<Endless Memories>

软件架构文档

版本 <2.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <07日/11月/2020年> | <1.0> | <初步拟定软件架构文档，包括简介，用例视图，逻辑视图，部署视图，实现试图> | <Naomi,Izumi,Edward,叶鹏鹏> |
| <28日/12月/2020年> | <1.1> | <最终决定软件架构文档，包括简介，用例视图，逻辑视图，部署视图，实现试图> | <Naomi,Izumi,Edward> |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

2. 用例视图 4

2.1 玩家用例模型 5

3. 逻辑视图 6

3.1 应用层 6

3.2 业务层 8

3.3 中间层 9

4. 部署视图 9

5. 实现视图 9

6. 设计战术 10

软件架构文档 （简化版）

# 简介

## 目的

本文档描述了项目Endless Memories在开发过程中构架的结构以及构架的设计思路。

由于项目整体为单线程，因此我们略去了进程视图，我们把重心放在分量更多（对于我们的项目来说）的用例视图，逻辑视图与实现视图上。

该文档的主要目的是为了让开发人员能够更好地理解项目整体的架构，方便他们进行开发，以及用例视图能够让玩家更好的理解项目的使用（游玩）方法。

# 用例视图

该节将介绍软件架构中的用例视图。该节将列出体现EndlessMemory中重要、核心功能的用例；或是涵盖了软件架构方面的许多元素；或是强调、阐明了软件架构中某些非常核心的部分。

