C++程序设计实验报告-宠物小精灵对战系统

318-薛锦隆-2018210990

1. 任务描述

用面向对象的设计方法来设计一款平台类对战游戏。

1. 运行环境
2. 系统：win10
3. 编程：5.14.2
4. 数据库：sqlite3
5. 不同阶段实现的功能

版本1：宠物小精灵的加入

1. 设计宠物小精灵的类
2. 小精灵的属性、等级、特殊技能、攻击方式、不同属性的小精灵的不同升级

版本2：用户注册与平台登录

1. 实现socket通信（C/S模式
2. 使用数据库sqlite3存储所有用户数据，并可以在server端查看所有用户数据
3. 实现client客户端的交互，处理不同事件
4. 为每个注册用户分配3个小精灵，为登录用户加载所属的所有小精灵
5. 新增登录界面，在登录或注册后可以进入主界面
6. 为小精灵添加小精灵图片，可以直观看到自己的所有小精灵的样貌

版本3：游戏对战的设计

1. 设计与服务器进行的虚拟决斗，有升级赛和决斗赛2种形式
2. 在升级赛中，用户可以通过赢得比赛获取经验，并升级
3. 在决斗赛中，用户可以通过赢得比赛对方的小精灵，失败的话，会失去自己所有小精灵中的一个
4. 用户增加新属性，为宠物个数徽章（金银铜）和高级宠物徽章（金银铜）
5. 实现查看所有用户、在线用户的胜率和徽章情况
6. 美化界面，提高程序的完成度，增加趣味性

1. 具体实现
2. 参数设计
3. **enum** KIND   //pokemon的所有种类
4. {
5. HIGH\_ATTACK,
6. HIGH\_BLOOD,
7. HIGH\_DEFENSE,
8. HIGH\_SPEED
9. };
10. **enum** SKILL  //pokemon的特殊技能
11. {
12. fire\_attack,
13. water\_attack,
14. defense\_attack,
15. double\_attack,
16. normal\_attack   //所有精灵都有
17. }
18. **const** QList<QString> POKEMONNAME= {  //所有pokemon的名字（特殊技能；属性）
19. "Charmander",   //小火龙
20. "Charmeleon",   //火恐龙
21. "Charizard",    //喷火龙(fire\_attack; high\_attack)
22. "Squirtle",     //杰尼龟
23. "Wartortle",    //卡咪龟
24. "Blastoise",    //水箭龟(water\_attack; high\_blood)
25. "Diglett",      //地鼠
26. "Dugtrio",      //三地鼠
27. "Snorlax",      //卡比兽(defense\_attack; high\_defense)
28. "Dodrio",       //嘟嘟利
29. "Duduo",        //嘟嘟
30. "Pikachu",      //皮卡丘(double\_attack; high\_speed)
31. }

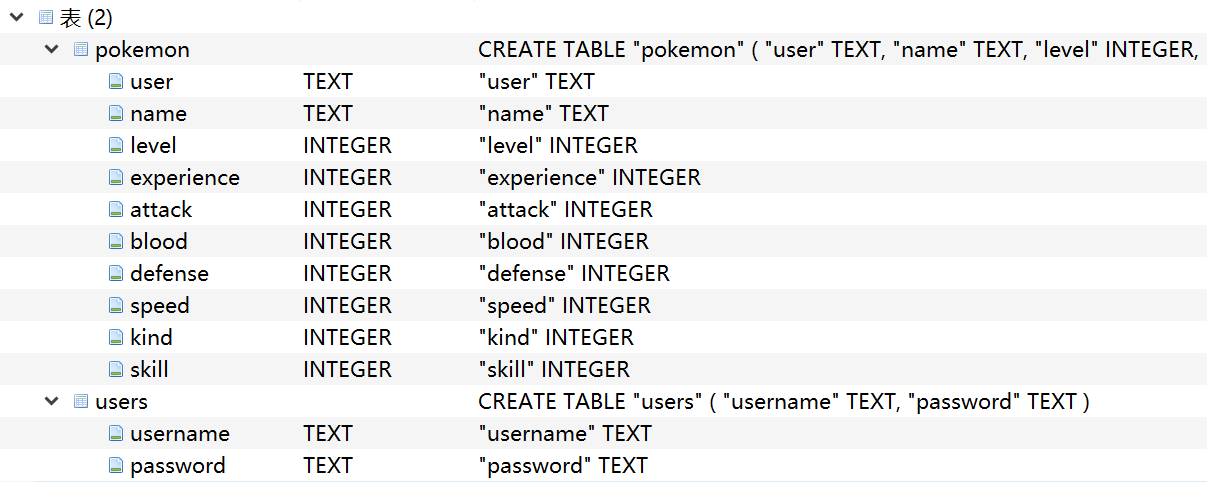
**enum** LOG\_TYPE  {SIGN\_IN, SIGN\_UP, NAME\_EXIST, SIGN\_UP\_SUCCESS, USER\_NONE, PWD\_ERROR, SIGN\_IN\_SUCESS, SIGN\_OUT, ALL\_USER, ONLINE\_USER,VIRTUAL\_PKM, PKM\_DATA}; //不同的发送的种类

