|  |  |  |  |
| --- | --- | --- | --- |
| 修订版本 | 修订人 | 修订日期 | 修订内容 |
| v1.0 | 王智键 | 2019.6.3 | 代码规范、代码框架初步编写 |
|  |  |  |  |

# 代码框架

GameManager - 游戏管理者，全局只有一个，代表整个游戏

ScenesManager - 场景，管理一个场景（或者嵌套多个子场景），包括一些场景管理。

BaseObject- 对象，应该是所有场景对象中的基础类，包含在Scene中

Player：保存玩家的状态、提供改变玩家运动和状态的函数

MovableCube：保存可移动地图块的属性信息，提供可改变运动的函数。

Door：保存门的状态，提供改变门的状态的函数，作为门的基类

ChargeDoor：充能门类，提供可改变玩家buff状态的信号的函数（充能buff）

ElasticDoor：反弹门类，提供改变玩家方向状态的信号的函数

PortalDoor：传送门类，提供改变玩家位置的信号的函数

GravityDoor：重力门类，提供改变玩家buff的信号的函数（重力buff）

Key：提供销毁自己的信号的函数

Reward：提供销毁自己的信号的函数

Trap：提供玩家已死亡的信号的函数

ThornTrap：地刺陷阱

BallTrap：链球陷阱

Enermy：保存敌人基本属性、状态，提供玩家已死亡的信号的函数。

UI - 界面相关

AI – 敌人的AI

AudioManager - 音效管理

Buff - buff类

Effect - 特效类

ResourceManager – 资源管理类

游戏管理者GameManager代表一个游戏，整个生命周期之内只存在一个实例。对于简单的游戏来说，这里面会包括一些全局范围的功能，例如加载和切换场景等等。

场景类Scene管理着当前场景的所有对象，包括玩家、敌人、NPC、物品、特效等等。例如控制玩家的运动和改变玩家的状态（buff、死亡等），地图内物品的运动或状态的改变等。

通常UI和Scene是相关的，所有可以在Scene的加载阶段，或者运行期生成相关的UI，然后在场景销毁的时候统一销毁UI。

实体对象基类Actor是所有能在场景中显示的对象的基类，例如玩家、NPC、特效等等。具有一些通用的逻辑，包括位置、方向，如果是士兵派生类的话应该需要有模型、武器、战斗属性等内容。

现在很多先进的引擎会把Object的功能拆分为组件组合的模式。例如空间实体组件具有位置、方向、走动、攻击等功能，而特效会具有播放、暂停、调整速率等功能。这些不同的组件组合成一个复杂的Object，组合方式简单的用list保存所有的component。

# 项目规范

1. **文件夹层级**

Scenes：保存游戏内的所有场景

Scripts：保存各脚本

GameManager

ScenesManager

AI

UI

AudioManager

Buff

Effect

ResourceManager

Object

Prefabs：保存各种预设体

Model：保存各种模型资源

Resources：保存各种资源

Music：各种音乐、音效

BGM：背景音乐

Sounds：音效

Image：保存各种图片

Maps：地图

Roles：角色

Props：道具

UI：UI

Material：保存各种材质

Shader：保存着色器

1. **命名规范**

小驼峰法：

变量一般用小驼峰法标识。驼峰法的意思是：除第一个单词之外，其他单词首字母大写。

大驼峰法：

相比小驼峰法，大驼峰法（即帕斯卡命名法）把第一个单词的首字母也大写了。常用于[类名](https://baike.baidu.com/item/%E7%B1%BB%E5%90%8D" \t "https://baike.baidu.com/item/%E9%AA%86%E9%A9%BC%E5%91%BD%E5%90%8D%E6%B3%95/_blank)，命名空间等。

常量const和宏定义使用全大写单词。

管理器：xxxManager 单实例统一getInstance();

玩家相关： plXxx

控制： ctrlXxx

地图： xxxMap

界面： UIXxx

道具： propXxx

特效： xxxEff

buff： xxxBuff/xxxDebuff

定时器： xxxTimer

物体： xxxObj

门： ­­xxxDoor