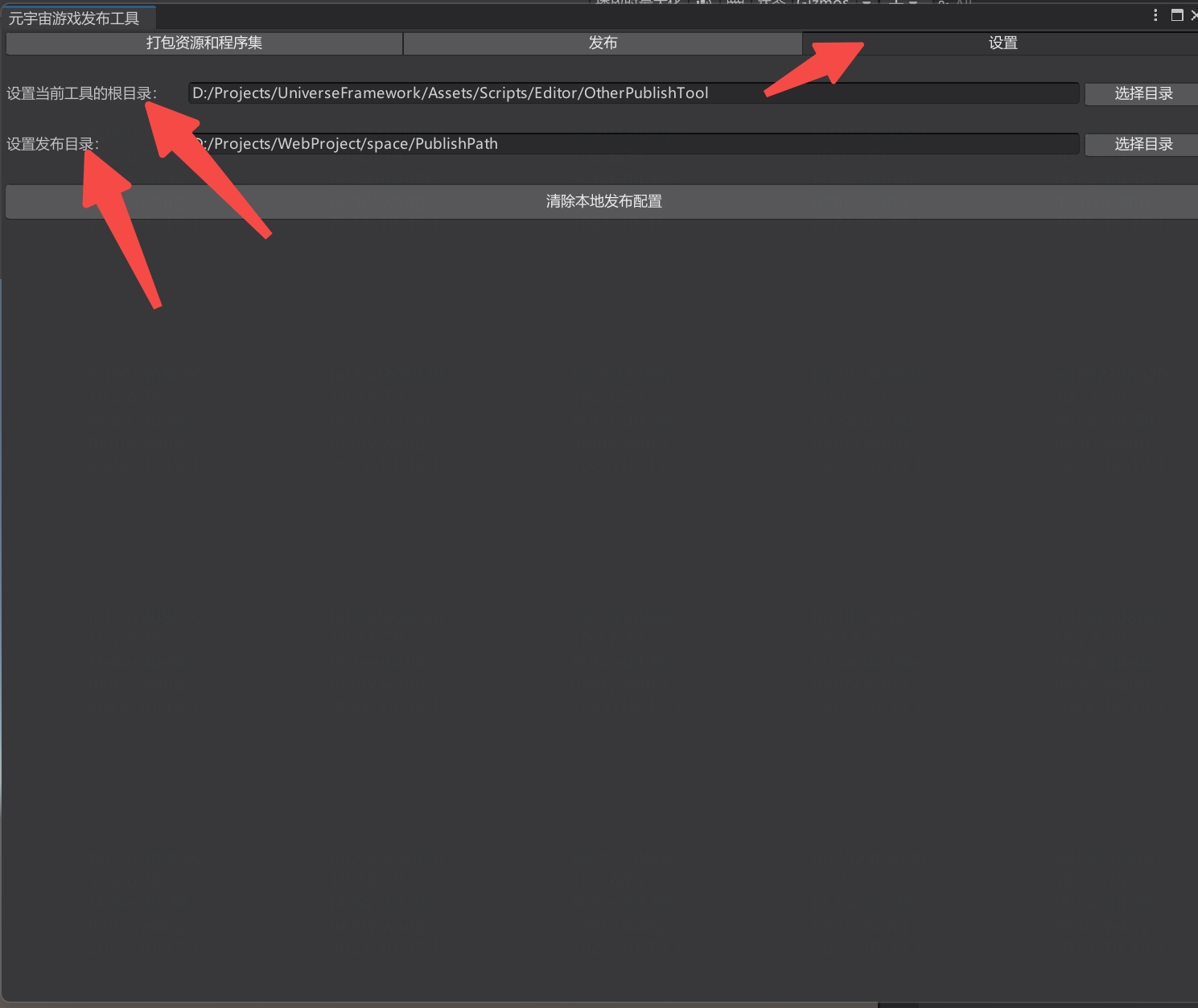
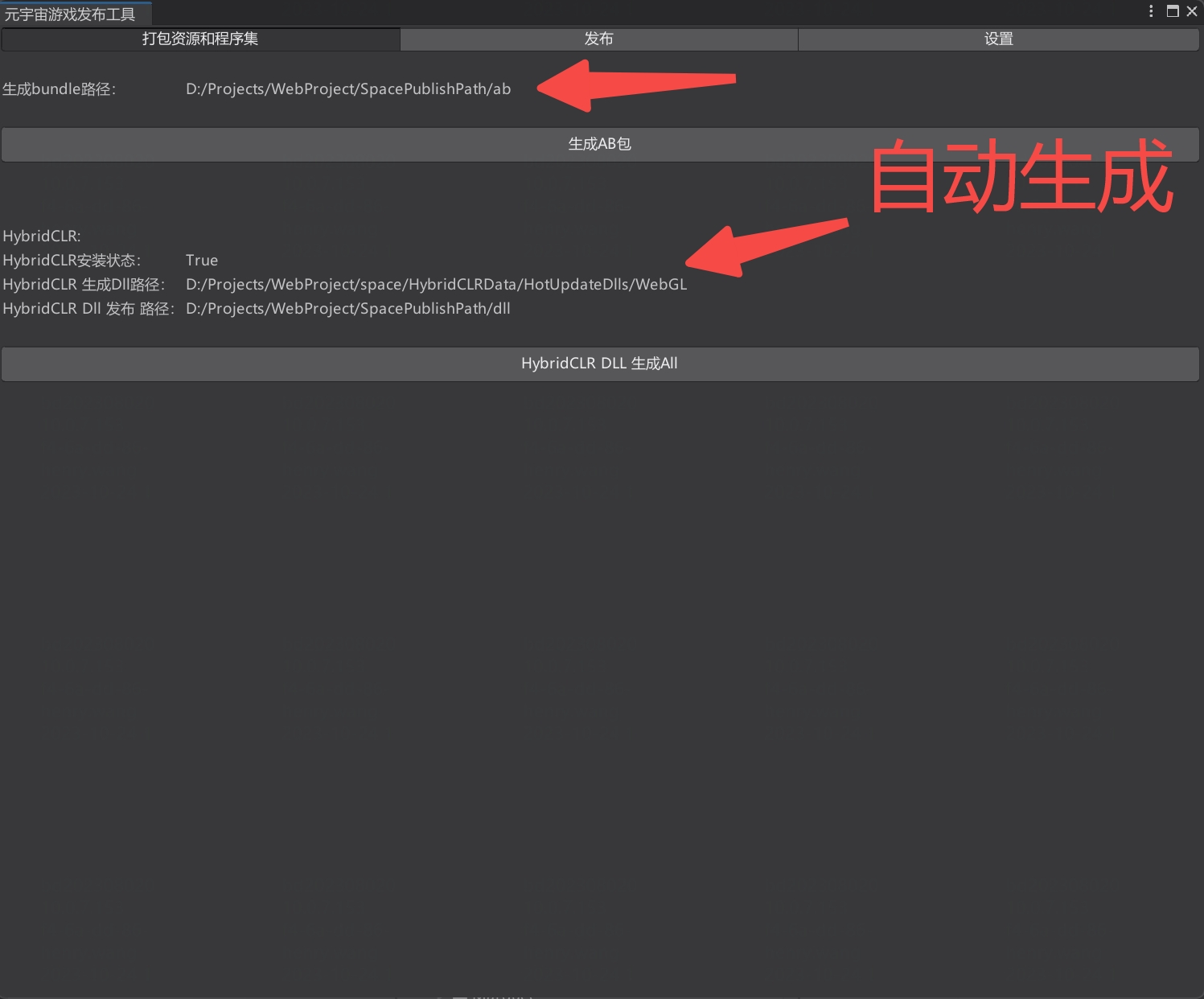
现有项目通过此工具和调整项目资源脚本目录，可直接兼容新的元宇宙框架。

