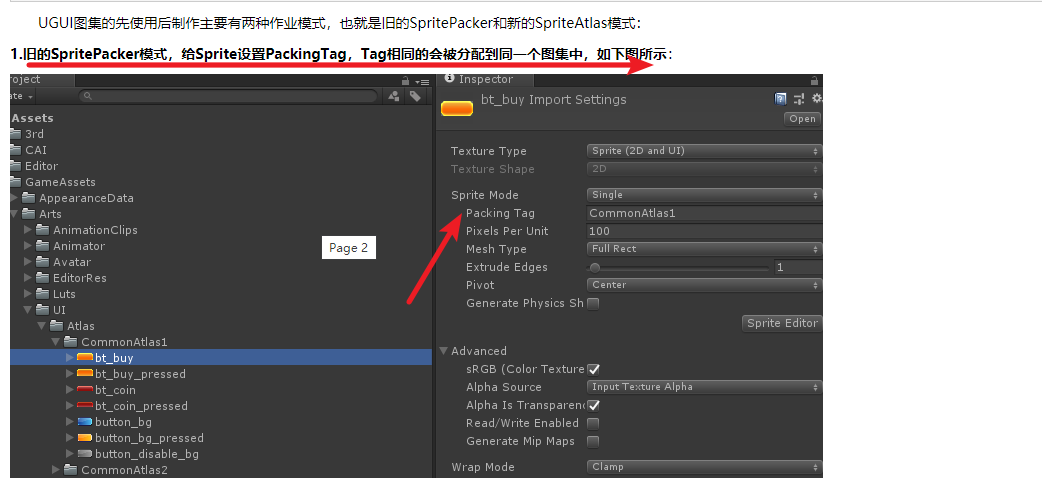
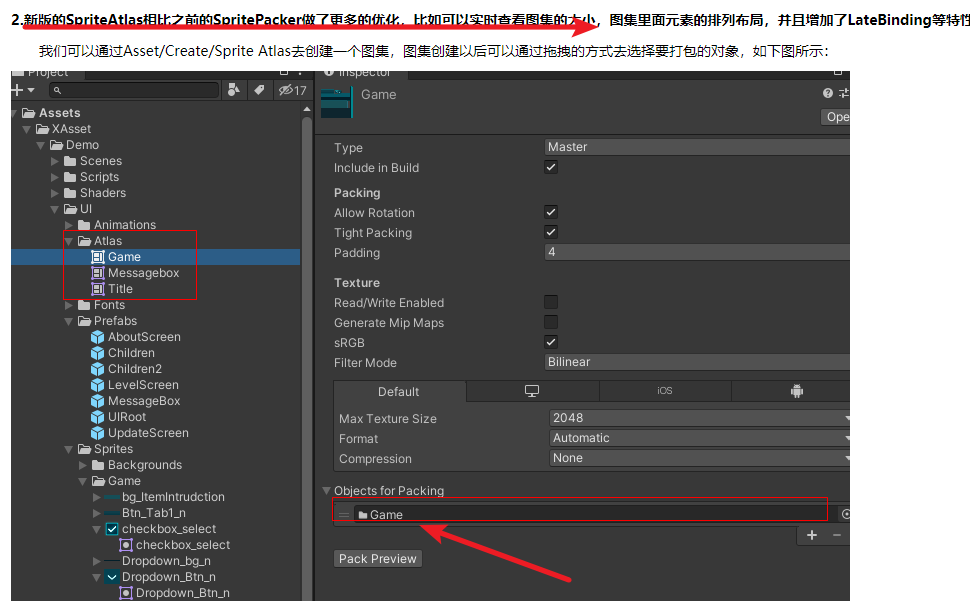
<https://www.cnblogs.com/msxh/p/14194756.html>