1. #define MAX\_LEVEL 15  //最大等级
2. #define BASE 100   //基础值（不同种类的pokemon的初始化的基础值）
3. Pokemon基类和其子类的设计（为节省空间，没有列出所有子类、函数）
4. **class** Pokemon : **public** QObject  //pokeomon 基类
5. {
6. Q\_OBJECT
8. **protected**:
9. QString name;   //pokemon名字
10. unsigned **int** level;   //等级
11. unsigned **int** experience;   //经验值
12. unsigned **int** attack;   //攻击力
13. unsigned **int** blood;   //血量
14. unsigned **int** current\_blood;  //当前血量
15. unsigned **int** defense;   //防御力
16. unsigned **int** speed;   //攻击速度
17. unsigned **int** kind;   //类型
18. SKILL skill;   //特殊技能
19. **void** setValue(unsigned **int** base\_attack, unsigned **int** base\_blood, unsigned **int** base\_defense, unsigned **int** base\_speed);
21. **public**:
22. Pokemon(){};
23. **virtual** unsigned **int** Attack(){**return** 0;}
24. //每种精灵攻击方法不一样，为虚函数，需重写
25. **void** experienceUp(unsigned **int** value);   //增加精灵经验值
26. QString getSkill(){**return** ATTACKKIND[skill];}
27. QString getAllValue();   //返回精灵所有信息
28. **virtual** **void** levelUp(){};   //每种精灵升级的情况不一样，为虚函数，需重写
29. **void** setLevel(unsigned **int** set\_level);  //升级到指定等级
30. Pkm\* getAttr();
31. **virtual** ~Pokemon() {}
33. signals:
34. **public** slots:
35. };
36. **class** high\_attack : **public** Pokemon  //子类：如：high\_attack属性
37. {
38. Q\_OBJECT
39. **public**:
40. high\_attack(SKILL skill);
41. unsigned **int** Attack();  //重写攻击
42. **void** levelUp();     //重写不同种类的精灵的升级方式
43. ~high\_attack(){};
44. };
45. 数据库设计

