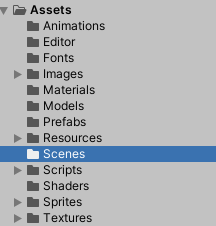
一、整体架构

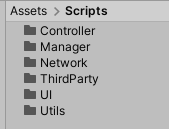
整体以Unity3D引擎2019.3版开发客户端，以python2.7为工具开发服务端

1. 客户端项目文件结构如下：

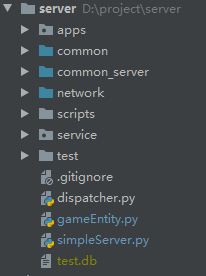


Assets中包括了场景，脚本，预设，纹理，材质以及动画等各种资源文件

其中场景包括welcome和main两部分构成，脚本按照Controller、Manager和Network等分功能设计



1. 服务端项目文件结构如下：



服务端与客户端相似，按照不同分工设计出user、player、room和game等类模块，结合提供的RPC框架辅以AI与服务多线程。

二、服务端

服主要分为以下几个模块：

1. simpleServer.py，服务器主模块，主要负责游戏所有entity的管理以及游戏服务器的主流程的运行。
2. common/目录, 设计了user\_model类，作为用户模型，用于和SQLite数据库交互，储存玩家账户以及游戏角色的信息
3. network/目录，框架中的网络通信模块，管理已经连接的客户端以及消息的收发
   1. netStream.py: 发送接收数据的底层模块
   2. simpleHost.py: client的管理模块
4. gameEntity.py, 服务器端的entity的类模块，各个entity在游戏服务器中管理,负责rpc Call的消息分发，调用各个类去完成游戏逻辑
5. scripts/目录， 用于生成初始db数据库和插入测试用户数据
6. test/目录，用于初期测试服务端与客户端的网络通信
7. test.db 作为用户数据库，提供user 和 character两张表
8. apps/目录,服务端游戏逻辑主模块
   1. user.py处理登录注册
   2. play.py 处理玩家行为，包含playerPool、player和game\_status由宏观到细节三个层次,分别负责玩家群里管理，玩家个体行为，和游戏中状态
   3. room.py 处理房间系统相关操作，包括roomPool和room两个层次
   4. robot.py 作为服务端AI主模块，通过状态机和B\*算法的实现去完成AI的寻路寻人、攻击和躲避等逻辑
   5. map模块负责维护地图数据和生成玩家以及随机物品

game

三客户端

代码都在Assets/Scripts目录下。

1. Manager/
   1. GameManager.cs: 客户端的主要游戏管理模块，存储大量用户属性和状态，通过传递服务端同步消息，调用其他脚本对场景中的GameObject进行相应的处理和收发消息
   2. MapManager.cs: 地图管理，通过读入txt恢复地图场景，0表示ground, 1表示 grass，2表示wall，此外还有随机出生的box，可以阻挡玩家通过
   3. PlayManager.cs: 游戏主场景中玩家对象的管理和状态同步
2. Controller/
   1. CameraController.cs 负责3D俯视视角跟随
   2. ArrowControlle.cs负责射出弓箭的碰撞检测与消息发送
   3. PlayerController.cs 负责玩家的行为与消息同步
3. UI/
   1. Login 登录注册
   2. Lobby 游戏大厅
   3. Room 房间内
   4. Result 游戏结算
   5. Hero 英雄升级

