系统设计说明书

目录

1 引言

1.1 编写目的

编写本系统设计说明书旨在描述本项目“饱满骑士”的系统结构和内部设计，内容包括：系统运行环境设计，主要描述了硬件设备、运行环境和开发环境；软件结构设计，主要描述了软件总体结构、软件技术架构和软件功能结构设计；类图设计；功能模块设计，主要描述了功能模块设计和层次图；接口设计，主要描述了接口设计说明；系统安全和权限设计，主要描述了安全性和权限设计说明。

本文档可作为:

(1) 向用户描述本项目功能的依据。

(2) 开发人员进行详细设计和编码的基础。

(3) 软件测试的依据。

1.2 读者对象

本系统设计说明书的预期读者为用户、本项目小组的开发人员、测试人员、PM 和未来的系统维护人员。

1.3 参考文献

[1]邹欣.构建之法[M].3 版:人民邮电出版社,2014.

1.4 文档约定

文档规定主要是指本系统设计说明书的排版约定，正文风格为:

标题 1:宋体，二号，粗体;

标题 2:宋体，三号，粗体;

标题 3:宋体，四号，粗体;

正文:宋体， 小四号;

列表:宋体，五号;

行距: 1.5 倍行距。

2 总体设计

2.1 系统运行环境设定

2.1.1 运行环境

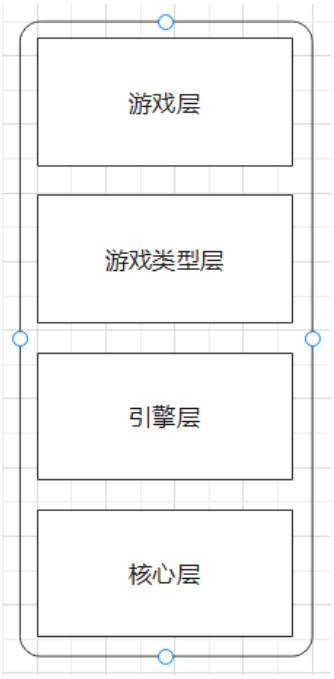
平台	发行厂商	版本号	用途
windows10	微软公司	1909	运行游戏

2.1.2 开发环境

序号	软件平台名称	发行厂商	版本号	补丁包版本号	用途
1	unity	Unity	2020.3.2	-	游戏对象编写
2	Visual Studio	微软公司	16.7.2	-	c#脚本编写