部分 EndlessMemory 用例模型有：

- 逃生者操控

- 物品收集

- 物品投掷

- 物品查看

- “死神”角色玩家设置“鬼”的初始位置

- “死神”角色玩家投掷物品

这些用例全部以玩家作为执行者，并且与“死神”角色相关的所有用例都在Multiplayer（双人剧情模式）下才能使用。

## 玩家用例模型

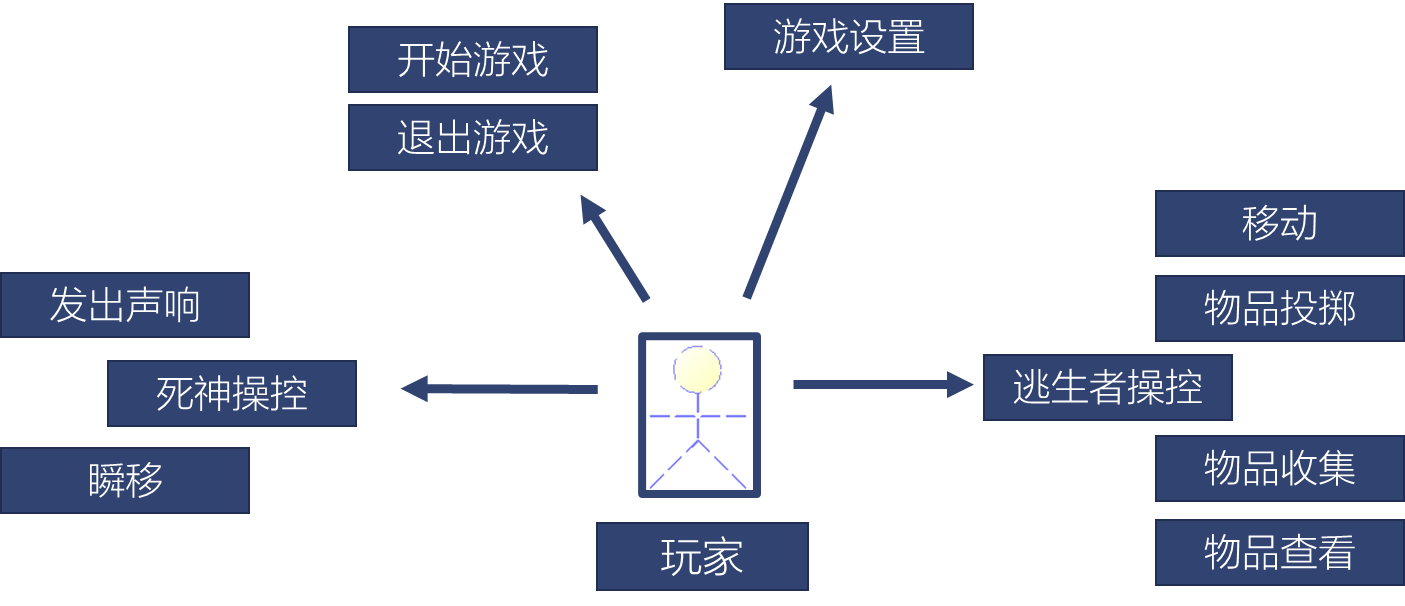


图1：模型名称：玩家用例模型

### 逃生者操控

第一人称视角玩家（单人剧情模式，PVP对战模式“逃生者”角色）在游戏场景中通过触摸手柄触摸板或头盔转动操控游戏主角。

### 物品收集

第一人称视角玩家（单人剧情模式，PVP对战模式“逃生者”角色）在游戏场景中收集近距离面对的物品。

### 物品投掷

第一人称视角玩家（单人剧情模式，PVP对战模式“逃生者”角色）在游戏场景中打开背包并查看物品。

### 物品查看

第一人称视角玩家（单人剧情模式，PVP对战模式“逃生者”角色）在游戏场景中打开背包并查看物品。

### “死神”角色玩家设置“鬼”的初始位置

PVP对战模式中扮演“死神”角色玩家进行每轮攻击前将“鬼”放置在地图的任意一处。

### “死神”角色玩家投掷物品

PVP对战模式中扮演“死神”角色玩家在游玩过程中将石头丢至地图某处以制造噪音，吸引“鬼”的注意力。

# 逻辑视图

EndlessMemory的逻辑视图由三个主要包构成：应用层、业务层、中间件

应用层包含边界类，接受用户的UI操作，处理游戏内的角色操控。

业务层包含控制类，管理所有数据，处理所有非玩家操控物体的举动。

中间层包含Unity，业务层把所有处理分解到一个个输入事件监听与场景变化后，Unity来实现实际变化，反应至屏幕上。

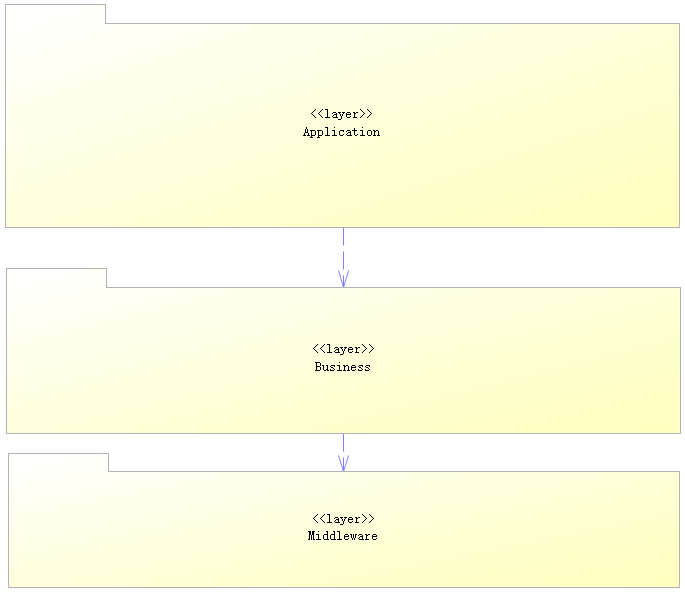


