# 战斗场景系统

## 1.综述

战斗场景即玩家在进入关卡之后正式进行游玩的部分，玩家可以通过合理的搭配卡片组成坚固的防线来防守敌人的进攻，直到最终胜利或者失败，需要使用Unity中单独的一个Scene实现。

## 2.用户需求

### 2.1

A：关卡设计上希望能设计几套固定的出怪模版，然后可以自定义几个不同的模式，并且支持上述出怪模版、游戏模式可以任意两两组合进游戏里。拿原版举例子就是，这里先写了一个曲奇岛的出怪模版和魔塔130的出怪模版，游戏模式写了普通通关战模式和争分夺秒模式，最终要系统不仅能做到普通通关战模式的曲奇岛出怪模版、和争分夺秒模式的魔塔130出怪模版，也能做到争分夺秒模式的曲奇岛出怪模版和普通通关战模式的魔塔130出怪模版。

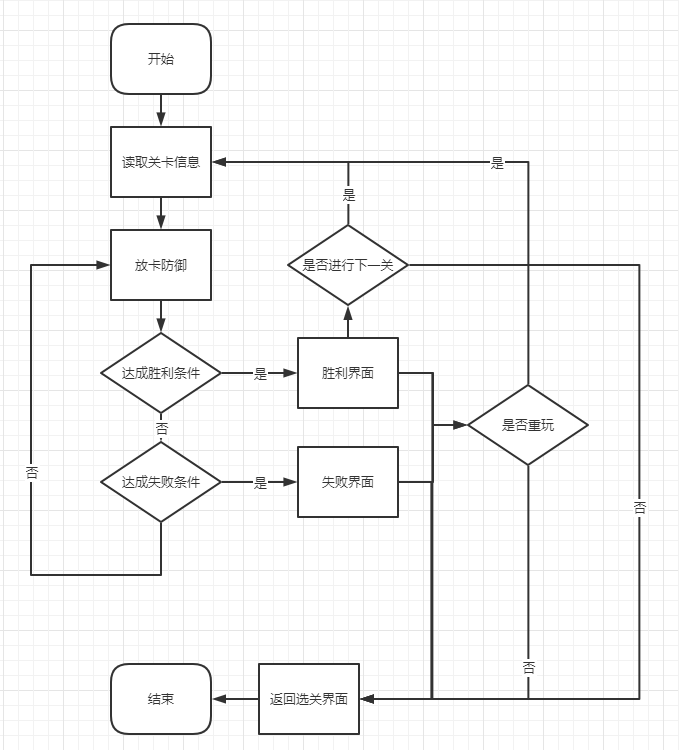
B：希望自定义的模式能够有极大设计自由度，能非常舒服得制订规则，比如说失败条件不一定像传统的老鼠进家或者人物死亡，可以改为别的，相应的判定也用代码实现。胜利条件也可以改成其它，换言之在符合开闭原则的基础上只通过增加新的类就能实现PVZ里所有支线益智小游戏的成功与失败条件。

C：想做一个类似于明日方舟危机合约的系统，并且可以附加在任意关卡上。（抽象的说就是能实现在原游戏基础上直接改变一些游戏规则，比如数值增加，并且提供GUI供玩家调整）

## 3.系统总体结构

本系统作为游戏的主要系统，由各种关卡选择界面进入。

系统流程图如下



## 4.类图的设计以及说明

### 4.1 战斗场景控制器

|  |  |
| --- | --- |
| 类名 | 战斗场景控制器(GameController) |
| 整体说明 | 控制战斗场景的游戏逻辑。 |
| 成员说明 | + 当前关卡（currentStage:BaseStage）  + 规则表(ruleList:Rule[])  + 存活的敌方单位表(enemyList:BaseUnit[])  + 格子表(gridList:Grid[]) |
| 方法说明 |  |
| 关系说明 | 继承自MonoBehavor类 |
| 实现细节 | 一个控制类，用来顺序执行其成员的方法。  举例运用：规则表里定义了两个Rule，其中一个Rule是设定初始火苗资源为100，并且每秒额外自动回复1点火苗。第二个Rule是设定所有卡片CD减半，但消耗增加50%。这些Rule的实现均使用Rule类中的Init()与Update()方法封装起来并且被该控制器调用以实现。因此也实现了每个规则能独立执行并且与原关卡模式效果相互叠加作用，能实现自由拓展游戏规则、满足设计危机合约的需求。 |