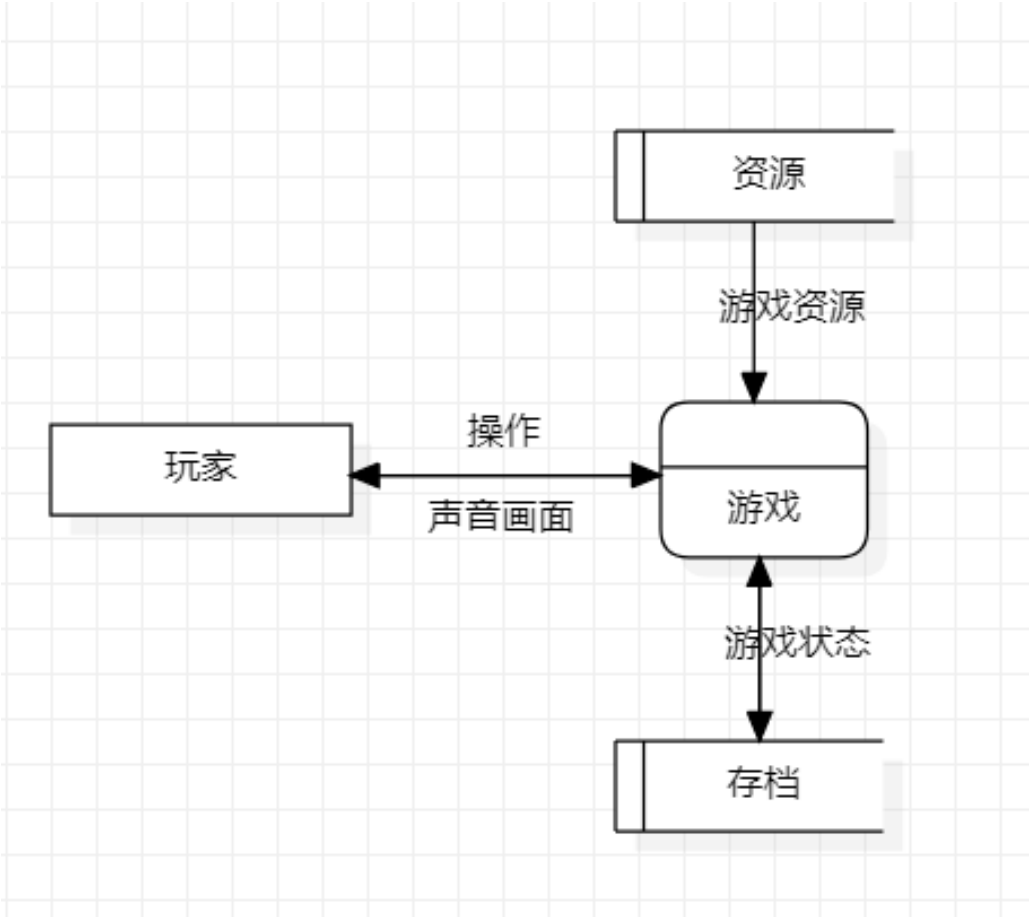
2.2 软件结构设计

2.2.1 软件总体结构设计

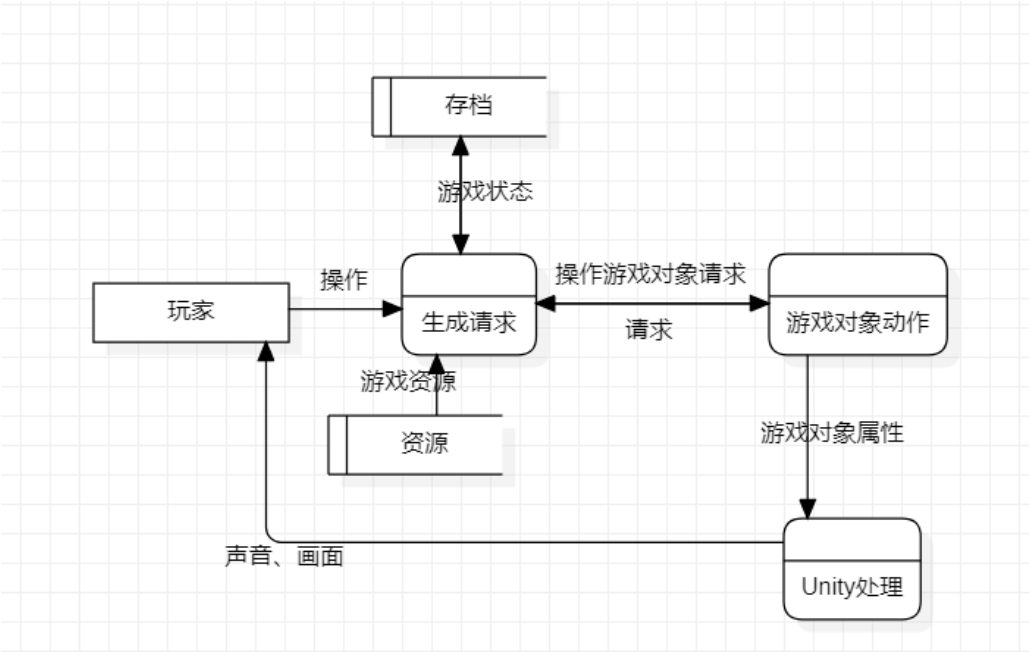


按照一般的游戏结构层次划分，我们的游戏可以先大体分为核心层（Core），引擎层（Engine），游戏类型层（Game Genre），游戏层（Game），一般的商用游戏引擎就用到了引擎层和核心层，而引擎使用者一般都是实现游戏层和游戏类型层。这种分层的架构设计就可以帮助我们把复杂的系统进行解构，从而实现每个子系统或者模块的功能单一化。