图 2：逻辑视图主要构成

## 应用层

应用层主要有三个包，游戏开始、设置、关卡互动。

### 游戏开始（Game Start）

选择游戏模式（单人/双人）后，开始游戏，进入关卡。纯UI操作流程。

### 设置（Setting）

调整音量、亮度、画质等。纯UI操作流程。

### 关卡互动（Stage）

根据选择了何种游戏模式会改变行为：

单人模式 / 双人模式的逃跑方：

玩家操控关卡内的角色，与场景互动，逃离鬼，使用各种道具想办法逃出建筑物。

双人模式的死神放：

协助关卡内的鬼，监视场景内的声音/鬼的视野范围，让逃跑者死于鬼的攻击下。

注：目前还未抽象化所有VR按键输入，所以实际上还未存在任何实际的边界类在应用层，暂时都是通过Unity的事件监听功能在业务层直接完成接受。以后预计进行抽象化。

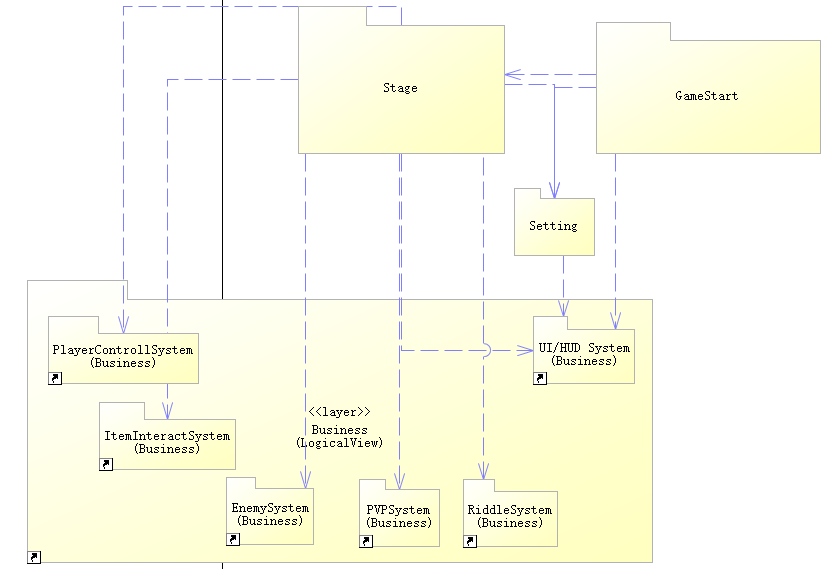


图3：应用层构成

## 业务层

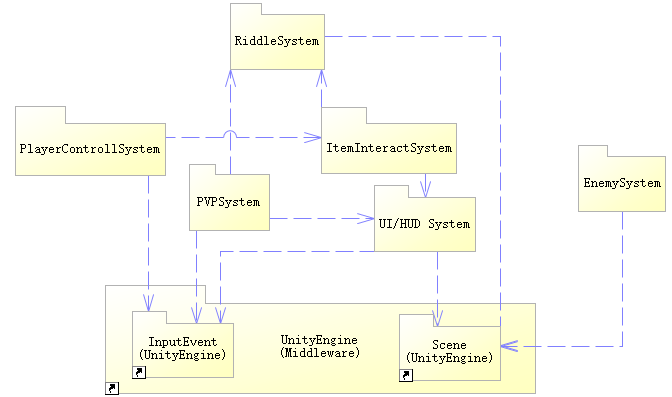
业务层由以下包组成：

图4： 业务层主要构成

业务层占了全系统的主要部分，有两个系统处理玩家的行为所产生的业务，有四个系统维护关卡的运行

### 角色操控系统（Player Control System）

PlayerBehavior负责所有角色操控处理，也就是实际的坐标移动与脚步声播放等随之而来的副效果。

### 道具互动系统（Item Interact System）

Raycast计算角色正在看向的物体，Item描述可互动物体的数据与互动内容。

### 鬼AI系统（Enemy System）

Chase计算鬼的行为逻辑与寻路，并通过noticeSound(int soundLevel, vector3 position)接受任何的声源信息，能听到时且还看不到逃跑者时会跟踪声源。