具体使用：

第一步：配置当前工具的目录， 配置发布项目的目录

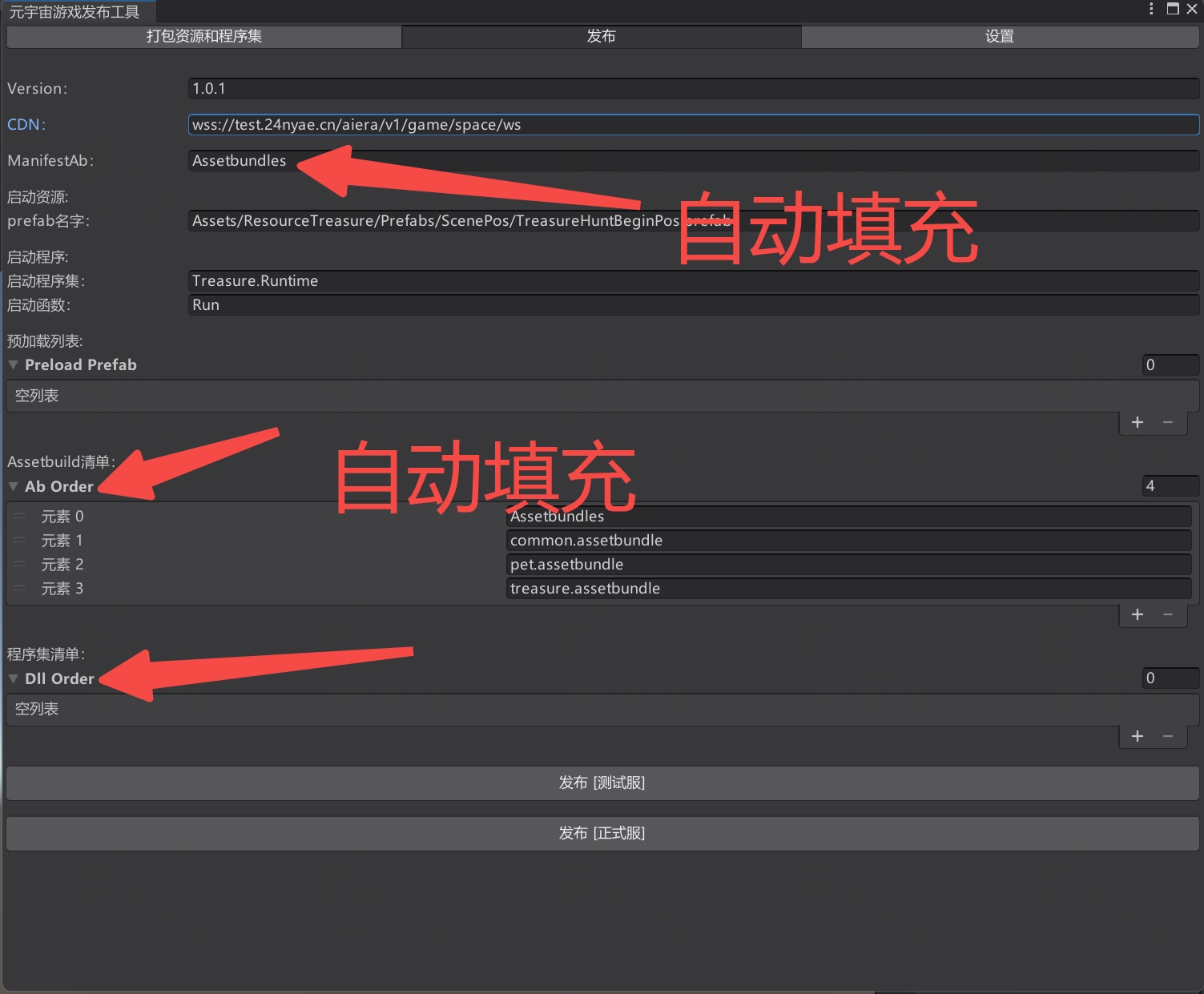


第二步：点击 “打包资源和程序集页签” 自动生成AssetBundle的目录，HybridCLR目录



第三步：点击 “生成AB包” ， 点击“HybridCLR DLL 生成”，等待生成完毕。

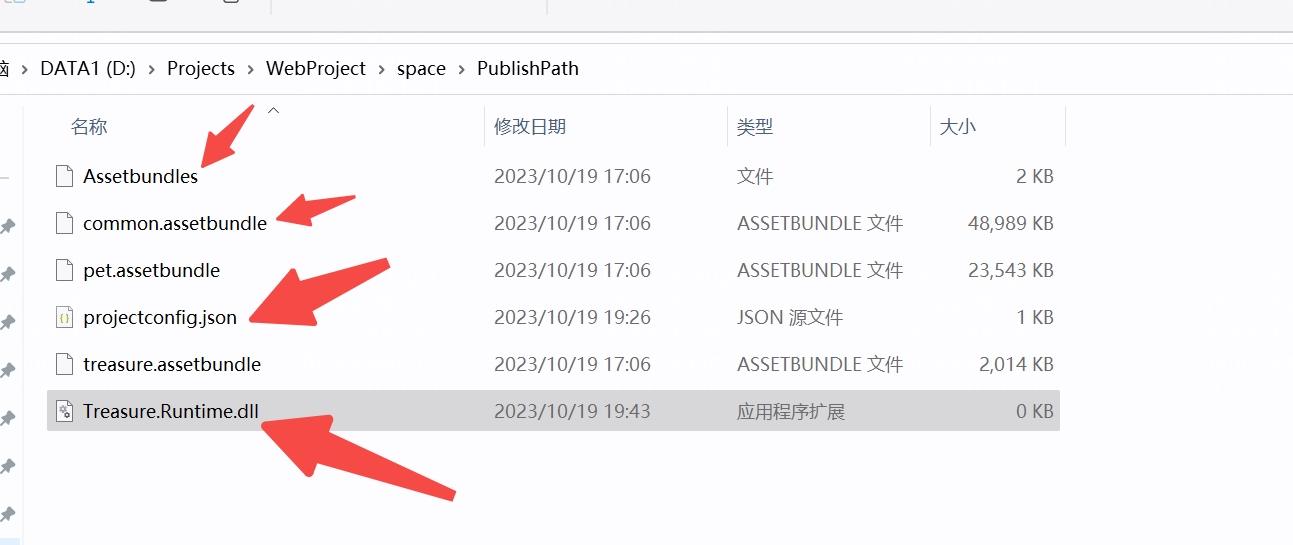
第四步：点击发布页签， 配置版本号，CDN地址，预加载资源列表，自动填充的无需填写。需要预加载时把prefab名字添加到PreloadPrefab列表



第五步：发布 [测试服] 或者 [正式服]

发布成功之后，内容大致如下：

包含Manifest不带后缀的AB包，资源AB包，项目配置的json，代码程序集



注意:目前的发布功能现在还是只能够导出文件，CDN地址还未确定，还未接入。发布正式服还不支持，后续添加。

一些编辑器下使用的脚本，整体添加#if UNITY\_EDITOR

# 关于业务层代码与资源的模块化管理框架

## 流程图：

最新版本流程和以下逻辑流程有部分调整，主要是不再强制各个h5项目并入到统一的一个工程， 还是保持各自工程的独立性，但是可以接入这套框架，并且可以选择是否要把代码程序集dll热更功能开启。