但是在实际使用中，我们组则是通过这样的思路，按多层次结构设计了一个核心架构，每个层次都可以更加细致的分派不同的人来负责，该架构分为五层：



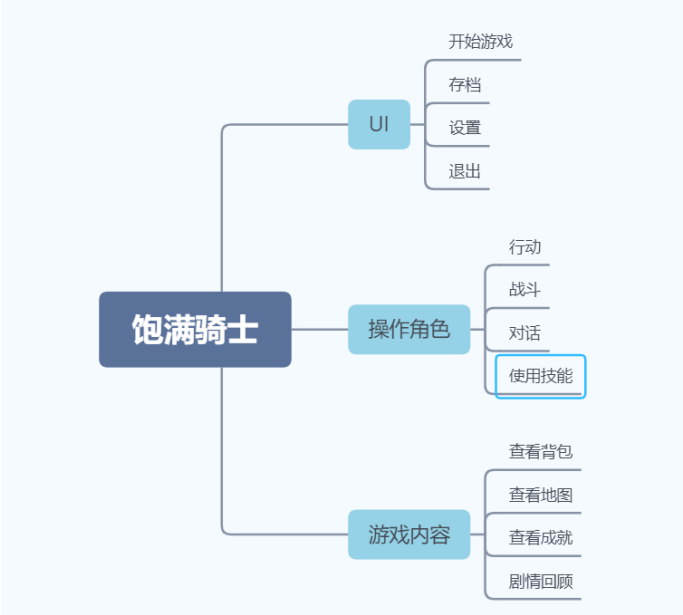
数据流图



数据流图

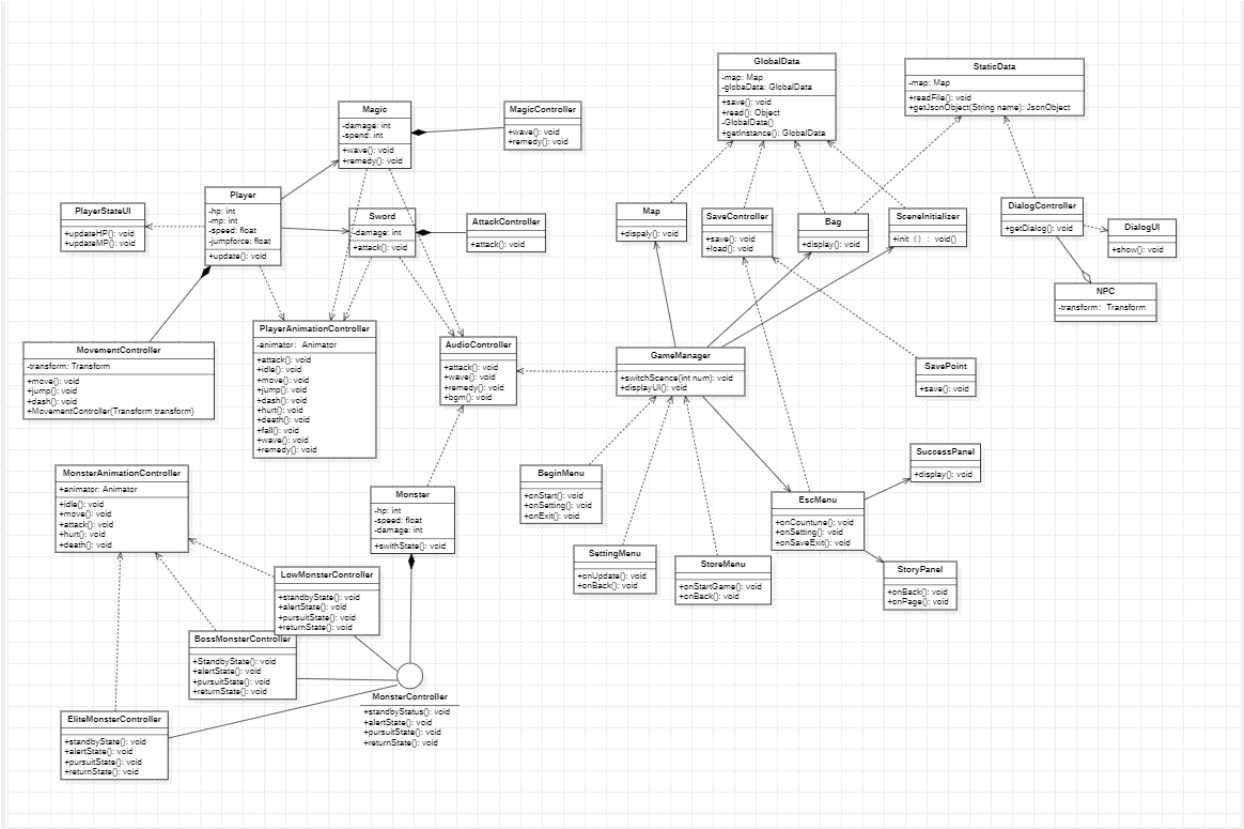
2.2.2 软件功能结构设计

本项目的主要功能有 UI、 角色控制、 游戏内容三个模块， 其中又包含多个子模块， 如下图。 其中 UI 包括开始游戏，存档，设置，退出， 角色控制则是关于控制角色移动战斗交互等的模块， 游戏内容模块有剧情，背包，地图，成就元素，增强游戏可玩性。



功能结构设计图

2.3 类图设计



类图的设计按照系统体系结构分为四个部分：信息层、决策层、行为层、表现层。信息层包括 SceneInitializer、GlobalData、StaticData 类，用来获取游戏信息，传递给决策层，包括 Player、Magic、Sword 等类，负责状态控制，指挥行为层（Controller）行动，控制表现层显示 UI 界面。