NGUI图集的图元时，我们需要先加载图集，然后再从这个图集中获取单个图元

UGUI 不需要关心我这个图元属于哪一张图集，属于哪一个AssetBundle，直接以散图的形式去使用、去获取就可以了

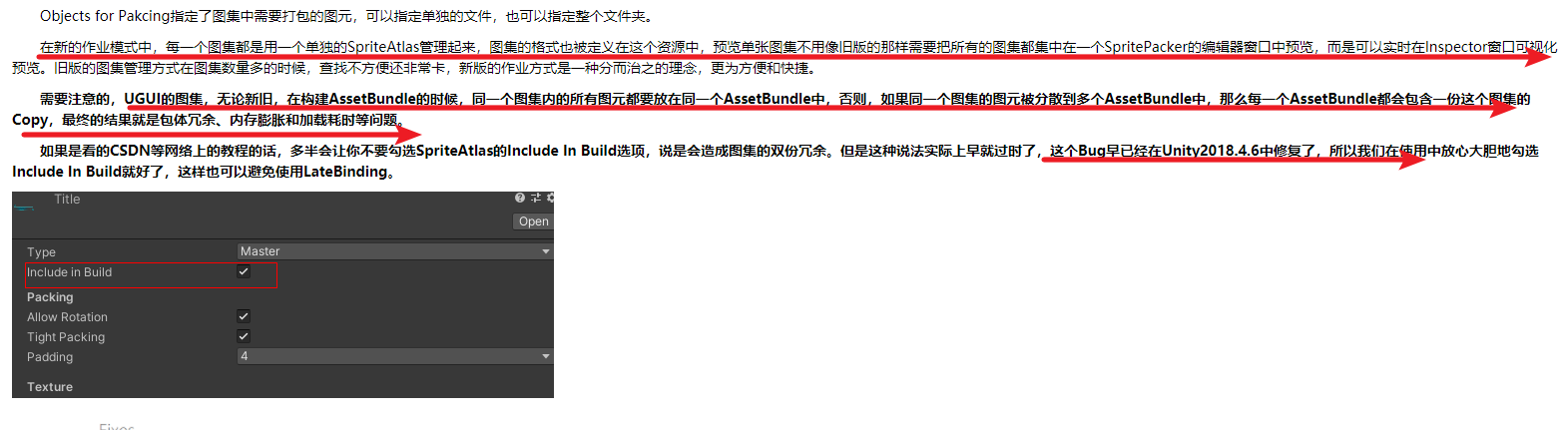




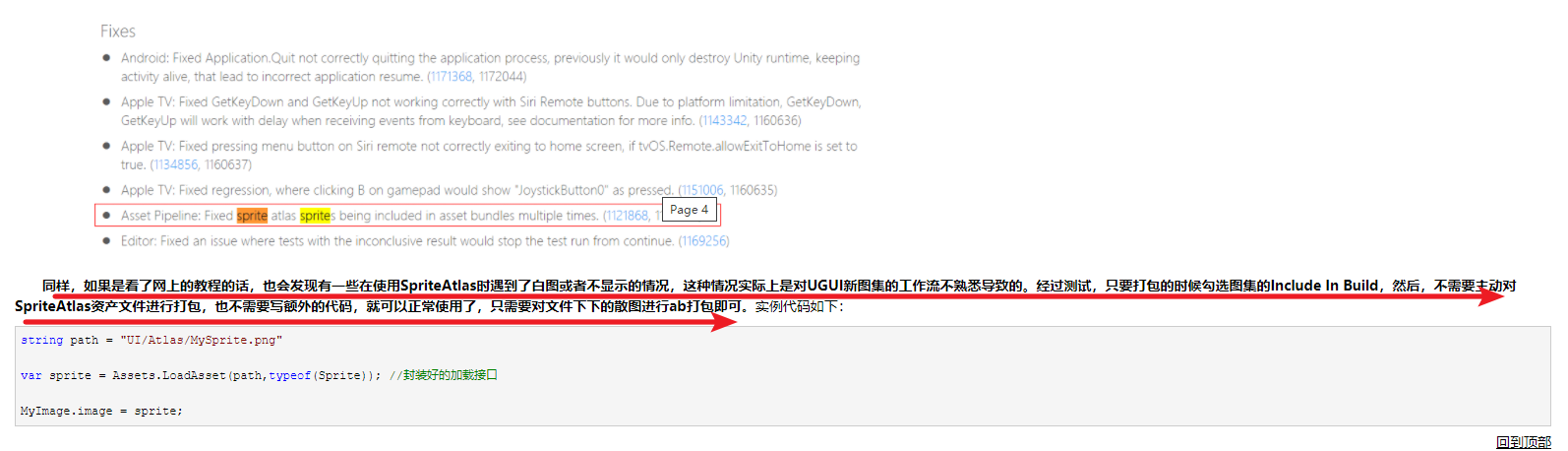


