July 5, 2017

Yinghao (Michael) Wang

《解救饭堂》游戏开发团队

《解救饭堂》游戏软件设计文档

V1.0

Table of Contents

[1 项目文档简介 4](#_Toc487192952)

[1.1 目的 4](#_Toc487192953)

[1.2 软件概况 4](#_Toc487192954)

[1.3 本文档组织 4](#_Toc487192955)

[2 设计考量 4](#_Toc487192956)

[2.1 假设 4](#_Toc487192957)

[2.2 约束条件 4](#_Toc487192958)

[2.3 系统环境 4](#_Toc487192959)

[2.4 设计方法论 4](#_Toc487192960)

[2.5 可能的设计风险 4](#_Toc487192961)

[3 架构设计 4](#_Toc487192962)

[3.1 架构概况 5](#_Toc487192963)

[3.2 塔楼建造逻辑设计 5](#_Toc487192964)

[3.2.1 用户点击建造位置 6](#_Toc487192965)

[3.2.2 用户点击建造一种塔 6](#_Toc487192966)

[3.3 塔楼与炮弹关系设计 7](#_Toc487192967)

[3.4 敌人对象的生成，管理，伤害与击退 7](#_Toc487192968)

[3.4.1 关系类图 7](#_Toc487192969)

[3.5 游戏伤害，收益，结束条件 9](#_Toc487192970)

[3.6 Buf（负面效果）设计 10](#_Toc487192971)

[3.6.1 关系图 10](#_Toc487192972)

[3.6.2 顺序图（以BleedBuf举例） 10](#_Toc487192973)

[3.7 血条实现与管理 10](#_Toc487192974)

[3.8 游戏结束 11](#_Toc487192975)

[4 用户交互设计 12](#_Toc487192976)

[4.1 玩家引导 12](#_Toc487192977)

[4.2 交互元素风格 12](#_Toc487192978)

[5 安全性考虑 12](#_Toc487192979)

修改记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **姓名** | **修改原因** | **日期** |
| *1.0* | 王颖豪 | *Initial Revision* | *07/05/2017* |
|  |  |  |  |

确认及许可

|  |  |  |
| --- | --- | --- |
| 姓名 | **签名** | 日期 |
| *王颖豪* | *Wang Yinghao* | *23/06/2017* |
|  |  |  |