信息层

SceneInitializer 场景初始化器

方法：init（初始化）

GlobalData

属性：map（数据映射对象）GlobalData（全局数据对象）

方法：save（保存）read（读取）

StaticData

属性：map（静态数据映射对象）

方法：getJsonObject（获取数据对象）readFile（读取文件）

决策层

读取键盘/鼠标数据，调用对应的 controller 进行响应

Player

属性：hp（生命），mp（能量），speed（移动速度），jumpforce（跳跃力）

方法：update 实时读取键盘/鼠标数据并调用 controller

Magic

属性：damage（伤害量），spend（消耗能量数量）

方法：wave（远程攻击），remedy（回血）

Sword

属性：damage（伤害量）

方法：attack（攻击）

GameManager

方法：

switchScene（场景切换）

displayUI（UI 展示）

Map

方法：显示地图

Bag

方法：显示背包

读取游戏场景上的状态变化改变自己的状态，调用对应的 controller 进行响应

Monster

属性：hp（生命），speed（移动速度），damage（伤害量）

NPC

属性：transform（位置信息）

SavePoint

属性：transform（位置信息）

行为层

MovementController

属性：transform 玩家信息

方法：jump （跳跃），move （移动），dash （冲刺）

AttackController

方法：attack （攻击）

MagicController

方法：wave（远程攻击），remedy（回血）

MonsterController（接口）

方法

standByStatus（站立状态切换）

alertState（警戒状态切换）

pursuitState（追击状态切换）

returnState（返回状态）

具体怪物实现类 EliteMonsterController，继承 MonsterController

PlayerAnimationController

属性:Animator （动画对象）

方法：

attack （播放攻击动画）

idel （播放站立动画）

move（播放移动动画）

dash（播放冲刺动画）

hurt（播放受伤动画）

death（播放死亡动画）

fall（播放坠落动画）

wave（播放远程攻击动画）

remedy（播放回复动画）

MonsterAnimationController

属性:Animator （动画对象）