创建pokemon.db，并创建两个表：pokemon，users。

* 1. pokemon表记录所有用户的详细小精灵信息（pokemon类中的属性）
  2. users表记录所有用户的用户信息（用户名、密码、胜场、败场）

下图为用DB Browser for SQlite查看数据库。



1. 用户类的设计（为节省空间，把不重要的函数省去）
2. **class** User : **public** QObject  //用户类，定义了用户的各种操作
3. {
4. Q\_OBJECT
5. **public**:
6. User();
7. **void** setUser(QDataStream& dsIn);     //根据传来的数据载入到用户中
8. **void** appendPkm(Pkm\* pkm){allPkmAttr.append(pkm);};  //得到一个精灵
9. **void** sendAllPkmAttr(QDataStream & dsOut);   //更新用户的所有信息
10. Pkm\* getPkmByIndex(unsigned **int** index){**return** allPkmAttr[index];};
11. **void** setUser(Pkm\*\* allPkm, unsigned **int** pkmNum);
12. //从login界面中传递的数据，加载到用户中
13. **int** attackOpponent(unsigned **int** index);
14. //用户的一个精灵发出攻击，返回攻击伤害
15. **void** addExperience(unsigned **int** index, unsigned **int** opLevel);
16. //用户的pokemon增加经验值（根据对方精灵的等级）
17. **void** levelUp(unsigned **int** index);  //用户的一个精灵升级
18. **void** popPkmByIndex(unsigned **int** index){allPkmAttr.removeAt(index);};
19. //失去一个pokemon
20. QString getUserBadget();    //返回用户徽章的情况
21. unsigned **int** win;   //赢场
22. unsigned **int** lose;  //败场
23. ~User();
25. **private**:
26. QList<Pkm\*> allPkmAttr; //用户的所有pokemon信息
27. QString username;   //用户名
28. unsigned **int** pkmNum;    //pokemon数量
29. };
30. 存放所有小精灵的数据结构
31. **struct** Pkm  //pokemon信息，与pokemon表一致
32. {
33. QString name;
34. unsigned **int** level;
35. unsigned **int** experience;
36. unsigned **int** attack;
37. unsigned **int** blood;
38. unsigned **int** current\_blood;
39. unsigned **int** defense;
40. unsigned **int** speed;
41. unsigned **int** kind;
42. unsigned **int** skill;
43. };
44. 服务器端
    1. 服务器端的接收端绑定45454端口，并设定为一旦有数据从端口传来，就执行processPendingDatagrams()函数，处理传来的数据。
    2. 使用QList创建在线用户的列表，记录所有在线用户的信息。并设定为有登录或注册添加进在线用户列表，有登出在所有在线用户中删除。

具体实现的函数：

1. **void** createPokemon(QString username, unsigned **int** create\_level);
2. //生成pokemon，写入pokemon数据库
3. **void** sendPekemon(QDataStream& dsOut, QString user);
4. //发送user的所有pokemon的信息
5. **void** sendAllUser(unsigned **int** port);
6. //从数据库返回并发送所有用户信息（用户名，所有精灵）
7. **void** setSql(QString user, Pkm\* pokemon);
8. //设置pokemon数据库的内容
9. **void** sendOnlineUser(unsigned **int** port);
10. //从online\_user中返回数据，并发送
11. **void** appendUserFromSql(QString username);
12. //在线用户增加，新增一个online\_user（登录或注册+1，退出-1）
13. **void** updateUserPkm(QDataStream& dsIn, QString username);
14. //更新用户在数据库的所有pokemon信息
15. 客户端

（1） 登录界面

实现用户登录、注册功能，并处理各种可能的错误，如：登陆时：密码不正确；注册时：用户名已经注册过；连接服务器超时等。

（2） 主界面

i. 实现展示用户pokemon信息，对方pokemon信息，所有用户、在线用户的详细信息（徽章、胜负情况）

ii. 实现对战机制，有两种比赛模式，在升级赛中可以通过赢得比赛获得经验值，在决斗赛中可以通过赢得比赛获得新的pokemon，但也有可能输掉比赛失去一只pokemon

iii. 可以在主界面和战斗界面切换，在主界面显示pokemon所有信息，在战斗界面可以看到比赛情况（如：血条、攻击、使出的技能）

具体实现函数：

1. QString getPkmAttr(unsigned **int** num);
2. //根据不同mode模式，返回pokemon信息(用于设置Text中的内容，为QString类型)
3. **void** setComboBoxValue(unsigned **int** Is\_my\_pkm);
4. //设置用户或admin的所有信息（combobox、label）
5. **void** updateUserPkm();   //在server端数据库中，更新user自己的pokemon信息
6. **void** showGameType();    //展示主界面
7. **void** hideGameType();    //进入对战模式，切换对战界面
8. **void** showGiveOut();     //决斗赛中，展示选择要送出的pokemon
9. **void** hideGiveOut();     //隐藏决斗赛中，送出pokemon界面
10. **void** fightBegin(unsigned **int** game\_type, User\* user, unsigned **int** userPkmIndex, User\* opponent, unsigned **int** opPkmIndex); //开始比赛
11. **bool** fightProgress(User\* user, unsigned **int** userPkmIndex, User\* opponent, unsigned **int** opPkmIndex); //比赛过程
12. **void** giveOutPkm(User \*user);   //送出一个pokemon
13. **void** delay(**int** time);   //模拟真实攻击间隔

**void** fightBegin(unsigned **int** game\_type, User\* user, unsigned **int** userPkmIndex, User\* opponent, unsigned **int** opPkmIndex); //开始比赛