# 项目文档简介

《解救饭堂》是一款定位在PC端的塔防类游戏。玩家通过在游戏中固定的位置建造餐厅，向下课的同学们投掷食物来增加学生的饱腹感。吃饱了的学生就不会一拥而上围堵食堂了。

## 目的

本文档的根本目的在于细化说明软件需求说明书中规定的设计内容。本文档将从设计的角度详细阐述本游戏的设计思路和部分实现细节。

## 软件概况

本软件开发环境为Unity 5.6。开发语言为C#。未使用其他插件。根据需求文档，本软件的部署环境为现代Windows与Macintosh系统。

## 本文档组织

本文档将分三部分来介绍本游戏的设计概况：项目简介，设计考量，设计架构。

# 设计考量

这部分将根据游戏的开发环境，开发限制，需求文档中的限制来说明软件设计需要考虑的因素。

## 假设

玩家具备操作计算机的基本技巧，并能理解常见的游戏提示。

## 约束条件

无

## 系统环境

本游戏支持在现代Windows与Macintosh操作系统平台上运行。要求玩家配置有基本的现代显示驱动。

## 设计方法论

MVC框架与基于组件的编程方法

## 可能的设计风险

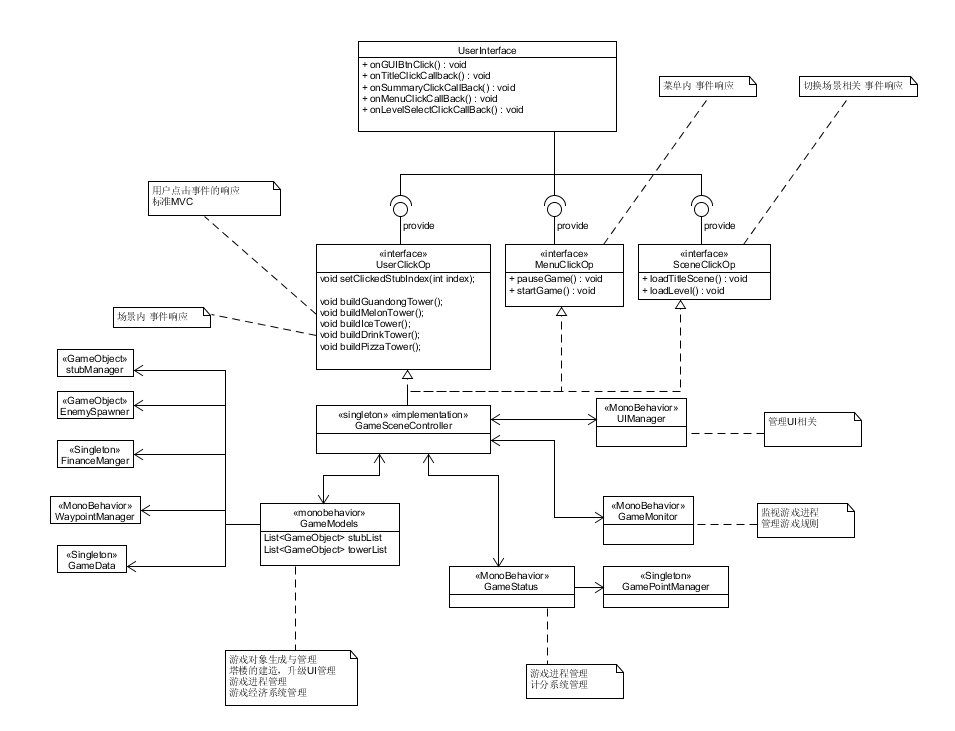
无

# 架构设计

本游戏的设计基于MVC框架来处理用户的输入。利用基于组件的编程模式来实现需要引擎驱动的功能。

## 架构概况

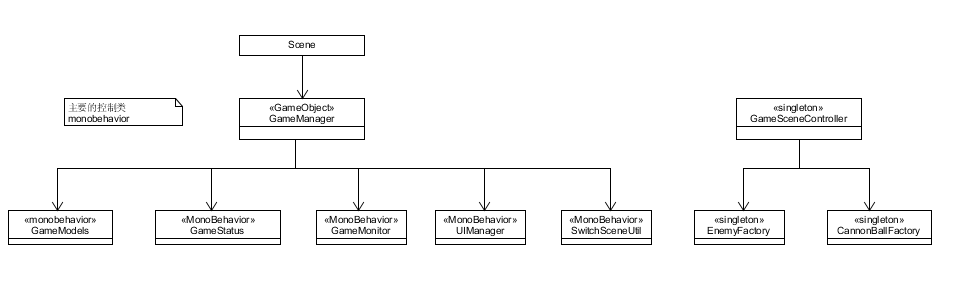
游戏主要控制类关系类图如下：



该设计的核心为GameSceneController控制器。该控制器作为MVC框架中的控制器单元，将用户的输入响应传递给下级的模型单元。

该控制器也很大程度上担任游戏的逻辑中枢，主要的Monobehavior之间的交互通过GameSceneController提供的接口来实现。目的在于减少GetComponent的使用，提高性能。