方法：

attack （播放攻击动画）

idel （播放站立动画）

move（播放移动动画）

hurt（播放受伤动画）

death（播放死亡动画）

AudioController

方法：

attack （播放攻击音效）

wave（播放远程攻击音效）

remedy（播放回复音效）

SaveController

方法：save（保存游戏）load（加载游戏）

DialogController

方法：getDialog （获取对话）

表现层

BeginMenu

方法：

onStart（游戏主菜单加载）

onSetting（选项菜单加载）

onExit（游戏关闭）

SettingMenu

方法：

onUpdate（游戏设置更新）

onBack（返回上级菜单）

StoreMenu

方法：

onStartGame（开始游戏）

onBack（返回上级菜单）

EscMenu

方法

onSetting（选项菜单加载）

onCountue：（继续游戏）

onSaveExit：（保存并退出）

PlayerStateUI

方法

updateHp（更新血条 UI）

updateMp（更新能量条 UI）

DialogUI

方法

show（显示对话）

StoryPanel

方法

onBack（返回上级页面）

onPage（翻页）

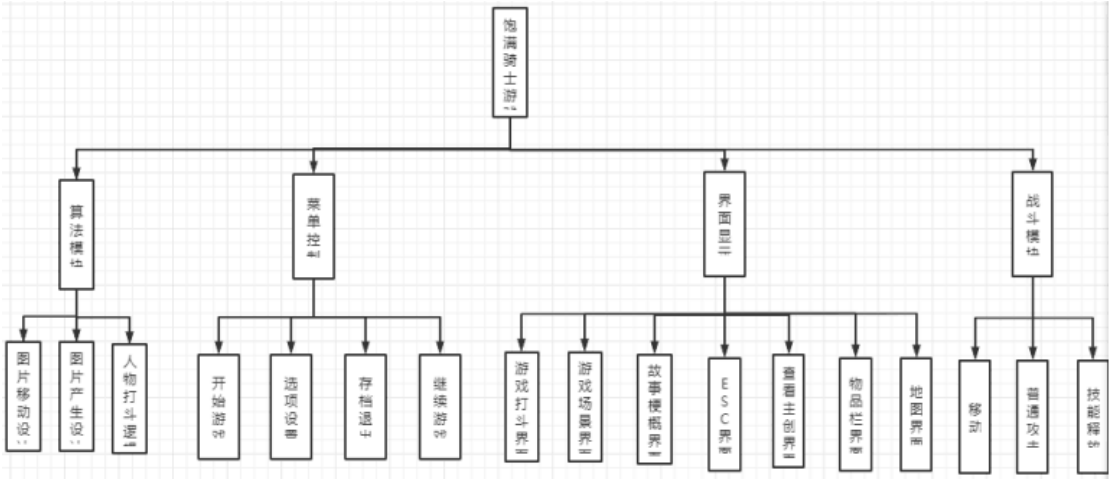
SuccessPanel

方法

display（展示成就页面）

3 功能模块设计

3.1 功能模块层次图



功能模块层次图

3.2 功能模块设计

1、算法模块

（1）图片移动设计

（2）图片产生设计

（3）人物打斗逻辑

2、菜单控制

（1）开始游戏

（2）选项设置

（3）存档退出

（4）继续游戏

3、界面显示

（1）游戏打斗界面

（2）游戏场景界面

（3）故事梗概界面

（4）ESC 界面

（5）查看主创界面

（6）物品栏界面

（7）地图界面

4、战斗模块

（1）移动

（2）普通攻击

（3）技能释放

3.3 功能模块描述

1、算法模块

（1）图片移动设计

在判断人物图片是否移动的逻辑上，如果玩家按了有效的移动按键（包括上

下左右），则人物图片则可以相应地移动

（2）图片的产生设计

在玩家又再次攻克一个关卡以后，则地图则可以立刻标记那个关卡点，并在地图界面显示

（3）人物打斗逻辑设计

其中怪物发射技能的时间间隔采用定时器算法，也就是指定具体的某一个时间值，怪物到指定的时间点后就可以开启攻击；而怪物采用技能攻击则是通过伪随机算法来实现，怪物是否开启技能攻击是随机发生的；怪物攻击的范围则是通过有限状态机做出响应。

