命名规范

类名（同 C# Script 脚本名）

首字母全大写（大驼峰）命名法，并控制在 1-3 词
如：PawnManager.cs : public class PawnManager : MonoBehavior {}

变量

|---简易整型/小数/布尔值变量

| 全小写字母+下划线命名法
| **如：** int global_money;
| bool flag=false;

|---对外部唯一对象的引用

| 同名+小驼峰命名法
| **如：** public PawnIManager pawnManager;
| pawnManager.AddPawn(Pawn citizen);

|---定值

| 全大写命名
| **如：** const int MAX_NUM=2147483647;

函数名

首字母全大写（大驼峰）命名法，命名时避免操作主体重复
反例如： PawnManager.cs : public void AddToManager(Pawn citizen);
……pawnManager.AddToManager(new_pawn);
<操作类型><操作主体>
PawnManager.cs : public bool ListIsEmpty();
……if(pawnManager.ListIsEmpty());
<操作主体><状态>

正例如： PawnManager.cs : public void AddPawn(Pawn citizen);
……pawnManager.AddPawn(new_pawn);
<操作类型><操作客体>
PawnManager.cs : public bool IsEmpty();
……if(pawnManager.IsEmpty());
<状态>
P.S. 如果 Manager 中有多个 List<Pawn>，可以改写为
PawnManager.cs : public bool IsEmpty(List<Pawn> list);

在设计函数名时，尽量保证形式相似则功能相似，包括函数名，参数名，参数排序等

数据结构名

|---批量的基础数据

| 同简易变量命名，全小写字母+下划线命名法
| **如：** Vector3Int[] directions = { new(0,1), new(1,0), new(0,0) };

|---批量存储自定义对象的数据结构或是映射这些对象的数据结构

| 小驼峰命名法
| **如：** List<Pawn> pawnList = new List<Pawn>();
| Dictionary<int, int> itemStorage = new Dictionary<int, int>() { { 0, 3 } };
| 其中一对键值对(key,value)映射了 id 为 key 的 Item 对象储存了 value 个

文档管理

BUG/报错信息格式

|---调试信息

| 采用 Debug.Log("something")输出调试信息，可加入自然语言说明信息具体含义
| Unity 会自动追踪 Log 语句所处文件位置，因此可以不在调试信息中加入位置
| **提示** C#支持通过+进行字符串拼接，也支持通过 num.ToString()将整形转化为字符串

|---Warning 与 Error

| 不会导致游戏卡死，但是属于数据异常等意外情况的 bug，用 Debug.LogWarning 输
| 出 Warning 提醒。
| 可能导致卡死, 闪退, 组件无法正常工作, 数据破坏性质的严重错误, 用 Debug.LogError
| 输出 Error 提醒。
| 提醒应说明异常情况，如果在遇到异常时进行了某些应急操作，也应在 Log 中简要说
| 明具体操作。

花括号的安置

左括号与 for/if 语句等位于同一行

缩进类型

进行一层嵌套时，左侧缩进 4 空格位。
为避免代码大幅移动时可能的格式问题，需尽量减少对制表符的依赖。

TODO 标记

对于处于代办事项的代码，需留有 TODO 标识。
待补充注释内容： 在函数体前留
///<summary>
/// TODO
///</summary>
的标识。（这种格式的注释会显示在函数悬停窗口，因此有必要对其进行完善，在下一小节说明。换行/换段标识符为<para></para>）
内容未填充完全的函数：在所有需要填充的代码段位置留 ;//TODO: <具体说明> 的标识
充当备忘录的 TODO 注释：记录在文件末尾，类定义之外。如果涉及具体行数，需在注释中指明。

注释

对函数的说明采用以上格式的注释。函数功能，参数说明，注意事项等均可记录。
函数体内的注释用于进一步说明，或是定位导致 bug 的核心代码
函数体外注释可用于交流，吐槽云云。

空行

非紧密相关/功能相近代码间需要空行，具体实施仅需最终视觉效果不杂乱即可。
如：LoadCache();

pawnManager.Init();
resourceManager.Init();
mapManager.Init();

SaveGameData();

空格

运算符两侧需要留有空格，分号后若未换行也需要留空格， 防止结构过紧造成阅读不便，如：
int x = 0; int y = 1; 而非 int x=0;int y=1;

for(int i = 0; i < n; i++) 而非 for(int i=0;i<n;i++)
if(x == LEAST_NUMBER) 而非 if(x==LEAST_NUMBER)

开发日志/更新日志

代码更新涉及了新功能的删改或是增加均是须记录的，因此推荐由负责人主动记录
如果不想要或是来不及进行日志格式排版，则仅需留下具体内容，当前时间/版本号和负责人信息，以待日后排版。开发日志的阶段性记录以不同版本号为分隔

版本号

v0.0.0.0
采用四段数字记录，
第一段 v1.0.0.0 表示游戏本体已足以发行一个可以自己运行，足以提供正式游玩的 release 版本，因此在达成这个目标前第一段版本号皆为 0
第二段 v0.1.0.0 表示一次卓有成效的开发里程阶段，往往是引入并完善了新的机制或玩法，或是有较多数量的新内容填充
第三段 v0.0.1.0 表示一次充分的补丁改良或严重 bug 修复，对游戏部分已发现并定位的缺陷完成了修复工作，或对游戏中仍有欠缺的地方进行了改进，但是并没有影响整体框架
第四段 v0.0.0.1 表示一次简易补丁或小缺陷的修复，这些缺陷不会影响游戏整体结构，也不会造成游戏卡死等严重问题，可能是一些显示偏移，特定情况下的贴图丢失之类
如果游戏在某个版本进入了测试阶段，在此版本号后加入 alpha，beta 标识，如 v0.7.8.11beta
其中 alpha 测试为开发者之间，或开发者周围小范围的测试
beta 测试为开放了公开下载地址，或公开测试平台的大范围公测

提交

“止于自洽原则”

协作提交最核心的原则，即所有的提交内容都需要保证他人下载并部署到本地后不会出现编译错误，运行错误等严重问题。
每次截至到自己的提交，保证自己的代码的自洽性与代码融入整体后整体软件的自洽性，同时如果开发过程容易中断，也尽量保证自己开发过程中断前代码也处于可以跑通的状态，或是以规约做全良好注释的状态，防止计划打乱

“信息溯源原则”

对于涉及到所有共享文档的提交内容，必须包含的信息有：内容说明，负责人
推荐包含的信息有：时间或当前版本号
推荐的做法有：按照时间顺序排列
需要应用此原则的提交内容有：
github commit message：对于每次提交的 commit message，推荐注明以上信息，如：
2.21 Base --cjh
<时间><内容说明><负责人>

更新日志：向更新日志/开发日志填充新内容时需要注明

todo 清单：清单的发布和接取均需要注名，接取可以没有内容说明
记录负责人时可以采用昵称，如果这样需要额外在群内注明自己昵称