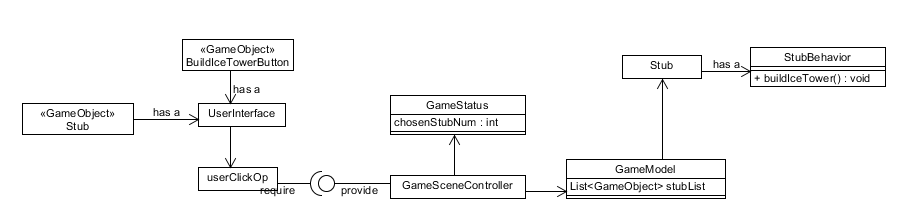
该控制器一定程度担任游戏单例对象的生命周期管理。见下图。



游戏主要控制类别的MonoBehavior的驱动，通过挂载在永久的游戏对象GameManager上实现。

## 塔楼建造逻辑设计

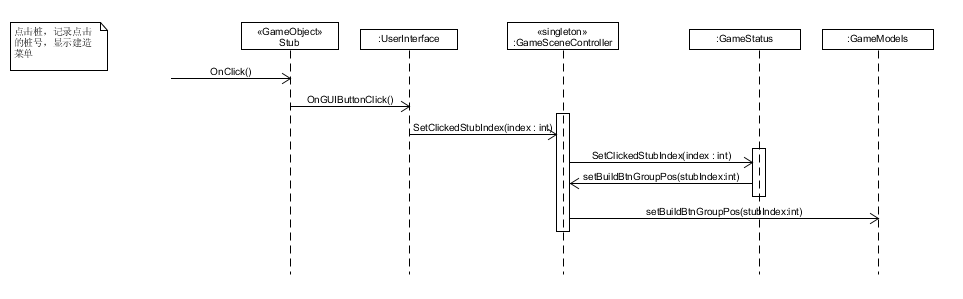
与塔楼建造相关的类图：



本图非标准UML类图。主要区别如下：为表示一个GameObject下挂载了这个组件，我们使用实线箭头加”has a”标注来表示，下同。

### 用户点击建造位置

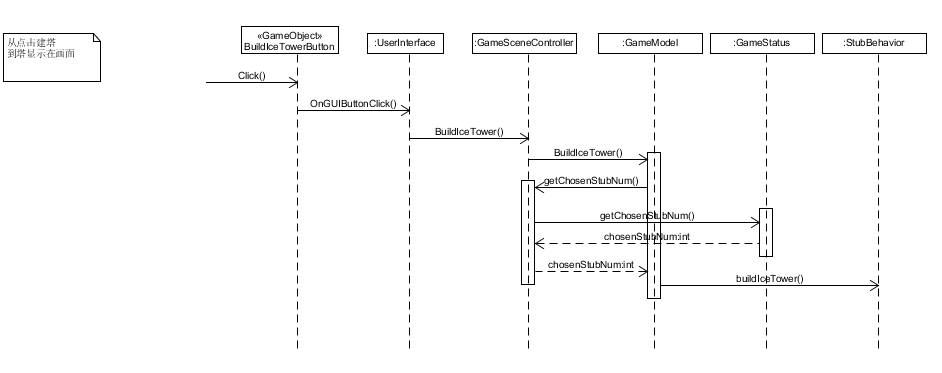
用户点击建造位置后，需要记录下当前点击过的位置并显示建造菜单。这部分的顺序图如下：