2、菜单控制模块

（1）开始游戏

进入游戏的主界面，有“开始”选项、“选项”选项、“主创”选项、“退出”选项，点击“开始”即开始游戏，进入存档界面

（2）选项模块

进入“选项”界面可以设置音效、设置各个按键，包括向左、向右、向上、向下，和动作按键，包括跳跃、突进、抓住、交谈

（3）存档退出

在“ESC”界面中点击“保存游戏并退出”则可以存档，下次在存档界面会有该存档记录。

（4）继续游戏

在“ESC”界面点击“继续游戏”则可以继续游戏

3、界面显示模块

（1）游戏打斗界面

人物与怪物打斗的场景，玩家可通过选项界面中的按键来攻击怪物，动作按键包括突进、抓住、跳跃、交谈

（2）游戏场景界面

玩家可通过移动按键（包括向上向下向左向右）来在游戏场景中移动

（3）故事梗概界面

玩家按“ESC”键进入 ESC 界面，点击选择“故事梗概”会出现原型设计界面中的“故事梗概”界面，待故事梗概呈现结束，玩家选择“下一步”进入游戏场景界面

（4）ESC 界面

玩家点击“ESC”按键显示 ESC 界面

（5）查看主创

点击主界面既可以查看主创人（界面暂时没做出来，期待后面成品）

（6）物品栏界面

用户按下按键“B”可查看背包，出现原型设计界面中的“背包界面”

（7）地图界面

玩家点击按键“M”进 rf 入原型设计界面的“地图界面”可查看地图，查看地图概貌和自己所处位置

4、战斗模块

（1）移动

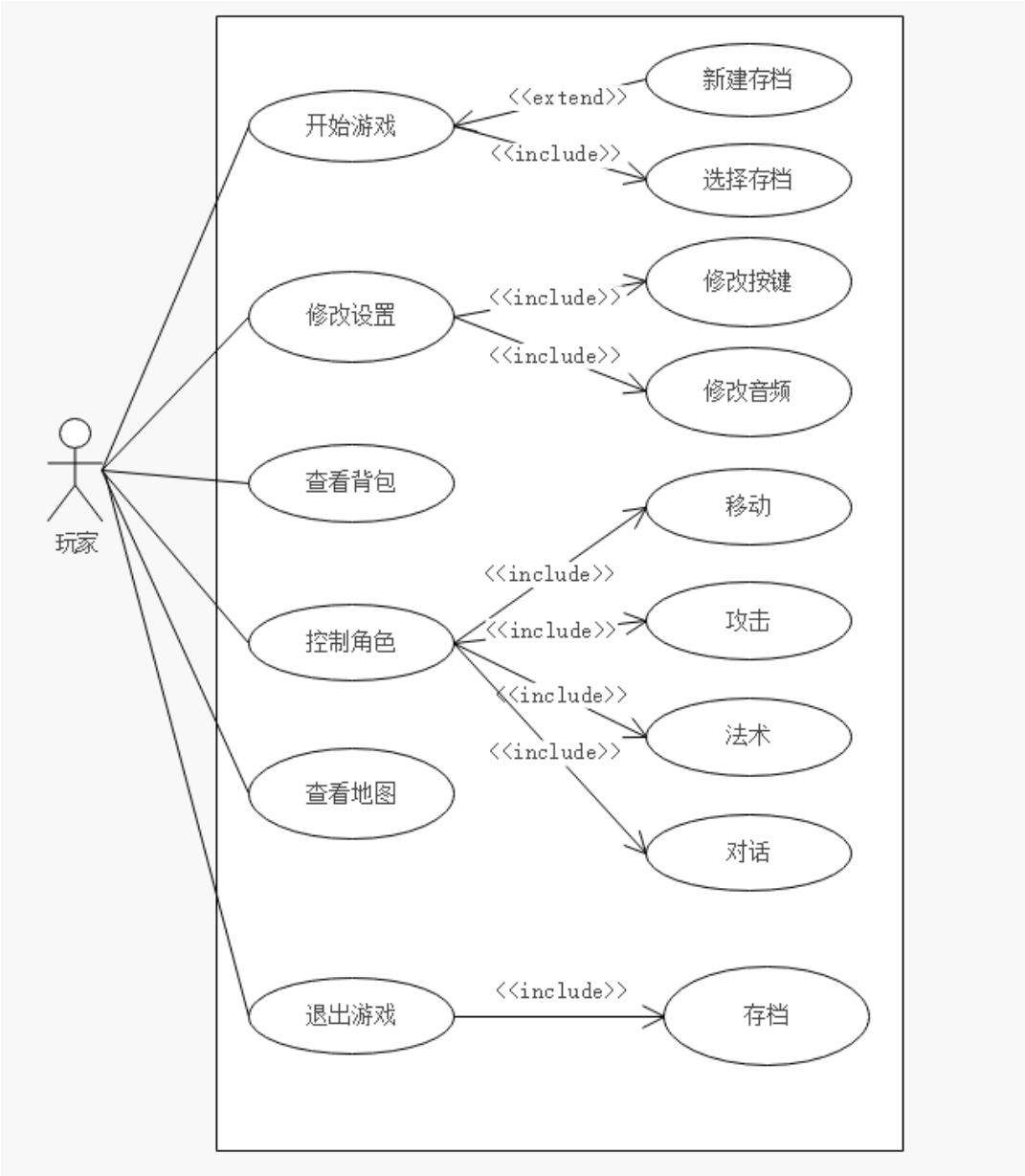
玩家根据选项界面中的上下左右和跳跃按键来进行移动，其中上下左右和跳跃可以搭配使用