### PVP系统（PVP System）

PVPManager管理双人模式下的房间、胜负处理。

### UI/HUD系统（UI/HUD System）

InventoryUI/InventorySlot显示背包栏，负责物品的查看。

## 中间层

#### 中间层仅有Unity存在，主要使用它的输入事件监听功能（Input Event）和场景功能(Scene)。

# 部署视图

由于我们的项目主体是一个单机游戏，因此正常的单人游玩模式的部署视图非常简单。

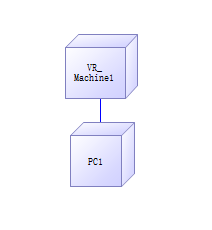


图5：单人模式部署视图

双人模式的部署视图较为复杂一些，但是总体实现的是两个主机之间的一个联机，并主要是通过PhotonPUN Server来实现。

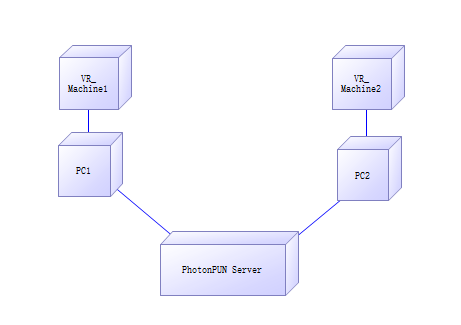


图6： 双人模式部署视图

# 实现视图

EndlessMemory的实现视图主要由Appl组件、Ghost组件、Backpack组件、Character组件、GUI组件组成。

Appl组件负责接收其他组件传来的指令；

Ghost组件会输出Ghost的转动视角、切换视角、初始位置、投掷物品等指令；

Backpack组件会输出背包的查看物品等指令；

Character组件会输出人物操控、场景交互、查看背包等指令；

GUI组件会输出开始游戏、退出游戏、游戏设置等指令；

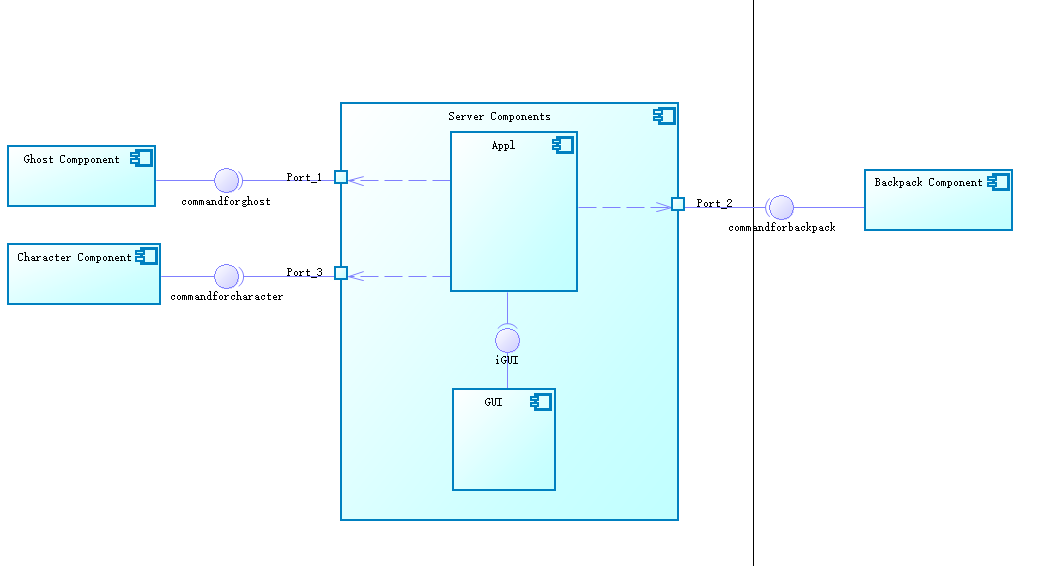


图7： 实现视图

# 设计战术

|  |  |
| --- | --- |
| 质量因素 | 设计战术 |
| 易用性 | 1. 会在游戏内容内设计新手教学内容 |
| 可用性 | 1. 错误预防，会在release之前进行内测来确保release版本的稳定性 |