暂时无法在飞书文档外展示此内容

## Demo工程universeframework：

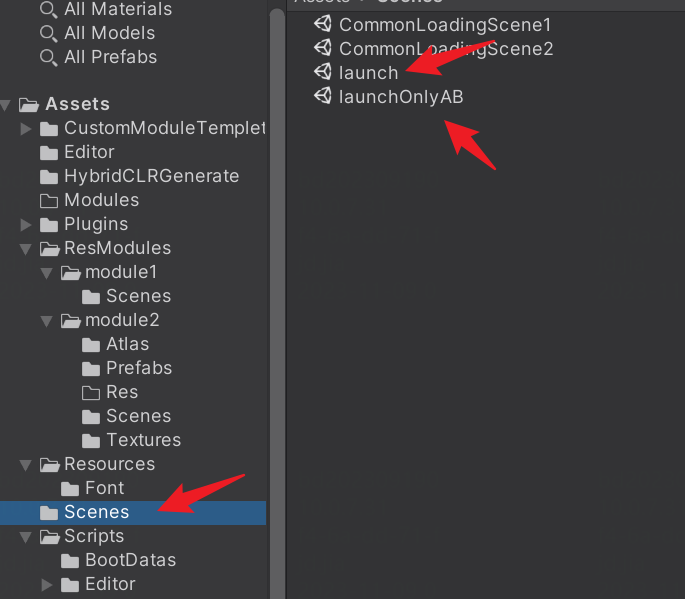
可以先从这个demo工程（git地址：https://gitlab.aiera.tech/jd.jia/universeframework），了解基于HybridCLR+Webgl的代码热更与AssetBundle资源包热更相关流程，此工程基于"土地空间转3D"需求开发过程中衍生而来。

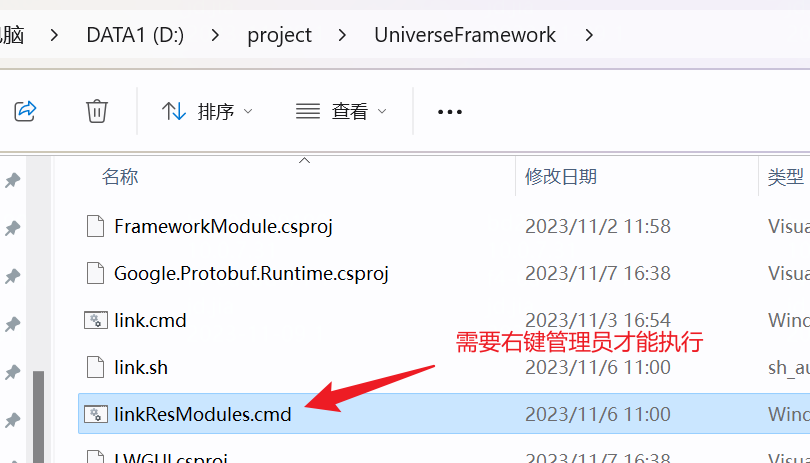
核心目的是优化内存占用，减少母包体积，加强模块化管理，将业务层的脚本代码与美术资源按照模块划分单独打包成独立文件，放入web server，等到用户需要进入到相应模块时，才去加载这些独立的模块文件，开辟内存空间，去往下个模块时，会把当前模块的程序集dll以及美术资源ab开辟的内存全部回收，内存空间腾出来留给下个模块使用

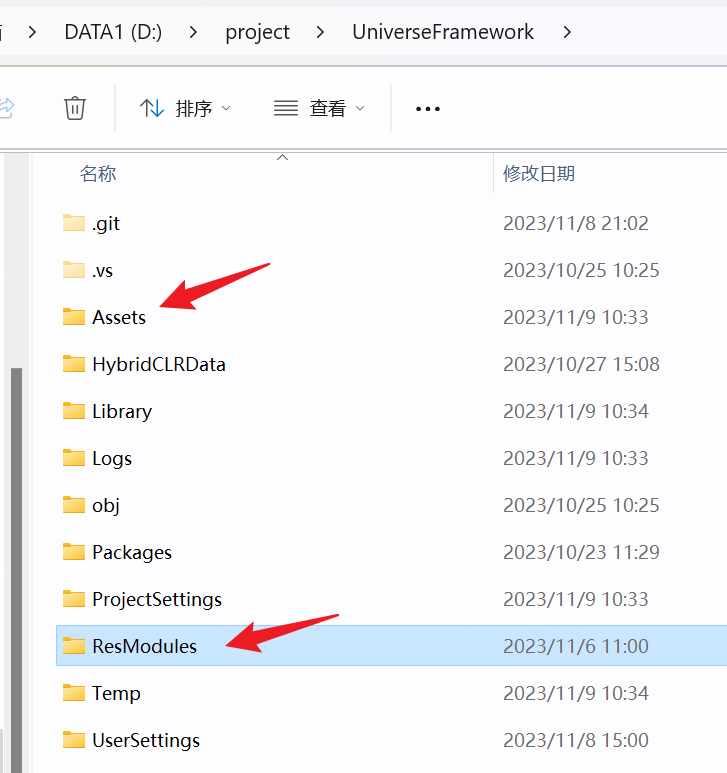
### launchOnlyAB场景（space工程开发相关着重看这个），

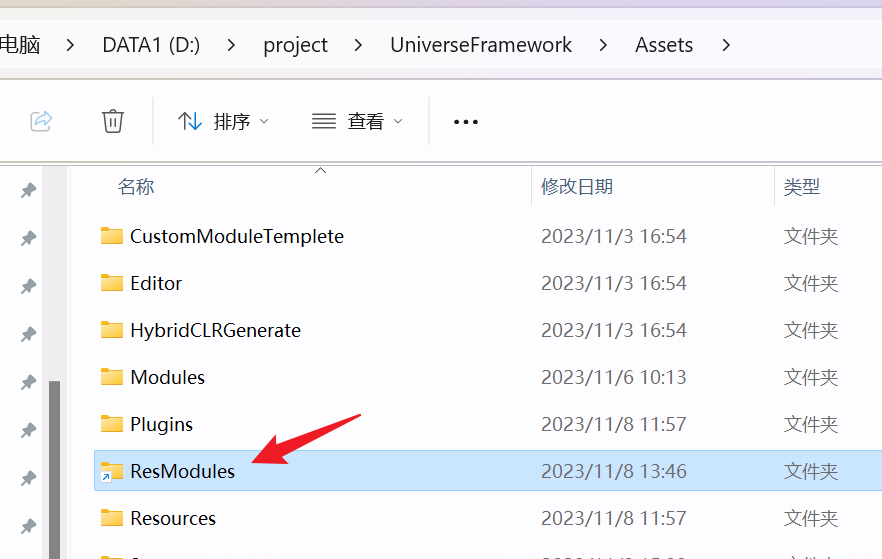
会演示ab包热更过程（代码在母包不分程序集不编译dll）

通过下图中的cmd批处理，把位于Assets目录外部的资源目录link到Assets内部，用于开发测试，发布 ab包之后，可删除母包内所有模块link目录进行测试，（不删也能侧但要注意不进母包的资源不要放入Resources目录）。