（2）普通攻击

玩家通过按键“J”可以采用普通攻击

（3）技能释放

玩家通过“u/i/o”按键来释放技能



用况图

4 接口设计

数据交互

键盘输入	响应
w	向上
a	向左
s	向下
d	向右

j	攻击
k	跳跃
l	冲刺
u	远程攻击
i	回复

游戏数据格式

静态数据

```
DialogData:{
    “id”:”标识符”,
    “data”:”字符串”
}
ItemData:{
    “id”:”标识符”,
    “data”:{
          “name”:””,
          “details”:””
       }
}
```

存档数据

```
SaveData:{
    “id”:”存档标识”
    “data”:SaveObject
}
```

5 系统安全和权限设计

5.1 安全性设计说明

使用 Unity 引擎开发游戏，易上手，效率高，成本低。Unity 引擎带来便捷和高效的同时，也引入了一些新的安全风险。游戏逆向分析的门槛也被大幅降低，分析人员的差距被拉平，即使没有太多逆向经验的人，也可以开始尝试制作外挂了。相比 C/C++ 游戏，Unity 游戏的安全风险存在分析工具多、破解门槛

低、攻击方法通用化等几个特点。

Unity 支持 Mono 和 IL2CPP 两种编译模式。使用 Mono 模式编译的游戏，会将 C# 脚本代码编译为 IL 中间码，发布到游戏客户端。也就是源代码会泄露，这种中间码安全性较低，可以被一键反编译，一键修改。后来基于安全性和执行效率方面的考虑，Unity 支持了 IL2CPP 编译，大大提升了游戏安全性，但还是存在被攻击的风险。

因此我们团队决定使用 IL2CPP 模式编译我们的脚本代码，使用 IL2CPP 模式编译，游戏的脚本代码没有了，脚本代码被编译成了 Native 代码发布。很多反编译的工具都失效了，安全性得到了一定的提升。

5.2 权限设计说明

1. 游戏的启动、暂停与退出权限
2. 主角操纵权限：玩家可以操纵游戏主角的移动、攻击和发动技能
3. 快捷键设置权限：玩家可以根据自己的习惯更改对主角操控的快捷键设置。
4. 背景音乐和音效设置权限：玩家可以更换背景音乐和音效
5. 查看地图权限：玩家可以查看地图了解自己的位置等信息
6. 对话权限：玩家在相应情景下可以和游戏对象进行对话交流以了解闯关相关信息。
7. 查看背包栏权限：玩家可以查看背包栏了解自己已经获得的物品。
8. 进入存档权限：玩家可以选择一个存在的存档并点击进入。
9. 创建存档权限：玩家可以创建一个新的游戏存档。
10. 删除存档权限：玩家可以删除一个存在的存档。
11. 故事梗概的查看权限：玩家可以打开故事梗概了解大概的故事。