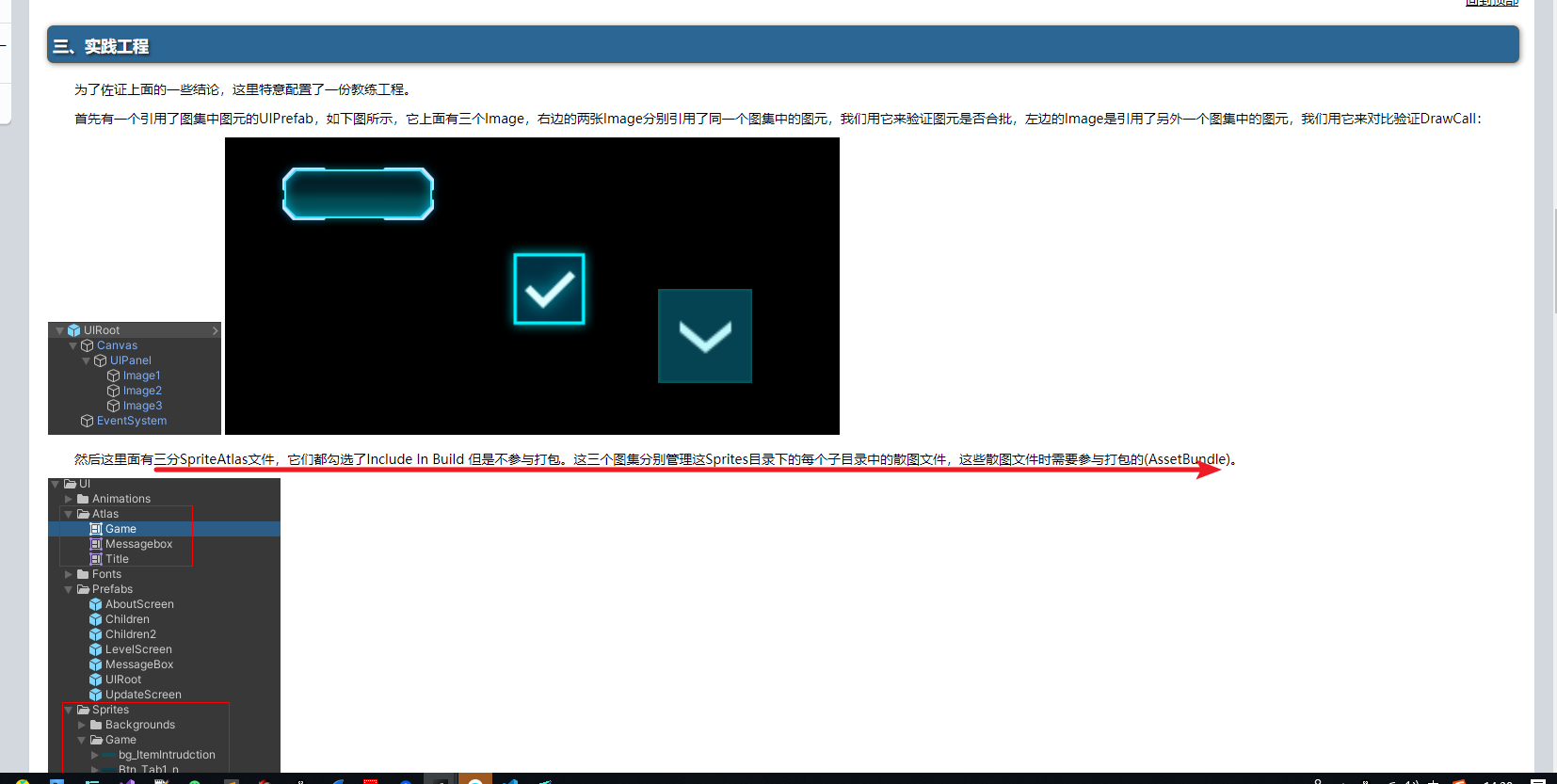
**在构建AssetBundle的时候，同一个图集内的所有图元都要放在同一个AssetBundle中，否则，如果同一个图集的图元被分散到多个AssetBundle中，那么每一个AssetBundle都会包含一份这个图集的Copy，最终的结果就是包体冗余、内存膨胀和加载耗时等问题**

