# 功能实现

## 登录：

四个账号：netease、netease1、netease2、netease3

密码：123：

端口不用选，被我关掉了，因为我发现不绑端口会自动分配一个

如遇未知数据错误，可以将根目录下的user.csv放到server下覆盖。。。

## 游戏操作：

WASD用来控制角色移动，鼠标控制屏幕旋转

鼠标左键点击射击，有0.6sCD

鼠标右键发射射线，击晕怪物，有5sCD，可以击晕怪物3s

## 游戏功能

玩家死亡自动注销账号，死亡不损失经验和等级。

也可以点击左上角按钮自动注销

注销后可以选择任意一账号进行游戏，可以复原上次离开的现场（服务器不能关）

玩家信息可离线保存在服务器，可以随时从服务器获得玩家信息

射击基础伤害20，每次升级加5

武器基础弹药量200，每个弹夹20，可以单发和点射

玩家基础血量2000，每次升级加200

远程怪血量100，伤害20

近战怪血量500，伤害1（每次collider触发为1，但是触发比实际看着频繁）

远程怪掉弹药包，加20弹药

近战怪掉血包，加40血

玩家和怪物都有基础动画，包括站立，攻击，跑动，跳跃，换弹，死亡

有简单的UI显示血量，弹药和射击模式（单发、点射）

服务器实现怪物寻路和刷新

同时只能存在五个怪，1近战+4远程

近战近距离开始攻击玩家，远程怪当与玩家距离为10时开始射击

## 代码主要实现逻辑：

### Server：

通信协议UDP，用以快速同步FPS游戏中的数据帧

采用帧同步的方式，client端计算数据，传给服务器，由服务器分发给所有人

没用sqllite,用的csv传数据。主要是csv的看着方便点。。。

游戏服务器只有一个逻辑服务器，没有像实际的项目中那样分布式管理

没调用其他库，就没用rpc和proto。用的UDPsocket传数据，然后server将数据解析为一个dict，根据dict的calling字段，采用remoteCall来调用不同线程的服务，其实也跟rpc原理差不多。。。

服务器使用queue进行了线程的数据同步，mainThread通过解析udpSocket，获得对应的Dict（其实就跟RPC传proto一个原理。。。），根据calling分发给不同thread中的queue进行处理。在子线程中循环的获取queue中的数据来执行client端的请求

在server端开了几个不同的子线程用以处理不同的任务，主要有LogInServer，用以处理登录、登出、注册等信息，并进行数据的持久化；LogicServer，用以处理游戏中的基本逻辑和数据，其实也没啥逻辑，logicServer就主要用来更新服务器端的玩家信息，再分发给其他线程使用了；EnemyLogicServer，用以控制怪物刷新、攻击、寻路、死亡和数据同步，这里没有用A\*，就是计算了下欧拉距离，然后用client的nevMesh进行寻路了。

### Client：

客户端分为两个sence，一个用来登录，另一个用来进行游戏

实现了单例的Client类用以进行客户端通信，启动一个子线程接受数据，主线程用来根据具体功能需要发送数据

LogicMain为单例类，主要用于客户端保存游戏现场，包括玩家信息，怪物信息等基本游戏数据

Turner类为单例类，用以将server端发送的数据解析为一个dict

MessageConvert类为单例类，用以将游戏中的数据封装为统一标准的字符串进行发送

Net脚本为Login Sence的对应脚本，在Net中启动了client和logicMain，在登录时使用client.sendMsg进行登录。将client端子线程recv到的数据存入logicMain中用以记录游戏的基本信息

MainController为Game Sence的对应脚本，主要负责任务的分发，以及根据收到server端的具体的信息进行游戏数据同步

User类用以记录玩家的所有信息，并在玩家的gameobject建立时进行数据同步

Charater为玩家对应的脚本，主要负责音效，CD计算，触发检测。还用来接受输入控制玩家的动作，通过movement类来进行了移动。玩家在fixedUpdate中进行了玩家信息同步到client

设计了Enemy基类用以描述游戏中的所有怪，实现了受击音效、怪物血条，移动，血量计算，死亡等逻辑，对怪物进行了统一的碰撞检测，并将怪物数据同步到服务器中。怪物统一为Enemy层级，互相不进行collider检测，

Soilder和Zombie分别为近战怪和远程怪的脚本，主要负责对应动画的播放和攻击。Zombie因为为近战怪，为怪物的手部添加了mesh collider，在zombie脚本中进行了mesh的动态同步用以检测碰撞（感觉这样有点费，不知道是不是因为面有点多。目前还没想到更好的办法）

设计了Weapon基类用以描述游戏中的武器-----枪，weapon中有基本的功能，射击CD，音效，soilder怪的武器就是用的weapon类，玩家的武器PlayerWeapon继承于weapon，主要是多了点射和眩晕射线biubiubiu

CameraController用以控制相机

EnemyController结合GameEvent进行了怪物死亡的一系列处理，包括数据同步，奖励展示，销毁物体等。

本来最初设计GameEvent是想通过Trigger的Register和Unregister的方式实现所有的client内部的事件，通过event来统一管理，结果后来喝了假酒，写跑偏了，都写在mainController中了。。。

相关UI脚本控制对应的UI显示，其实我应该写在一起干净些的。。。

## 后续优化方案：

1. 整理client端代码，将一些可以合并和抽象的类进行整理
2. 对应任务分发逻辑的修改，每个脚本负责注册GameEvent中的Trigger，maincontroller负责对gameEvent中的所有任务进行分发，交给不同的子模块进行处理。各个子模块进行相关的数据运算，交给client类进行数据发送，交给服务器
3. 两个小怪的模型还不太好，在近战怪身上加了meshcollider后感觉很费，想办法优化下
4. 一些数据运算放在服务器端进行，以优化客户端代码，并提升游戏的数据安全性
5. 帧同步有点费，服务器试着导入相关寻路算法，给怪物下指令，将消息同步变为指令序列，以替代怪物的帧同步，以减少网络负担
6. 开发多人pve功能