**样例 2048**

使用AD中的命名空间有

AD AD.UI AD.Base AD.Utility

代码总量 <500

脚本数 1

类总数 9

* 架构 1
* 控制器 1
* 数据 1
* 命令 5+1(基类)

效果简介

光标在游戏盘面上拖动即可准备向对应方向合并与移动,同时触发简单的倾斜动画

在松开指针后若满足一定要求则可以立即处理合并与移动,同时触发盘面复原动画与尝试生成一个新的块

类型讲解

# 架构

文本

描述已自动生成

名称 App

ADArchitecture的泛型参数便是类本身,(少数情况下例外,它将告诉ADArchitecture的instance单例的实际类型)

* SendCommand

当使用**ARCHITECTURE\_VIRTUAL**宏时,架构中的部分函数将成为虚函数,

可以复写并实现更加自由的功能,在本例中,其控制**DebugExtension.LogMethodEnable**以指示仅在本架构内命令发送与执行时才启用**DebugExtension.Log & DebugExtension.LogMessage**,这两个函数拥有一个字符串类型的参数,能够将其保存至Debug.dat文件中并记录当前的函数帧栈

*这两个函数在****AD Invokable Call****中被使用,因此使用****AD Invokable Call****的内容可以有效查找死循环的位置与函数异常暂停的位置 你一样可以通过****DebugExtension,LogMethodEnable=false****指示其关闭以提高效率*

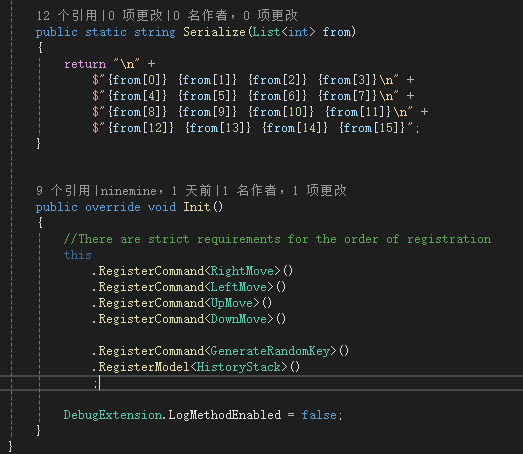
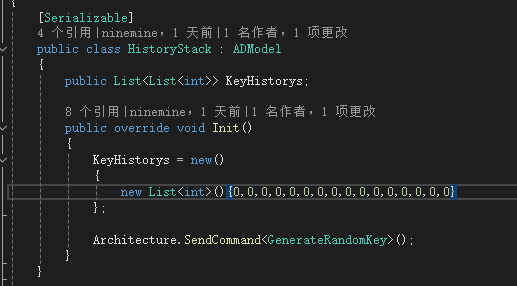
文本

描述已自动生成

* Serialize 将某一盘面数据转换为字符串
* Init 对命令与数据类进行注册,并对**DebugExtension**进行指示

在架构中的Init所进行注册的顺序是有要求的,因为某些实例的Init对其他实例有依赖关系

# 数据

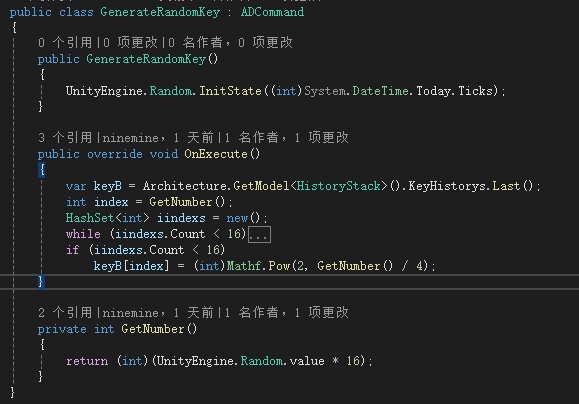
HistoryStack

某一时刻的盘面是一个List<int>类,而按时间顺序收集后作为KeyHistorys,使得游戏中可以被保存历史操作或回退至某一操作

初始化**Init**中赋予了初始盘面,同时发送命令**GenerateRandomKey**来生成第一个块

# 命令

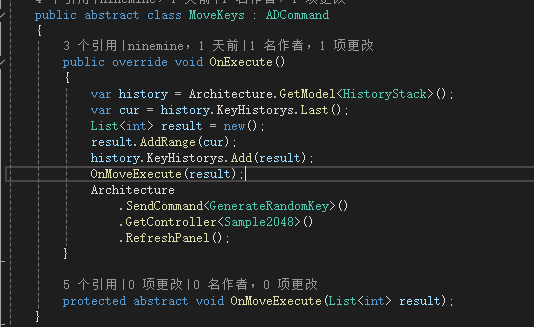
GenerateRandomKey



构造函数中通过时间初始化随机数的种子

执行命令的内容是尝试在一个空闲的格子内生成一个块,大小为1~2^4

MoveKeys



抽象命令,执行命令的过程中,首先获取当前盘面的数据,并在最终发送一次**GenerateRandomKey**后调用控制器刷新

其具体行为由四个子类命令实现**OnMoveExecute**来完成

# 控制器

Sample2048

所有可以看见的内容均由其负责控制与交互

控制器为了保证有序执行又不依赖于Unity提供的Order特性,一般不推荐使用Awake来进行初始化

* 首先,初始化由Start消息函数进行,在对App进行初始化后将自身注册
* **Init**,调用一次**InitVecCounter**,这将清理计时器与控制向量,随后在**TouchPanel**上添加监听函数**CatchPointerDrag**,并通过ADG注册鼠标左键的点击和松开事件,最终进行一次刷新

通过ADG注册输入事件时尽量接收返回值RegisterInfo,并注意其生命周期或手动释放,以免其自行构析后导致监听被解除

* 在**TouchPanel**与ADG上注册的监听将会处理输入,并通过计时器和控制变量**Delta**决定发送哪一方向上的命令
* **OnDestroy**,**ADController**的base版本会将自己从架构中解除注册,此处又将注册的输入监听也手动解除注册