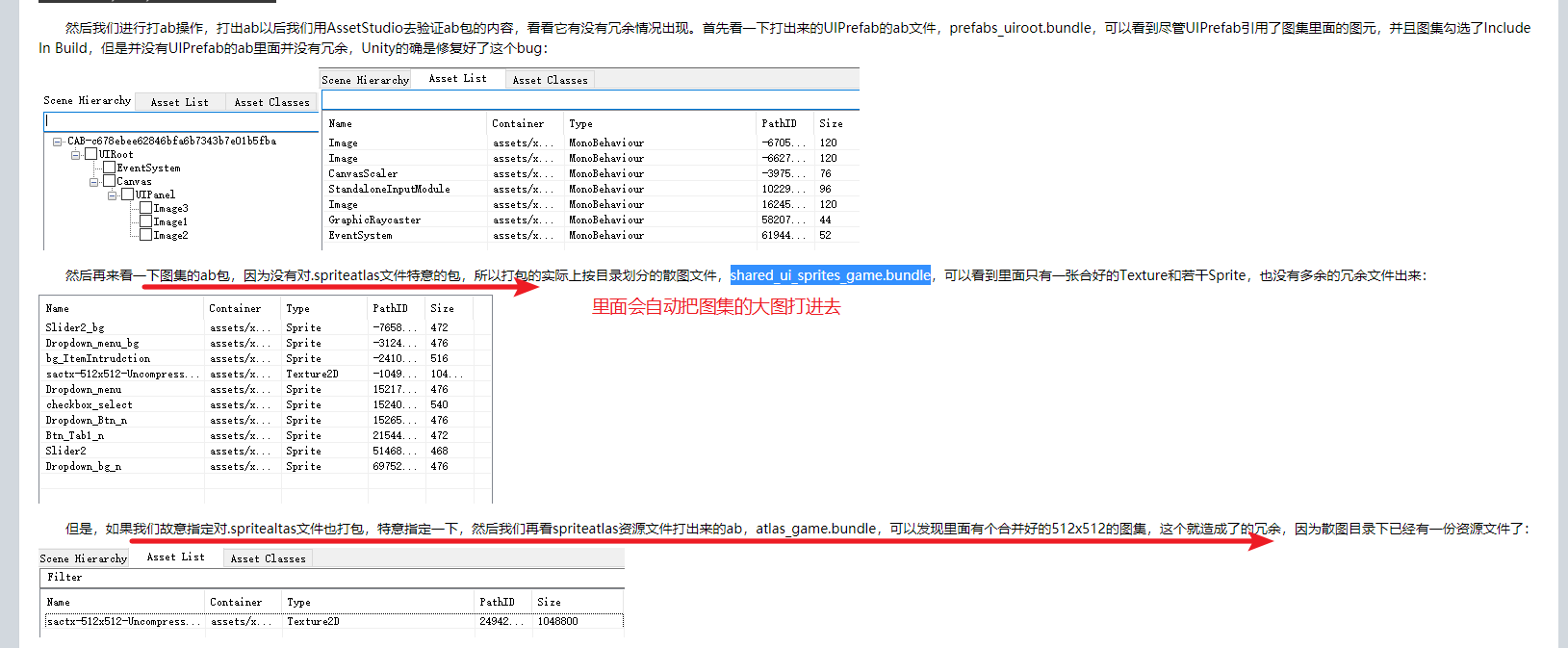
**如果是看的CSDN等网络上的教程的话，多半会让你不要勾选SpriteAtlas的Include In Build选项，说是会造成图集的双份冗余。但是这种说法实际上早就过时了，这个Bug早已经在Unity2018.4.6中修复了，所以我们在使用中放心大胆地勾选Include In Build就好了，这样也可以避免使用LateBinding。**