### launch场景，

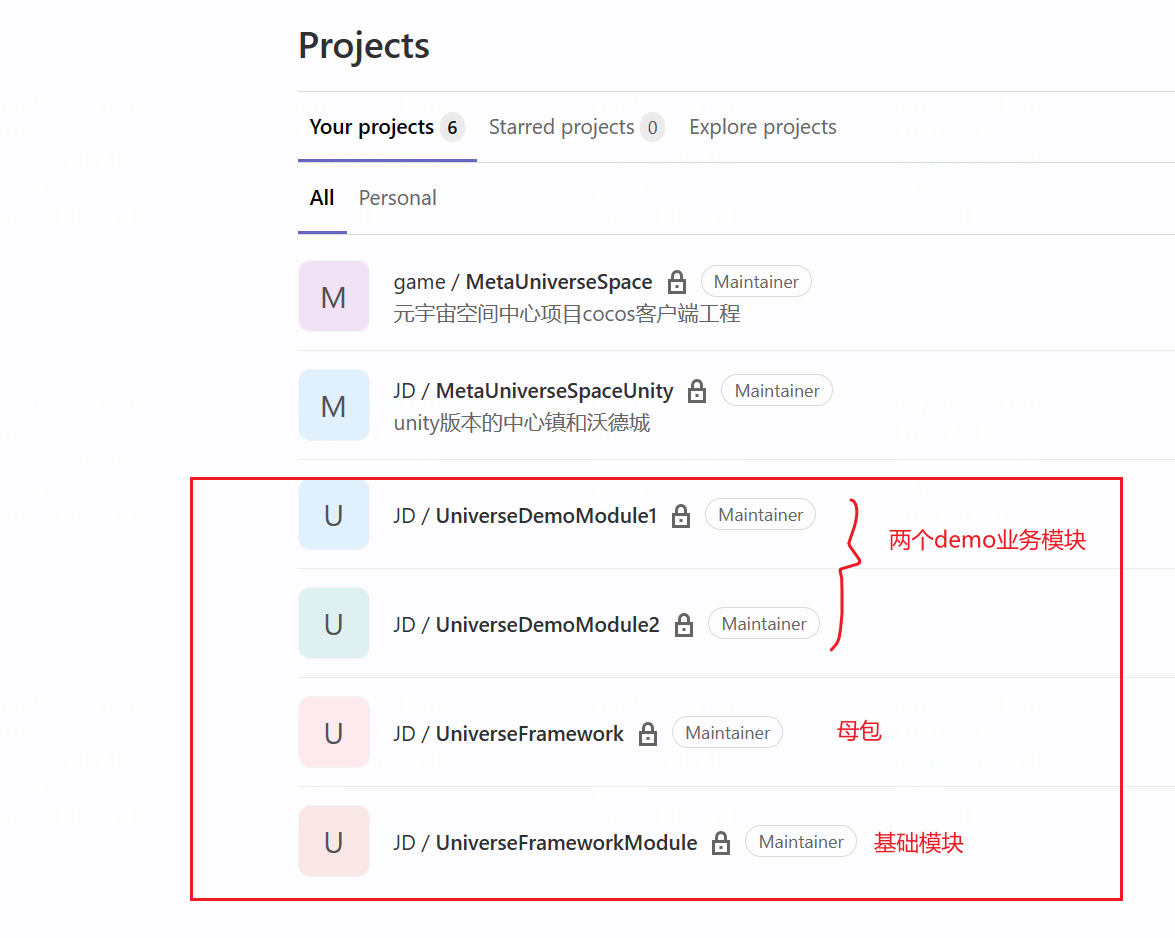
会演示dll+ab包热更过程；

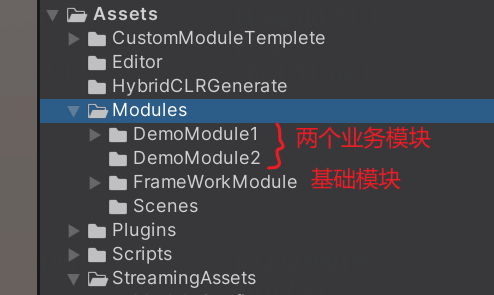
为测试方便，彻底，所有模块相关的代码以及资源都位于Assets目录之外，launch场景测试的包含代码与资源的模块，都是一个个的独立git仓库：

https://gitlab.aiera.tech/jd.jia/universedemomodule1

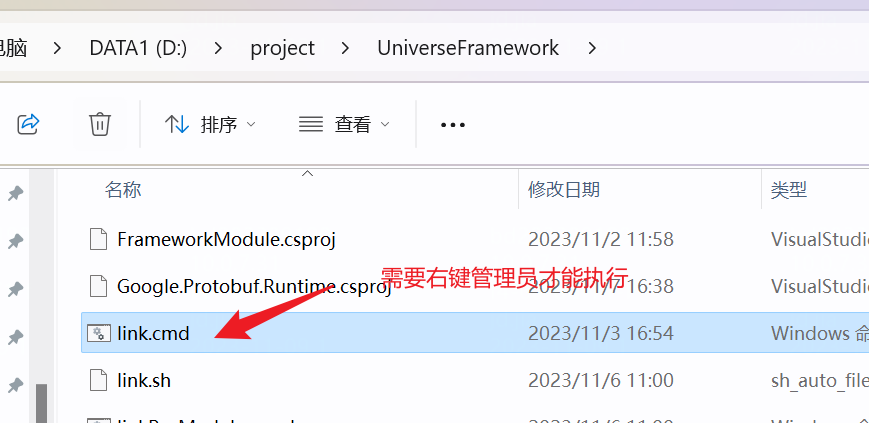
https://gitlab.aiera.tech/jd.jia/universedemomodule2

https://gitlab.aiera.tech/jd.jia/universeframework



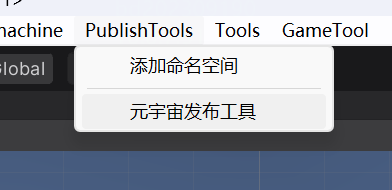


这些模块通过以下这个批处理自动mklink到母工程内，用于开发测试，编译dll和ab包之后，可删除母包内所有模块link目录进行测试，（不删也能侧但要注意不进母包的资源不要放入Resources目录）。



## Space工程接入了哪些：

space工程现已接入其中的AssetBbundle资源包管理流程，按照前面提到的”模块化"的概念，客厅、狗窝、救助站、挖宝就是现有的四个模块，可以使用unity editor顶部的发布工具（见下图），单独针对一个模块进行ab打包，每个模块打包后最终产出的文件包含：



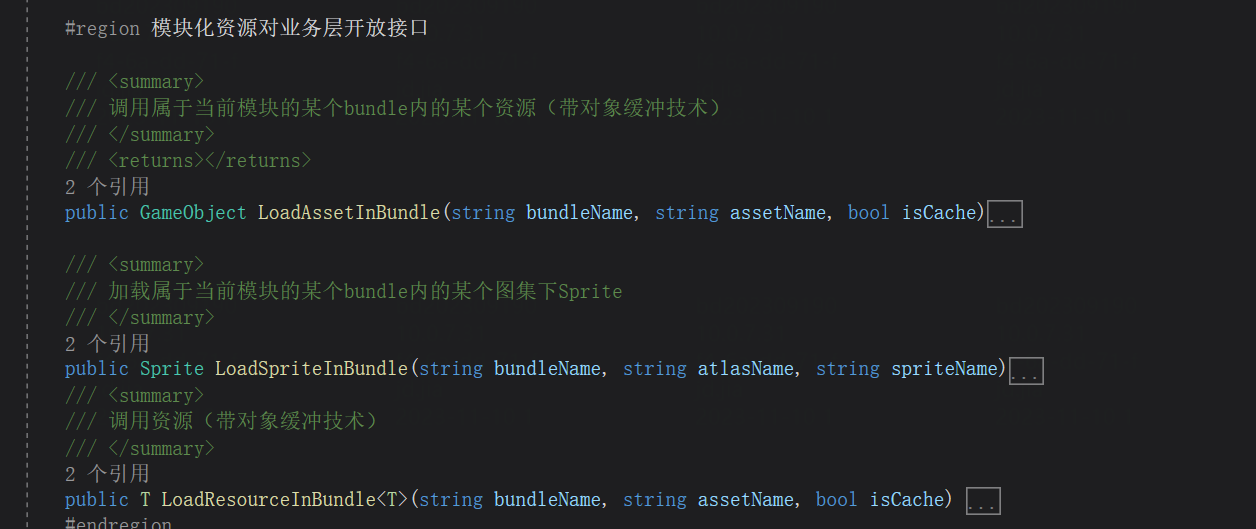
一个叫做config.json的配置文件，包含所有ab包，模块版本号，以及启动场景等参数

.assetbundle文件

## 相关对外接口：

### ResourcesMgr:

提供三个接口，



#### LoadAssetInBundle

获取ab包内的prefab等asset资源，调用此接口时可通过isCache参数指定是否读取到之后存入一个hashtable内存池，下次再次获取时直接从内存缓存里拿

