原则一：

1. 每一个场景Scene内尽量少的添加GameObject 和节点
2. 场景内用到的物体尽量通过代码控制加载

优点：

1. 方便多人协作开发，如果多人需要操作Scene，容易引起冲突，且Scene 的冲突解决很麻烦，基本上就是选择舍弃一些人的工作内容
2. 方便打空包，如果场景有依赖其他资源，打包的时候都会被打进安装包
3. 方便维护因为场景内GameObject 少，清晰
4. 避免场景引用一些预设，由于预设的删除，导致场景丢失引用

原则二：

1. 尽量不要在节点或者预设上直接挂脚本
2. 如果GameObject 需要挂载脚本，尽量通过在Runtime时，先加载、获取到GameObject然后通过 AddComponent 方法添加

优点：

1.方便维护，修改、删除脚本，不需要关心哪些预设或者场景内的节点挂载了这个脚本，如果多个GameObject 挂载了这个脚本，修改逻辑后一些GameObject 可能已经不再适用挂载这个脚本了，或者要删除脚本的时候，都需要将挂载这个脚本的预设和节点找出来，一一删除，工作量大，不好找

原则三：

1.尽量减少场景Scene，跟 **原则一** 是相呼应的，如果能做到一个场景内什么都没有，那么只需要在代码逻辑上实现切换逻辑即可，不必要添加两个空的场景，比如创建一个项目启动场景 GameRoot，开始游戏后就直接切换到 主城Scene，主城都是一些UI界面，不需要编辑主城Scene，那么我们只需要在游戏状态机中逻辑上切换到主城，其实还是在GameRoot 场景

原则四：

1. 第三方的插件最好放在一个统一的文件夹下

优点：

1. 方便查看，需要查看用到哪些插件，可以一目了然
2. 方便维护，当需要删除、更新插件的时候，操作方便

原则五：

1.目录结构参考 **资源规范** 文档