后置条件：点击过的建造位置被记录；建造菜单显示在建造的位置上。

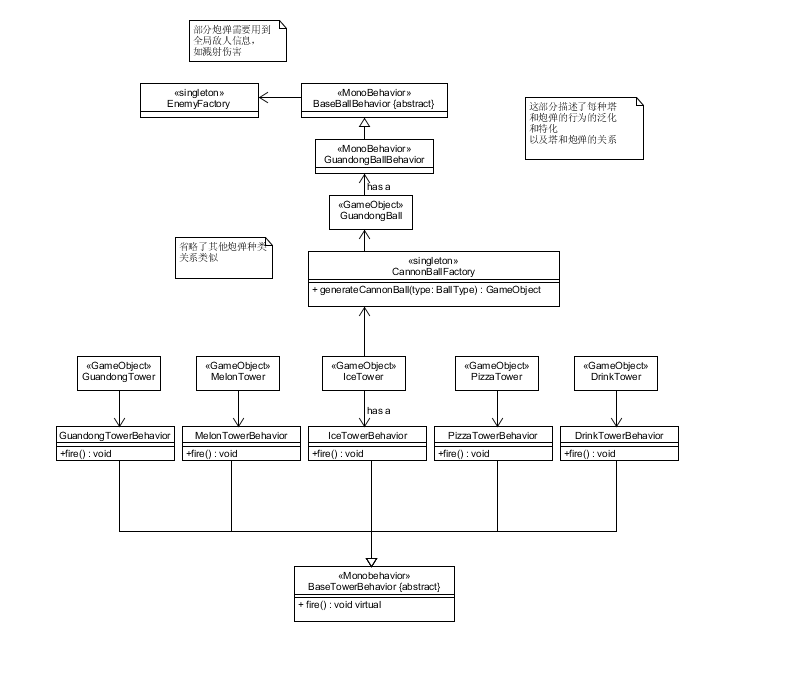
### 用户点击建造一种塔

用户点击建造某一种塔后，需要将该位置的桩的游戏对象替换为用户需要建造的塔的对象。这部分的顺序图如下：



后置条件：建造桩的对象被用户需要建造的塔所替代。

## 塔楼与炮弹关系设计

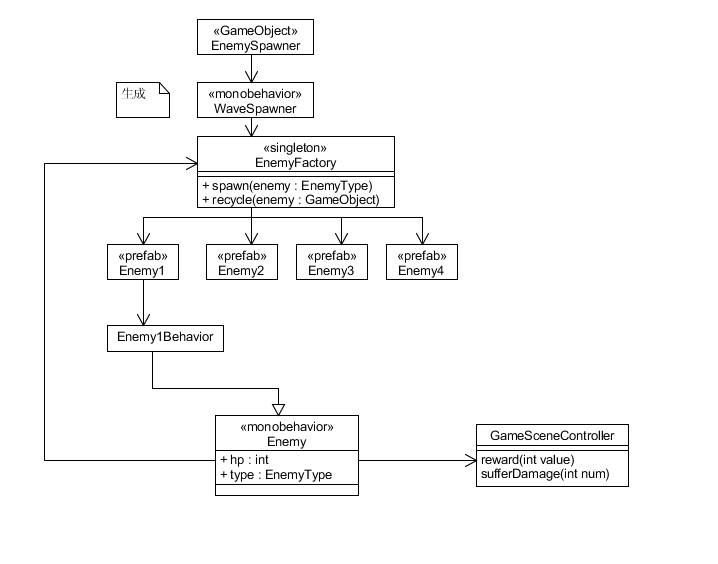


塔楼都有一个特化的Behavior组件，这个组件里重载了fire方法，因而每种塔楼都有自己独特的发射炮弹的方法。所有的TowerBehavior均继承BaseTowerBehavior基类。

与塔楼设计类似，炮弹也有特化的Behavior组件，也均继承BaseBallBehavior基类。

## 敌人对象的生成，管理，伤害与击退

### 关系类图



敌人对象的生成通过场景中永久挂载的游戏对象EnemySpawner上的Monobehavior – WaveSpawner以及单例EnemyFactory来实现。