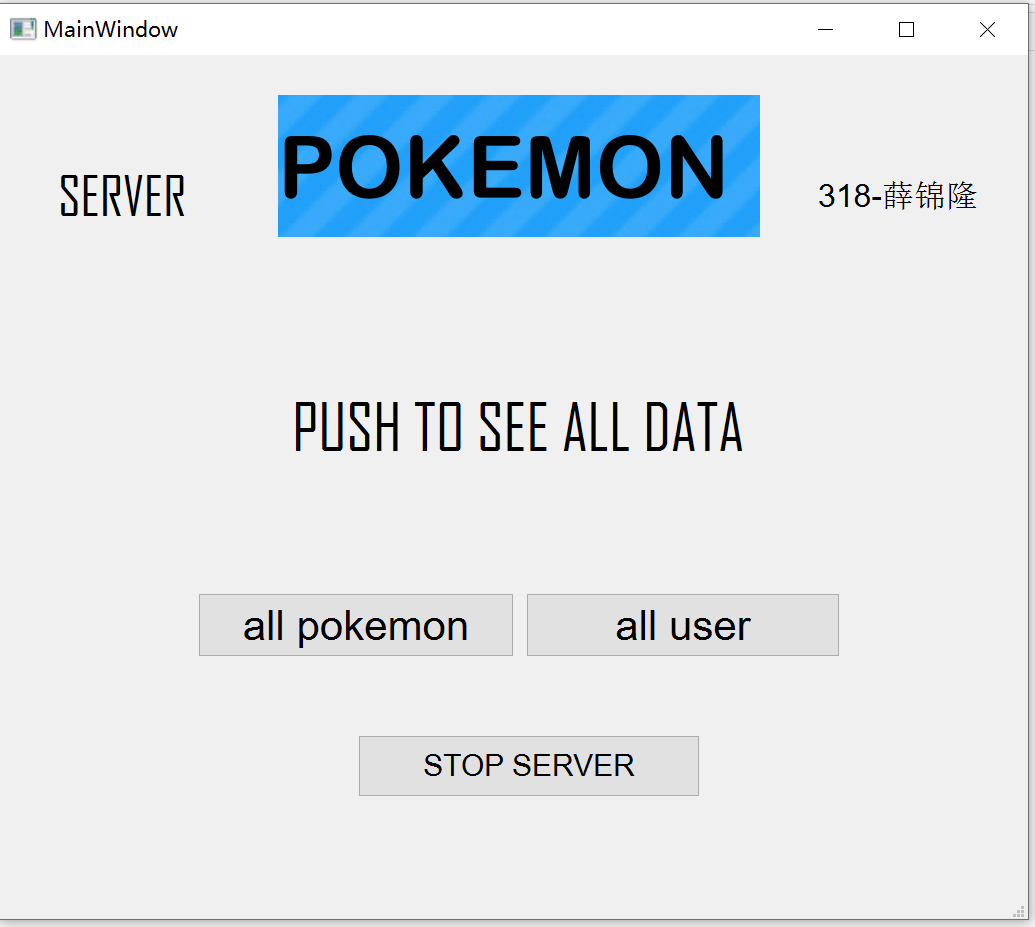
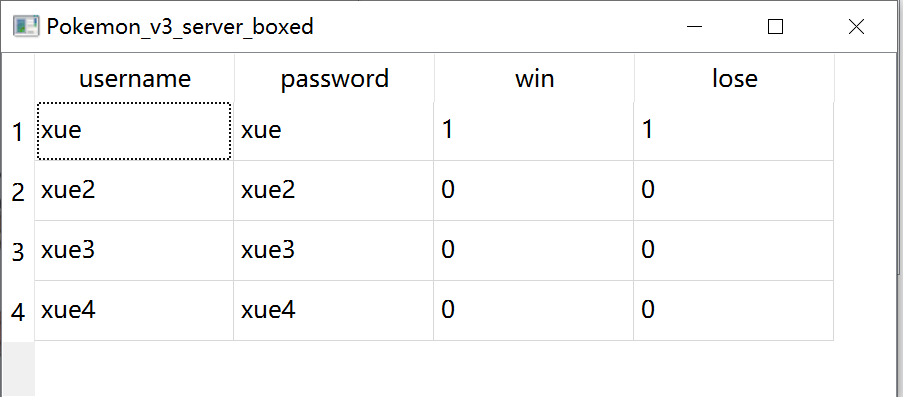
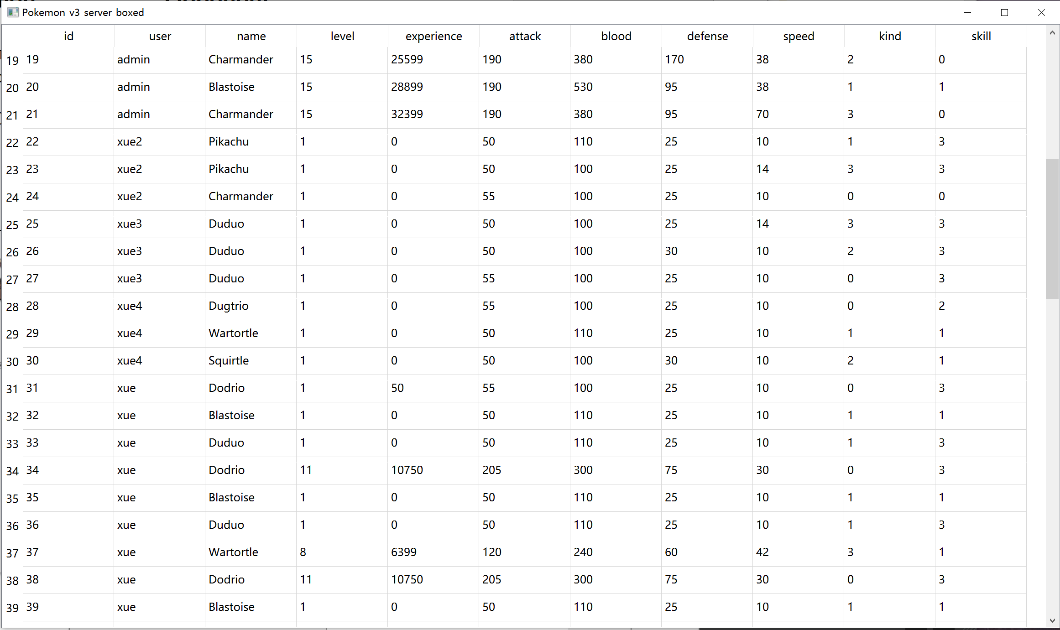