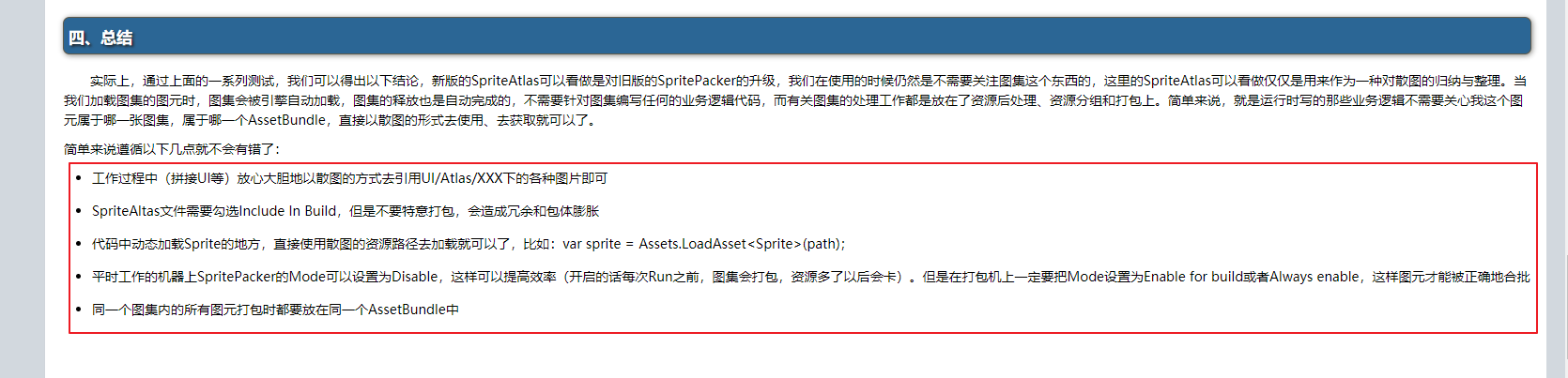


**只要打包的时候勾选图集的Include In Build，然后，不需要主动对SpriteAtlas资产文件进行打包，也不需要写额外的代码，就可以正常使用了，只需要对文件下下的散图进行ab打包即可**

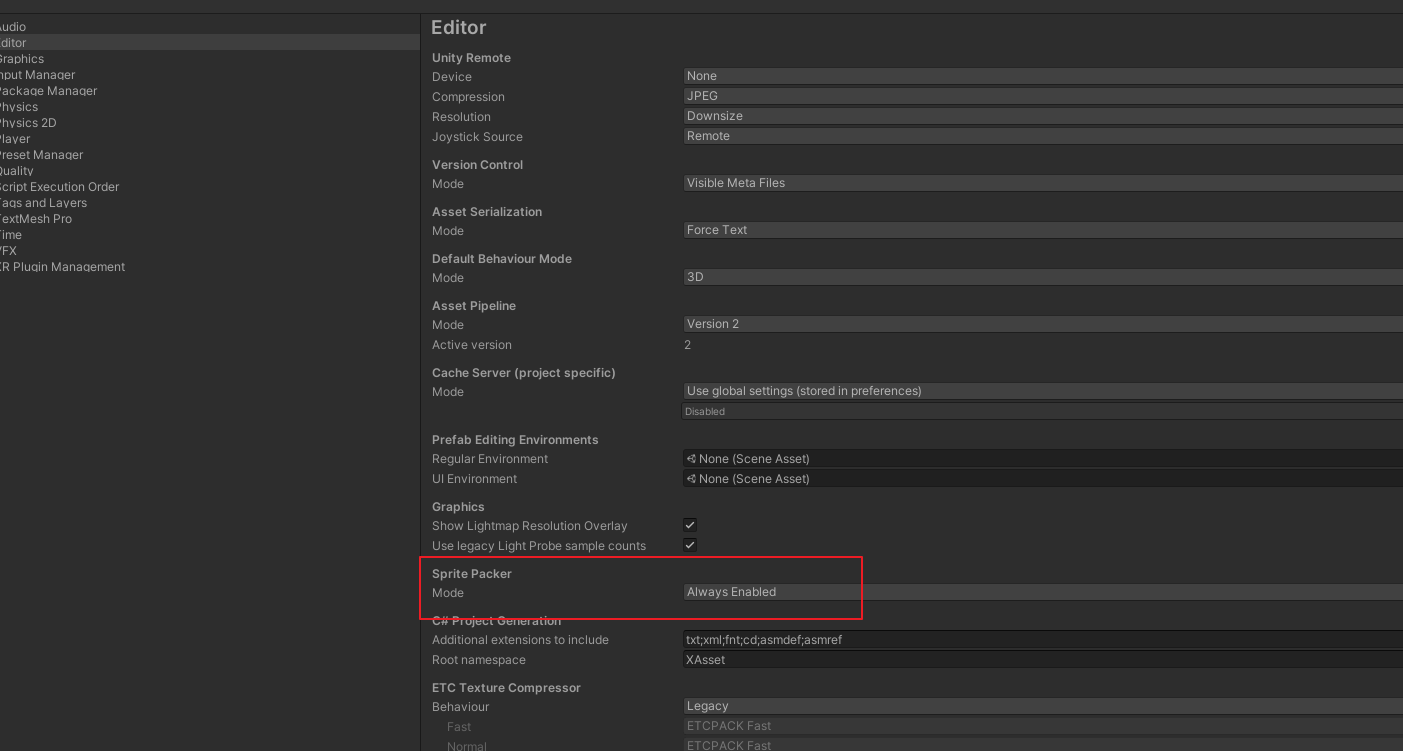








编辑器模式下设置 sprite Packer ==》 always enable就行，真机上没有这个问题



简单来说遵循以下几点就不会有错了：

工作过程中（拼接UI等）放心大胆地以散图的方式去引用UI/Atlas/XXX下的各种图片即可

SpriteAltas文件需要勾选Include In Build，但是不要特意打包，会造成冗余和包体膨胀

代码中动态加载Sprite的地方，直接使用散图的资源路径去加载就可以了，比如：var sprite = Assets.LoadAsset<Sprite>(path)；

平时工作的机器上SpritePacker的Mode可以设置为Disable，这样可以提高效率（开启的话每次Run之前，图集会打包，资源多了以后会卡）。但是在打包机上一定要把Mode设置为Enable for build或者Always enable，这样图元才能被正确地合批

同一个图集内的所有图元打包时都要放在同一个AssetBundle中

Editor下加载ab包中图集，需要将SpritePackerMode切换至AlwaysEnabled状态，否则加载出来的就是白图。（此条感谢海崖提供）