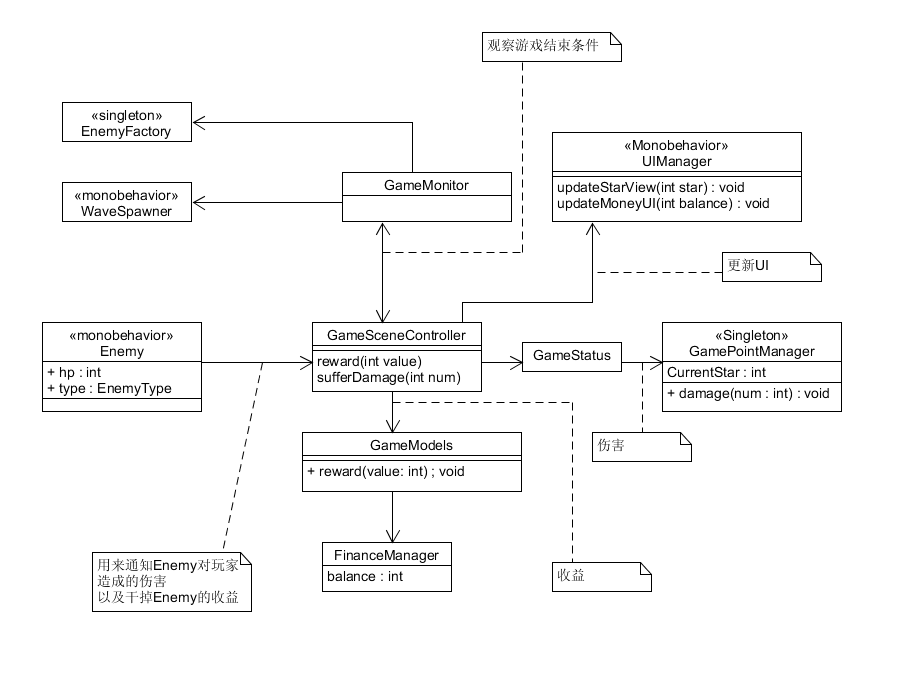
WaveSpawner管理生成的节奏。

WaveSpawner调用EnemyFactory，生成并重复利用Enemy对象。工厂模式提高性能。

每一个Enemy挂载特化的Behavior，因而可以定制每种Enemy的行为特点。他们继承基类Enemy。

Enemy能向GameSceneController发送消息以通知该Enemy对玩家造成的伤害，或Enemy被干掉时玩家获得的收益。

## 游戏伤害，收益，结束条件



GameSceneController通知GameModels处理收益相关事项。通知GameStatus处理玩家生命值（剩余星数）事项。

每次玩家遭受损伤或者获得收益，GameSceneController通过UIManager更新HUD显示。

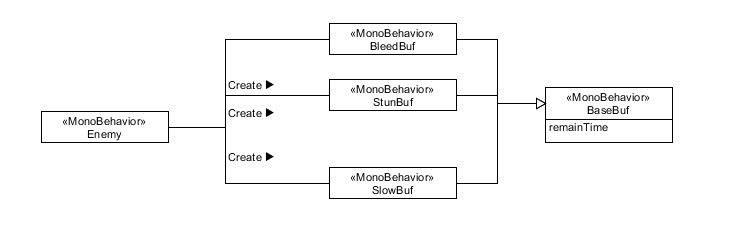
游戏对象GameManager上挂载永久脚本GameMonitor，记录该关剩余时间，通过监察WaveSpawner和EnemyFactory，当满足游戏结束（胜利或失败）通知GameSceneController。

游戏胜利与失败条件：

|  |  |
| --- | --- |
| 当前关卡时间结束，玩家生命值不为0 | 胜利 |
| 所有波数生成结束，所有敌人被消灭，玩家生命值不为0 | 胜利 |
| 玩家生命值为0 | 失败 |
| 临界条件：玩家生命值为0，关卡时间刚好结束 | 胜利 |