注意：离开当前模块时，内部接口会把这个hashtable内存池清空。

#### LoadSpriteInBundle

获取图集里的sprite

#### LoadResourceInBundle<T>

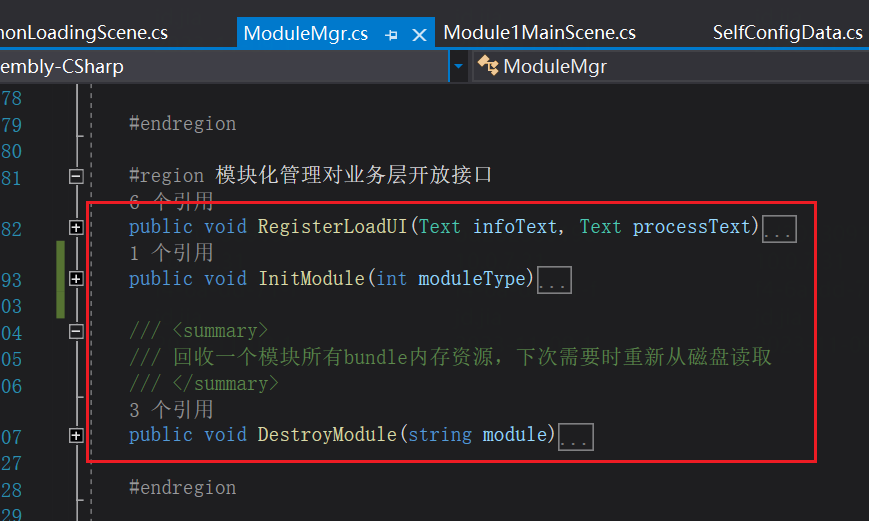
根据泛型获取想要的资源

以上两个接口与现有的老接口对应。在module2模块演示了bundle场景的启动，module2模块内部的Module2MainScene.cs这个测试脚本演示了bundle内prefab对象的获取，以及spriteAtlas图集内sprite的获取。

### ModuleMgr：

各业务层开发基本上不需要调用此manager的接口，目前在PetdensLoading脚本、SceneLoadManager脚本的场景初始化和加载流程中已经接入ModuleMgr。

对外开放了三个接口在ModuleMgr这个单例脚本：



#### RegisterLoadUI

注册下载过程中的loading页面内的下载进度和下载文件信息文本组件用

#### InitModule

当进入一个模块时，此接口内部流程会走完与当前模块相关的所有远程配置拉取解析，版本号对比，本地bundle文件检测与读取，等全部流程，全部流程完成之后，会调用SceneLoadManager这个单例的以下两个接口：

LoadBundleSceneWhenModuleReady 加载位于bundle内的新场景

LoadSceneWhenModuleReady 加载位于母包内的现有老场景

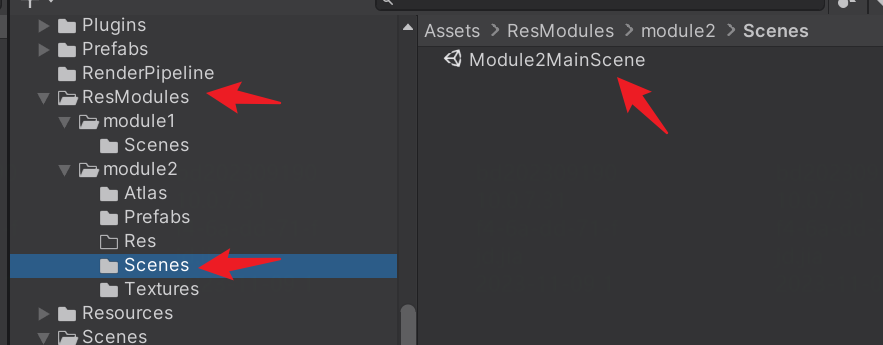
#### DestroyModule

回收当前模块全部bundle资源

注意：以上两个manager管理类，都在universeframework和space工程走通，不同的是，universeframework工程没有单独拆出来一个ResourcesMgr，基本上逻辑代码全写在了BootOnlyAB.cs这个脚本内，space工程的这两个manager类基于BootOnleyAB拆分而来。

## 操作步骤演示：

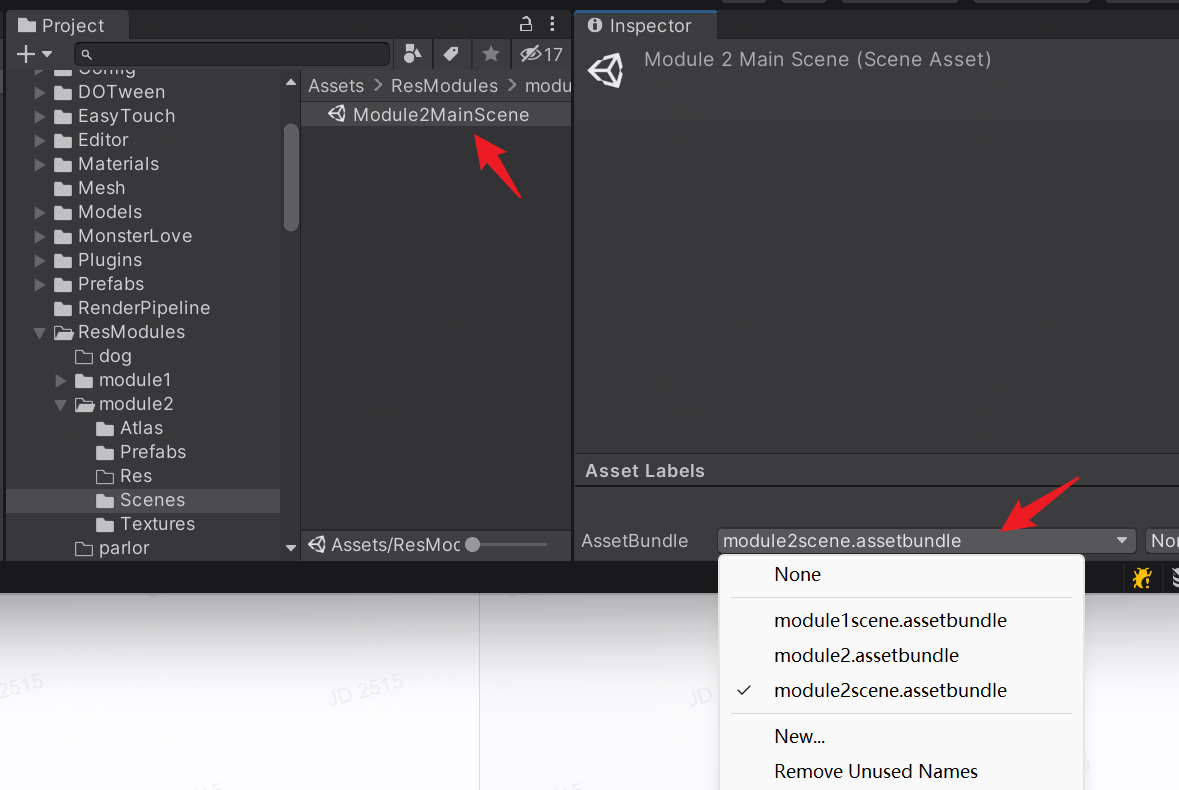
space工程内，单独新建了两个测试用模块，这里为了减少麻烦没有像demo工程那样把资源放到Assets 目录之外,可以理解为是与现有的四个模块（客厅、狗窝、救助站、挖宝）平级的新模块，其中module2的这个场景演示了上面ResourcesMgr接口的调用方法