其他资源如prefab，可变化的资源由Resources存储并调用

四．作业情况

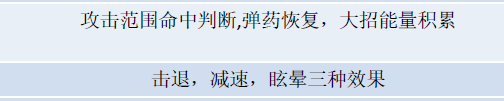
1. 服务端



完成



逻辑完成 但寻路有bug,会卡死



部分与地图元素的碰撞检测完成，大招与弹药回复完成，击退效果完成



没有毒圈



完成



合理管理，基本流畅

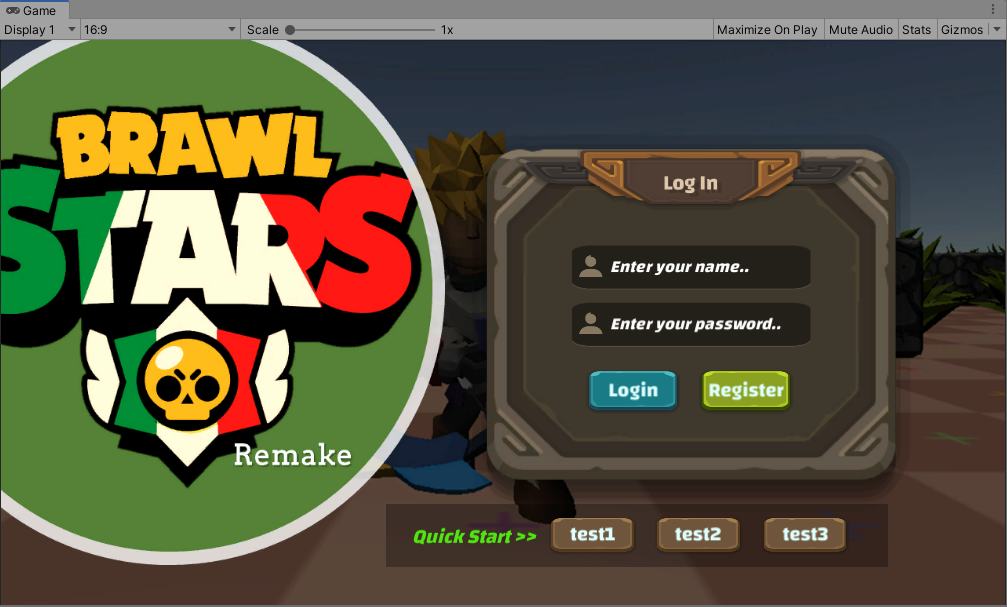
2. 服务端



除投弹手与坦克的技能，植被隐身，毒圈外均有实现

1. 游戏截图

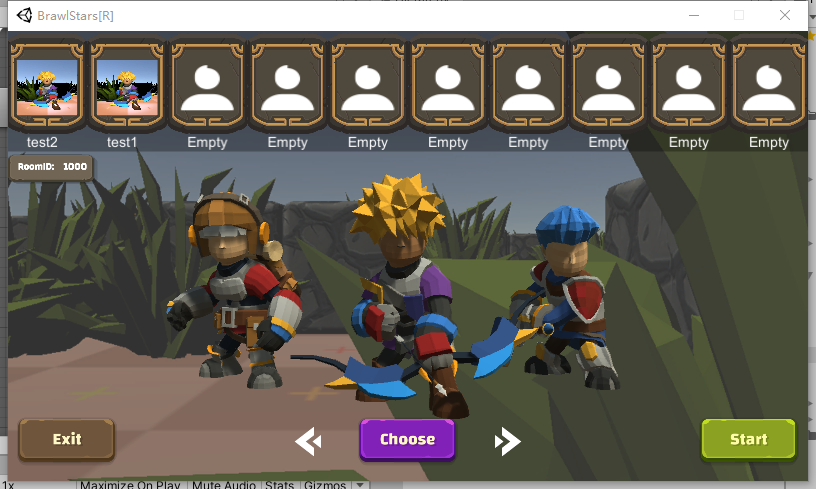
登录注册



大厅



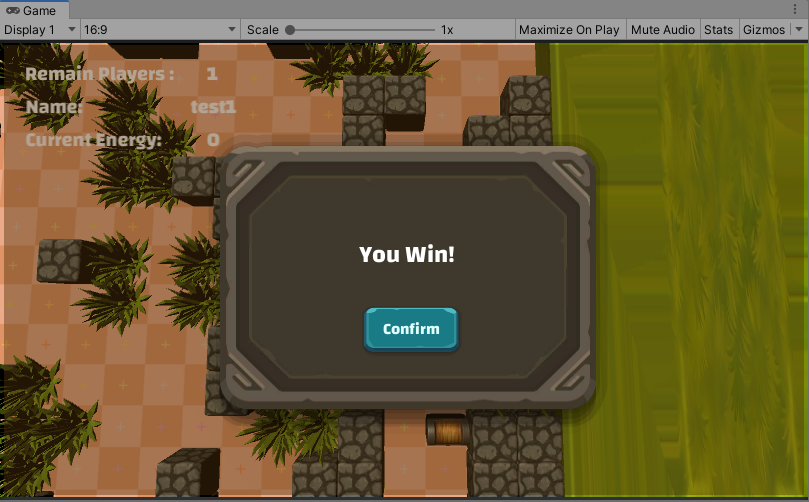
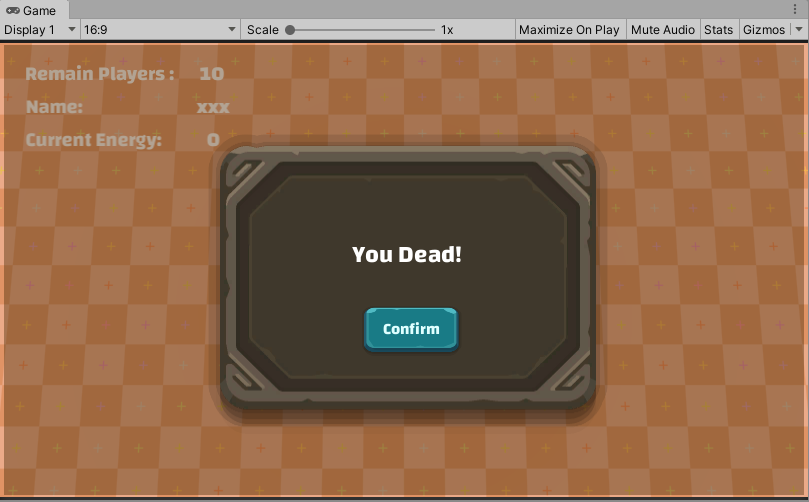
房间内



游戏主场景



胜败判断与结算





英雄升级