### 4.2关卡类

#### 4.2.1 关卡基类接口

|  |  |
| --- | --- |
| 接口名 | 关卡基类接口(IBaseStage) |
| 整体说明 | 声明每个关卡必须实现的接口。 |
| 方法说明 | - 成员变量初始化方法(Init():void)  - 更新方法 (Update():void)  - 胜利条件(WinCondition():bool)  - 失败条件(LossCondition():bool) |

#### 4.2.2 关卡信息结构体

|  |  |
| --- | --- |
| 结构体名 | 关卡信息(StageInfo) |
| 整体说明 | 存储静态关卡信息数据，实现序列化，信息可转化至本地。 |
| 成员说明 | + 关卡名(name:string)  + 轮数表(roundList:Round[])  + 一大波标志下标表(waveIndexList:int[]) |

#### 4.2.3 关卡基类

|  |  |
| --- | --- |
| 类名 | 关卡基类(BaseStage) |
| 整体说明 | 最基础的关卡类，可以作为默认关卡的模型。 |
| 成员说明 | + 关卡信息(stageInfo: StageInfo)  + 奖励(prize:Prize)  - 战斗场景控制器(mGameController: GameController) |
| 方法说明 | - 成员变量初始化方法(Init():void)  - 更新方法(Update():void)  - 胜利条件(WinCondition():bool)  - 失败条件(LossCondition():bool) |
| 关系说明 | 被GameController类聚合，实现关卡基类接口(IBaseStage) |
| 实现细节 | 在构造方法被调用后，在GameController的OnStart()中调用Init()方法来处理。  使用责任链模式来处理每轮出怪逻辑（根据轮数表），并且每当轮数值在一大波标志下标表有作记录时，显示一大波标志特效，当到达最后一轮时显示最后一波标志特效。  更新方法Update()可以用来实现不同关卡类的特色规则，基类里默认为空。  更新方法Update()、胜利条件WinCondition()和失败条件LossCondition()在GameController的Update()方法依次调用，一旦达成胜利条件便暂停当前战斗场景逻辑，显示出相应的页面。在达成胜利条件一刻马上调用奖励(prize)成员的Reward()方法来给予玩家对应的奖励。 |

#### 4.2.4 普通模式关卡类

|  |  |
| --- | --- |
| 类名 | 普通模式关卡类(NormalStage) |
| 整体说明 | 原版游戏的默认通关战模式。 |
| 成员说明 | - 所有敌方单位是否被清除(isClear:bool) |
| 方法说明 |  |
| 关系说明 | 继承自关卡基类(BaseStage) |
| 实现细节 | 胜利条件为所有敌方单位被清除，失败条件为敌方单位进家。  所有老鼠被清除：在Update()中先看看轮数表roundList责任链是否为空，访问控制器GameController中的是否还有存活敌方单位，若都没有，则置isClear为true，之后胜利条件部分直接判断这个布尔值即可。  敌方单位进家：遍历控制器GameController中存活的敌方单位表enemyList，任意一只的横坐标在特定范围之外即可判定进家。 |

#### 4.2.5 争分夺秒模式关卡类

|  |  |
| --- | --- |
| 类名 | 争分夺秒模式关卡类(TimeLimitStage) |
| 整体说明 | 在普通模式关卡的基础上增加一定的时间限制的模式。（为了便于维护，不继承普通模式关卡类） |
| 成员说明 | - 总时限(TotalTime:float)  - 剩余时限(TimeLeft:float)  - 是否超时(isTimeout:bool)  - 所有敌方单位是否被清除(isClear:bool) |
| 方法说明 |  |
| 关系说明 | 继承自关卡基类(BaseStage) |
| 实现细节 | 胜利条件同普通模式关卡类，失败条件在普通模式关卡类的基础上外加一条是否超时(isTimeout)  其中，剩余时限(TimeLeft:float)会在每次Update()中减去1/（正常游戏帧数），当剩余时限小于0时置isTimeout为true。 |