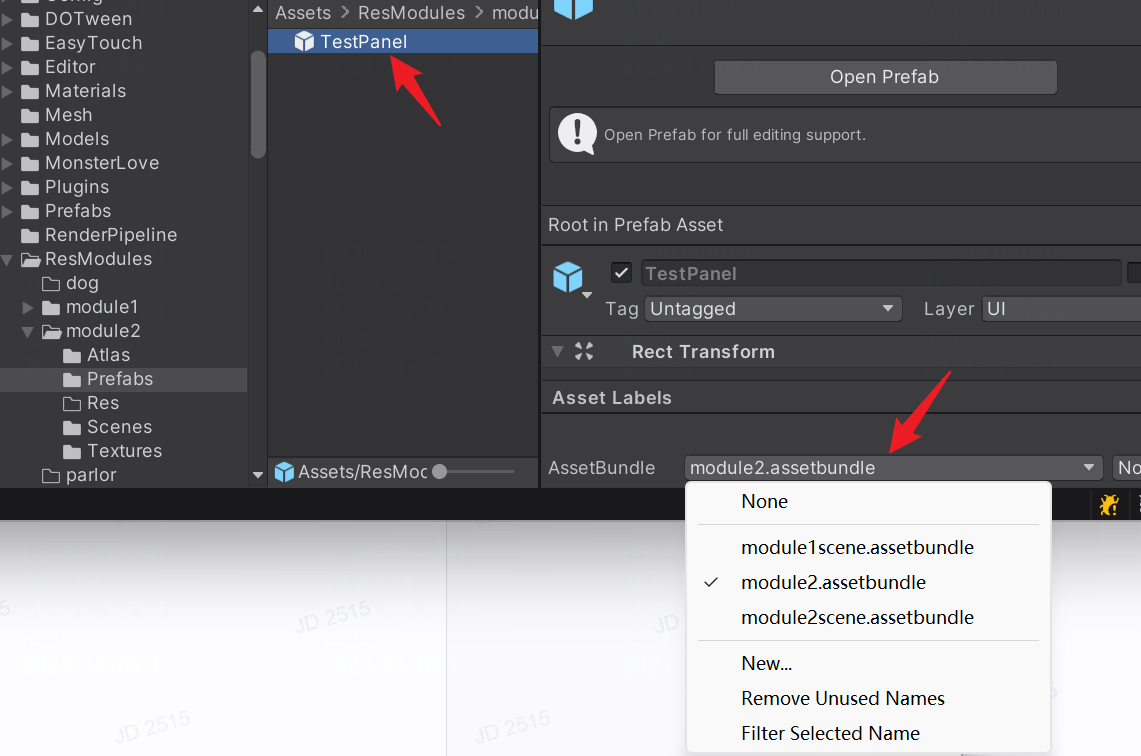


Assets目录下，有个ResModules目录，之后开发过程中，所有不希望进母包，希望走assetbundle流程的资源都放入这个 ResModuels目录内

需要进ab包的资源，在inspector面板右下角选择或者创建相应的bundle归属

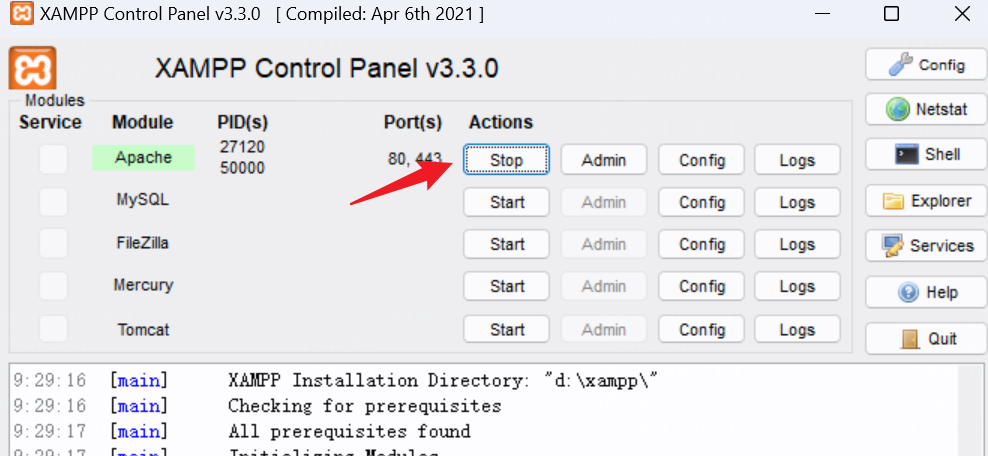
.unity后缀的场景文件需要单独打包



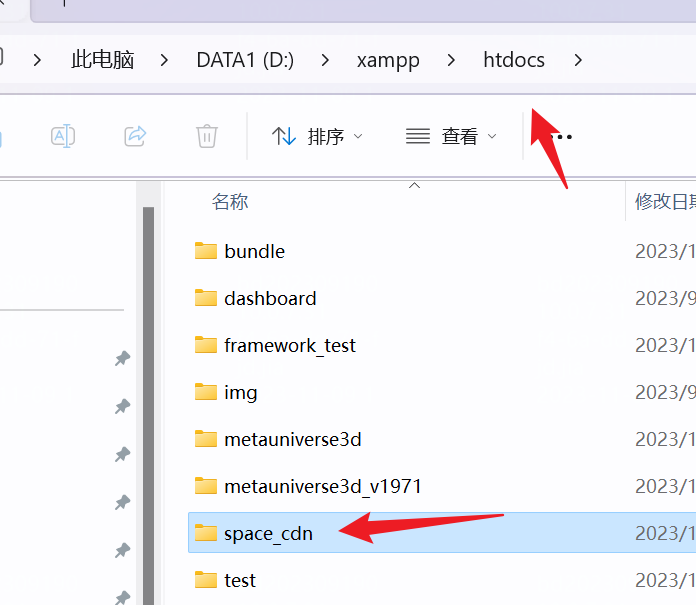


安装一个本地web server服务器，下载下面的xampp安装包，安装号之后，开启apache服务，

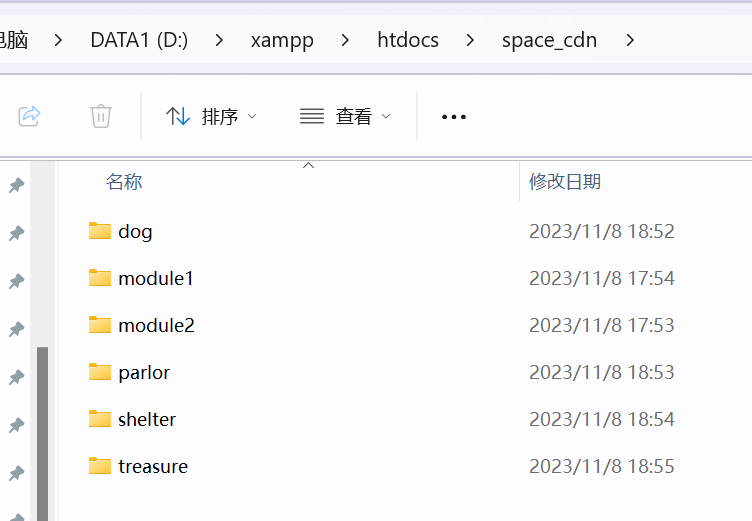
暂时无法在飞书文档外展示此内容



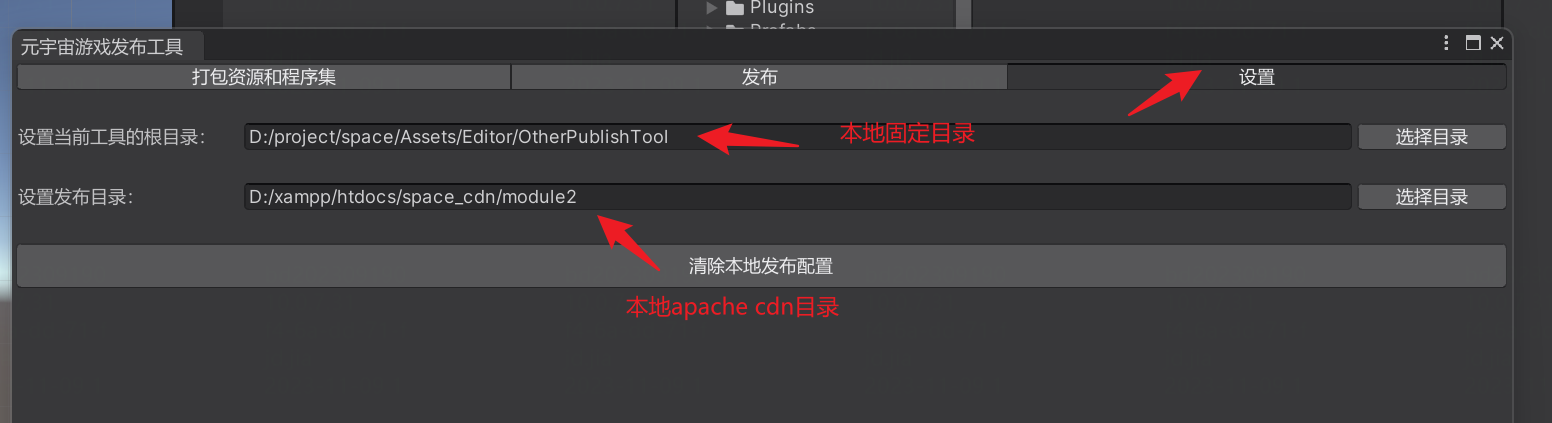
安装好之后，在本地的安装目录内建立space工程的远程资源目录，



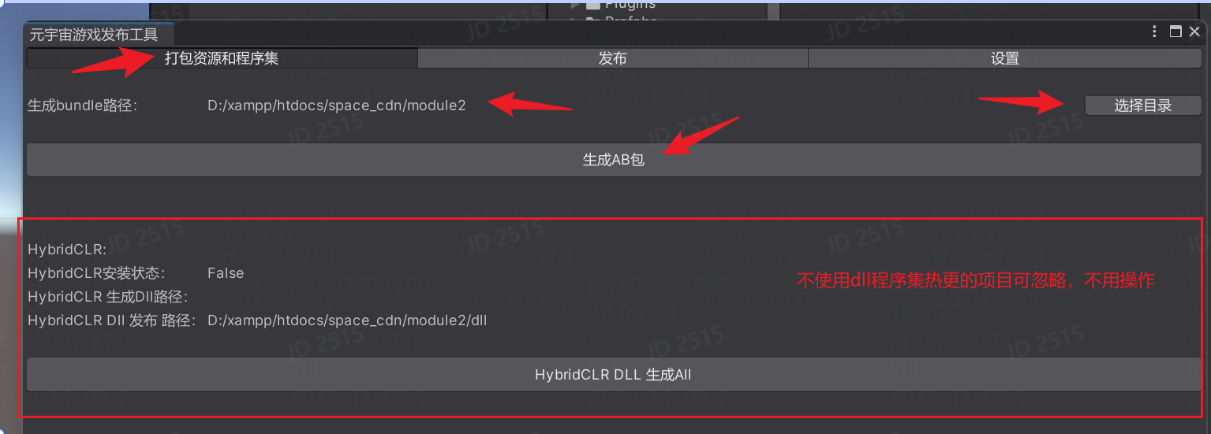
在目录内建立相应的模块目录，这些模块目录也会被下面步骤作为发布目录