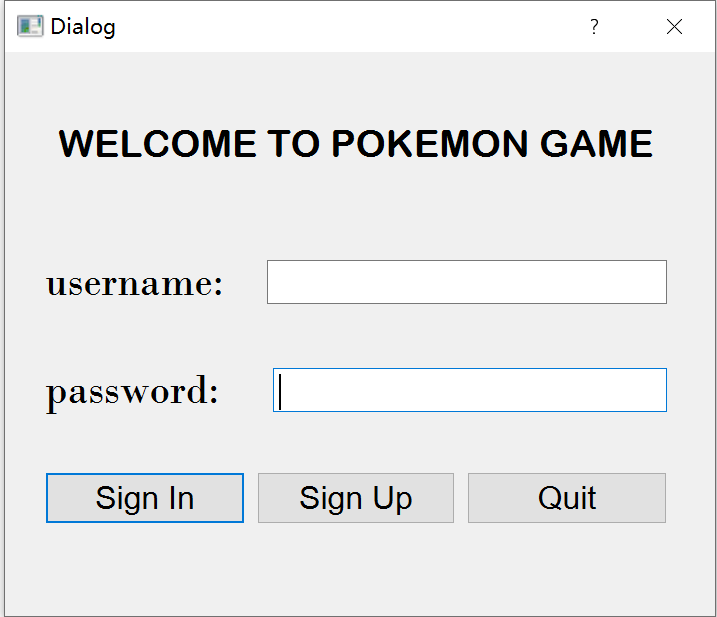
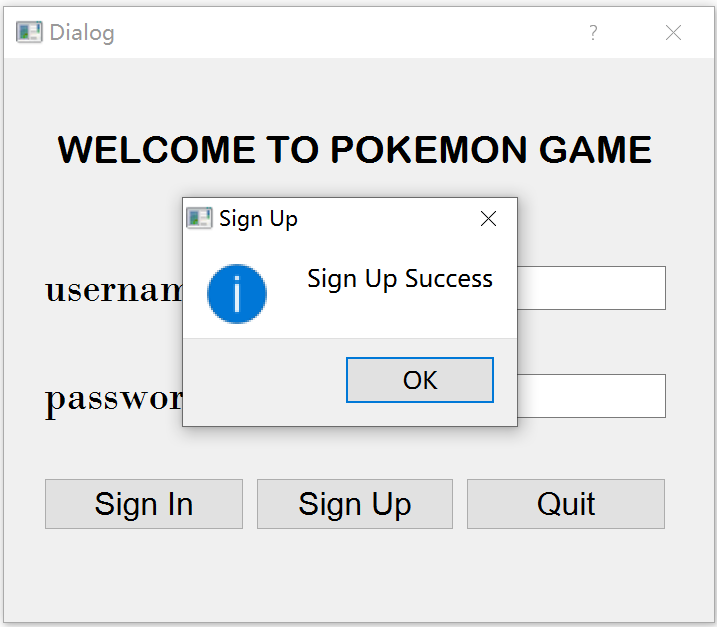
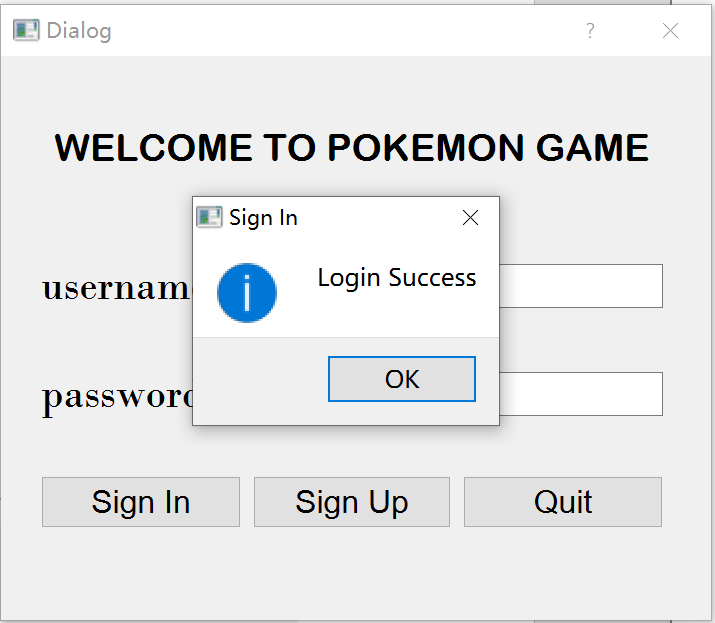
解释：