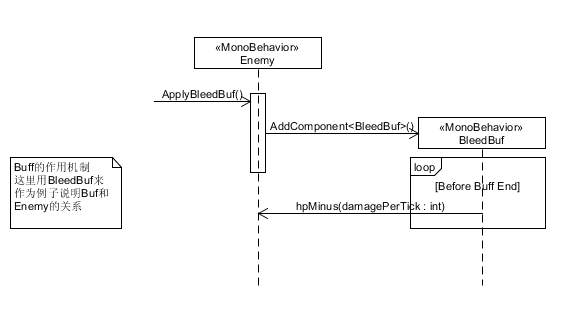
## Buf（负面效果）设计

### 关系图



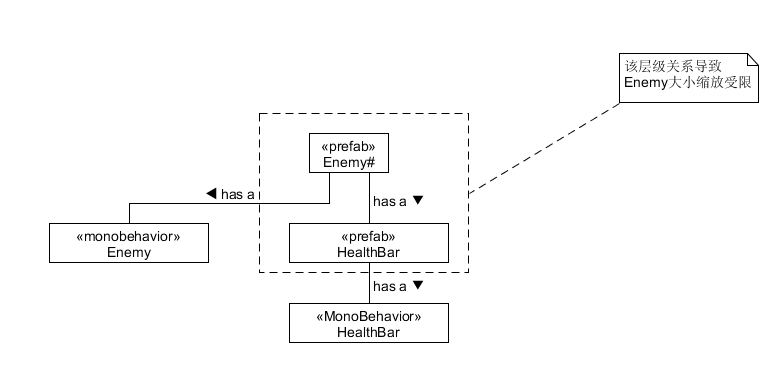
当敌人受到负面效果时，动态创建Buf Monobehavior。

### 顺序图（以BleedBuf举例）

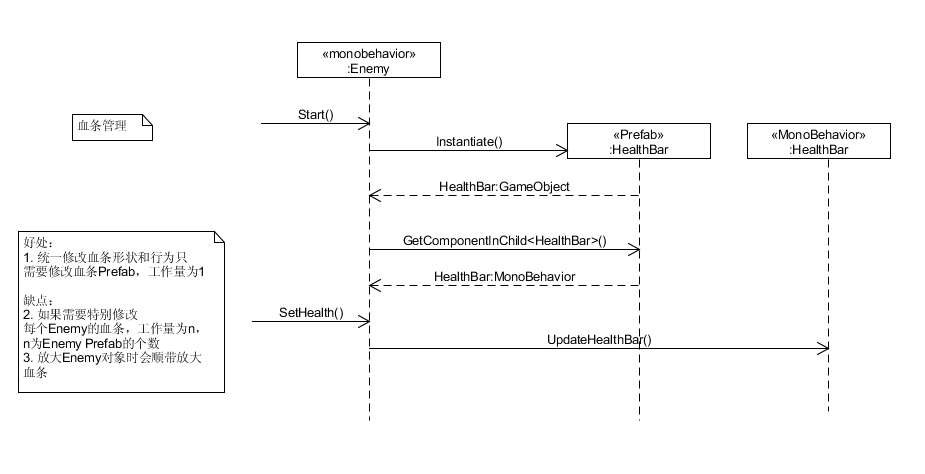


（炮弹）作用在Enemy上，炮弹本身具有BleedBuf的效果，就会调用ApplyBleedBuf()方法，该方法动态创建BleedBuf MonoBehavior。该Buf的效果是固定时间削减Enemy的生命值。

## 血条实现与管理

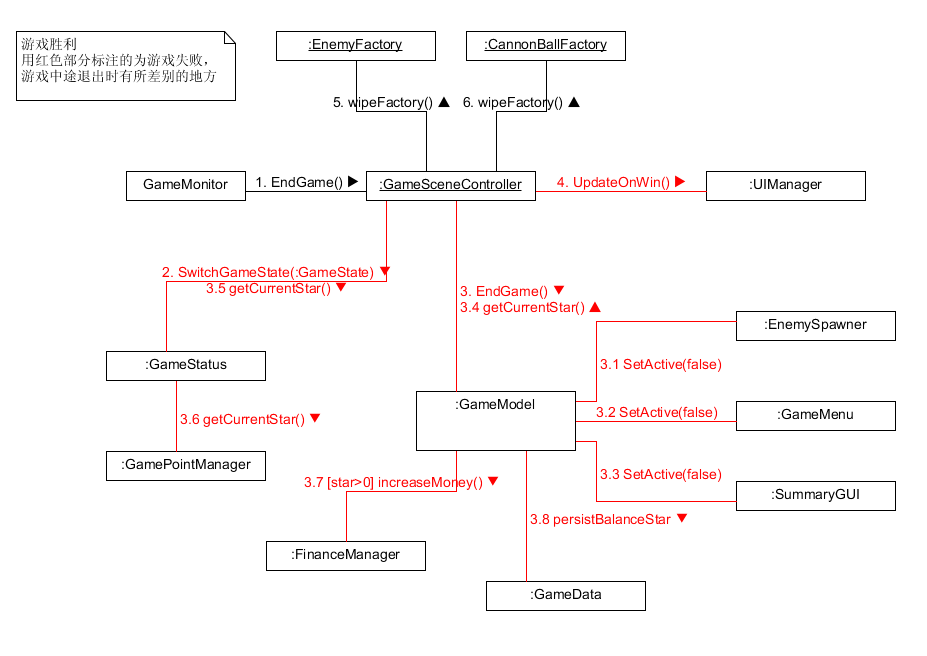


我们将血条设置为Enemy的一个子对象。这样血条对象能跟着Enemy移动。血条对象是动态创建的。由于不同的Enemy对象共用同一个血条对象，因此用代码控制只需一行。



可以看到，Enemy对象中长期持有MonoBehavior HealthBar的一个实例，因此能随时更新。

## 游戏结束



由于涉及对象过多，这里采用Communication Diagram来代替Sequence Diagram。

当GameMonitor发出游戏结束请求时，GameSceneController需要做三件事情：

1. 通知UIManager更新UI
2. 通知GameModel进入游戏结束逻辑
3. 清空工厂类中的游戏对象（否则会因访问销毁的游戏对象而出现异常）。

# 用户交互设计

需求文档中并未明确指出用户交互设计的定义。因此，本文档将根据通用的用户交互设计准则，以及项目实际开发约束来给出交互设计的定义。

## 玩家引导

本游戏应具备基本的故事背景介绍，应尽可能早将故事背景显示给用户阅读。

本游戏从启动到进行游戏，到退出游戏，都应当有清晰的用户交互元素供用户满足其目的。“清晰”的定义为：具备基本计算机操作技巧的用户，在不需额外说明的情况下，能自行使用。

## 交互元素风格

本游戏所有交互元素应具备统一的交互元素风格。

# 安全性考虑

本游戏无特殊安全性考虑。

本游戏无外部数据访问，因此也无特殊数据安全性考虑。