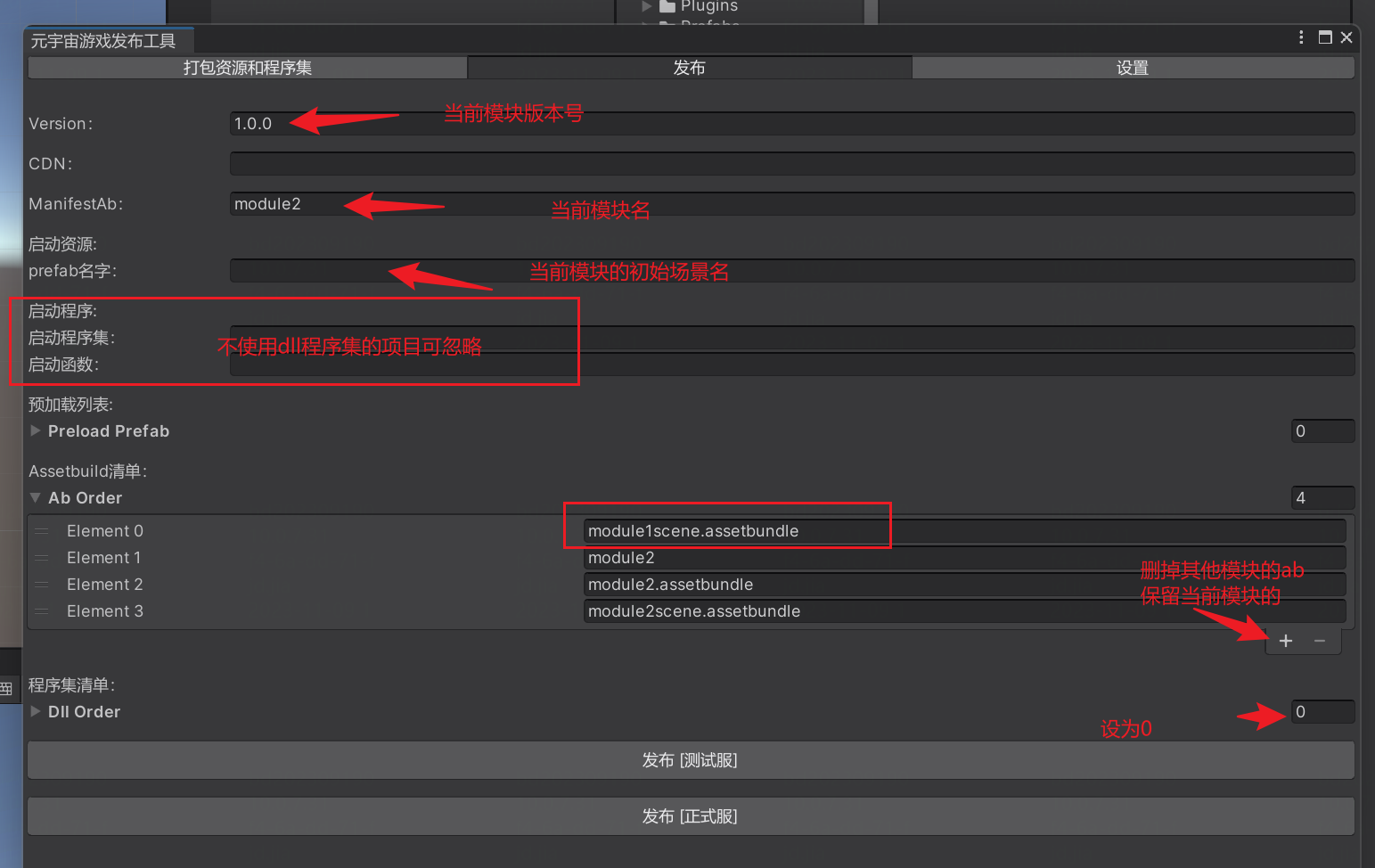
选择要发布的当前模块目录



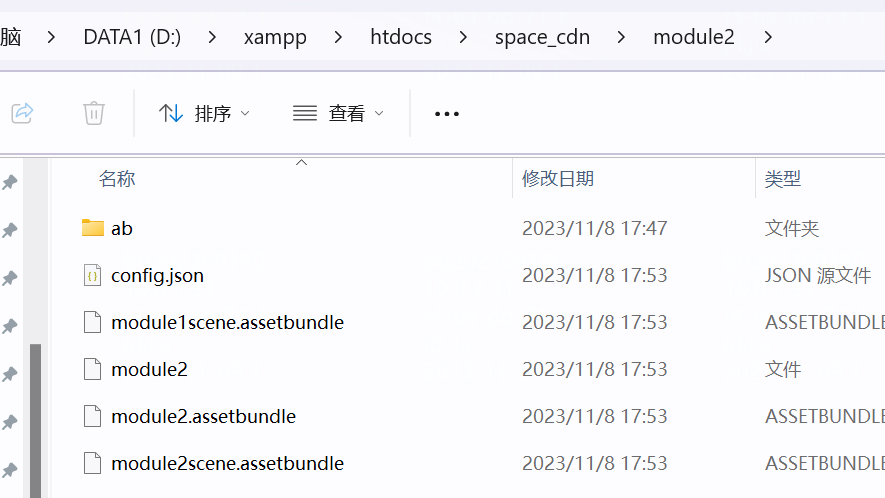
再次选择模块目录并生成 ab包



关闭发布工具面板，再次打开，

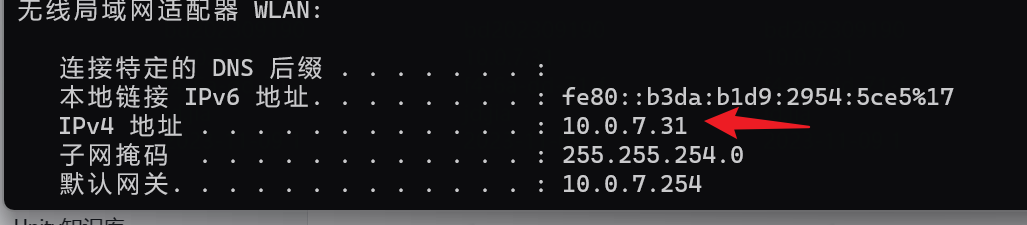


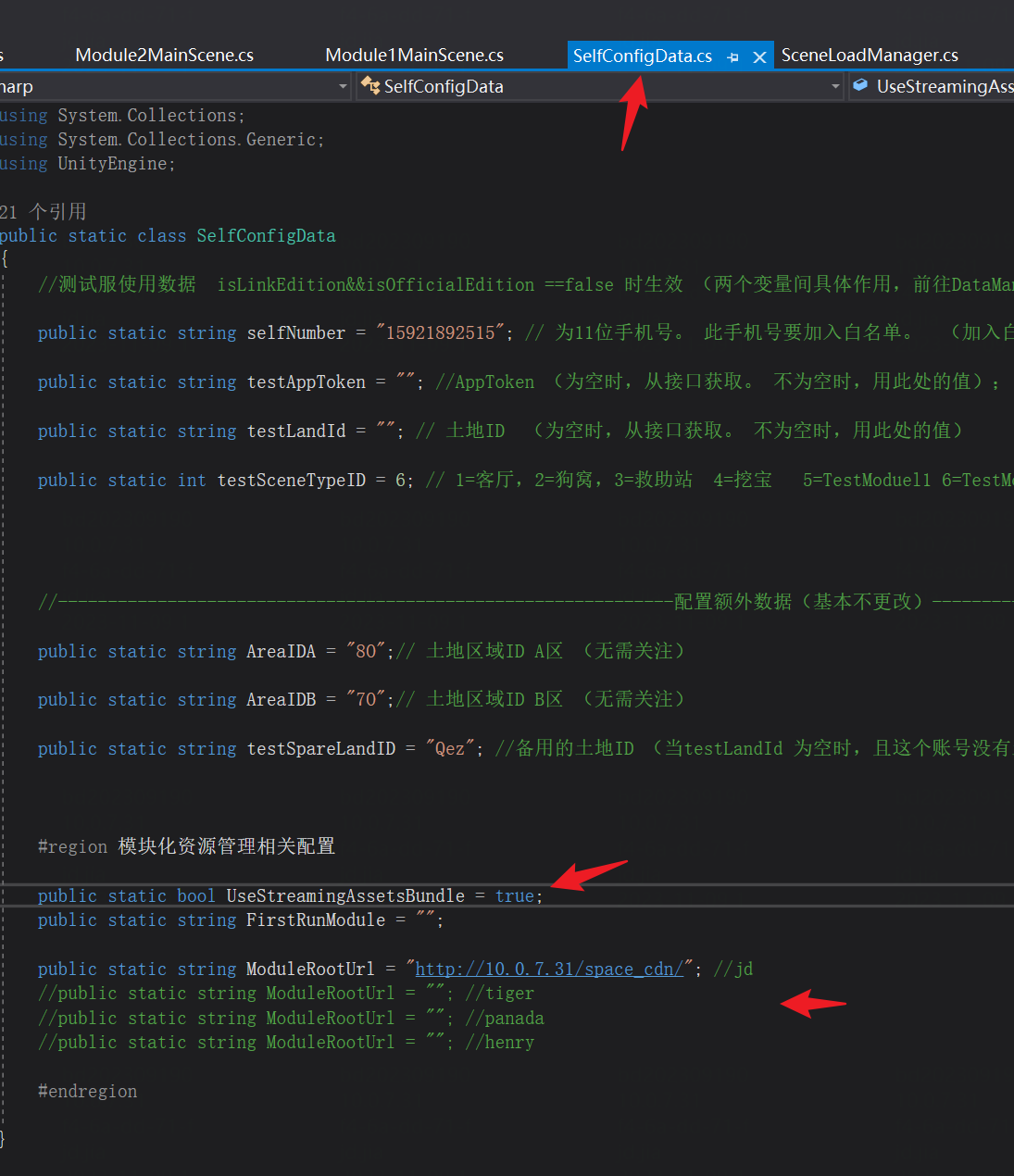
最后点击发布【测试服】，本地apache目录就有了下图的文件：



同时这些文件也会被自动拷贝到项目内StreamingAssets内

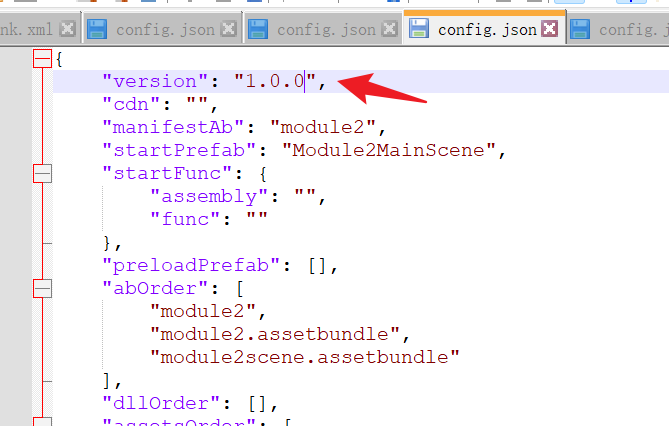
命令行输入ipconfig命令查询自己局域网ip:





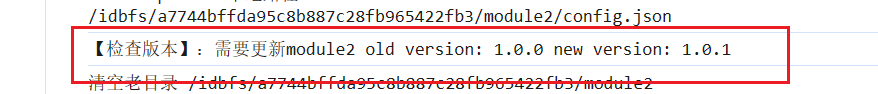
UseStreamingAssetsBundle变量用来设置unity\_editor开发环境下，是直接读取本地工程StreamingAssets 内ab包，还是通过http下载自己本地的apache目录，前者不走http request，后者模拟的正式http request流程.

UseStreamingAssetsBundle设为false，程序运行，加载一个模块的场景之后，去往下一个模块，然后修改本地apache里上一个模块的配置文件里的版本号：



保存，然后程序再返回上一个模块，此时就会因为版本号变更而重新加载这个模块的最新资源。

会看到这样一行打印：



以下2个步骤，是项目build出webgl之后，脱离于hotdog app而单独在浏览器里测试的流程，在universeframework demo工程走通了，但是space工程还是需要打包发布到真机测试，原因是space工程的各种内部参数以及从app侧获取的参数还未针对脱离app而在浏览器单独测试而适配过。

母工程发布到cdn下，D:\xampp\htdocs\space\_cdn

这一步把母工程与所有资源模块放到了一起，具体线上生产环境之后可能会有调整。

可以在电脑浏览器或者链接了公司 wifi的手机上打开具备自己局域网ip地址的工程测试了。

比如我本地的这个universeframework工程的发布地址，大家也都可以访问：http://10.0.7.31/framework\_test/