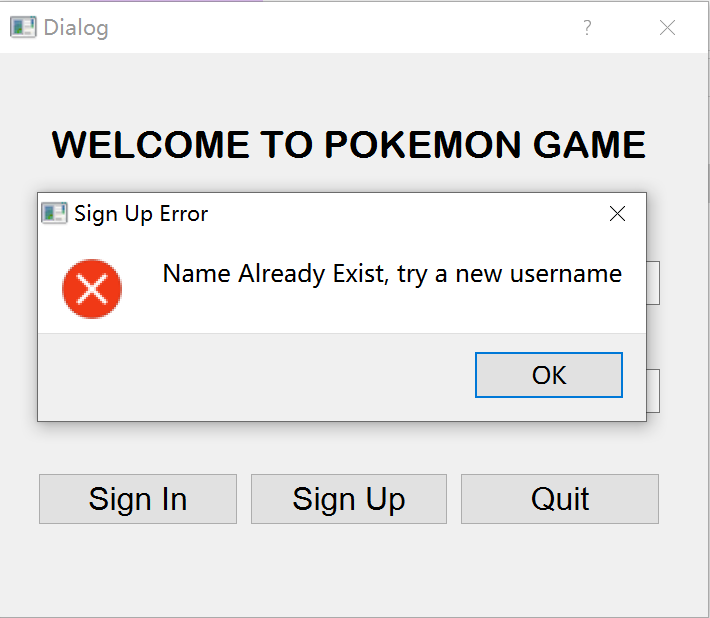
在fightBegin()函数中，通过game\_type进入不同的对战模式，然后进入fightProgress()函数，展示比赛的过程

**bool** fightProgress(User\* user, unsigned **int** userPkmIndex, User\* opponent, unsigned **int** opPkmIndex); //比赛过程

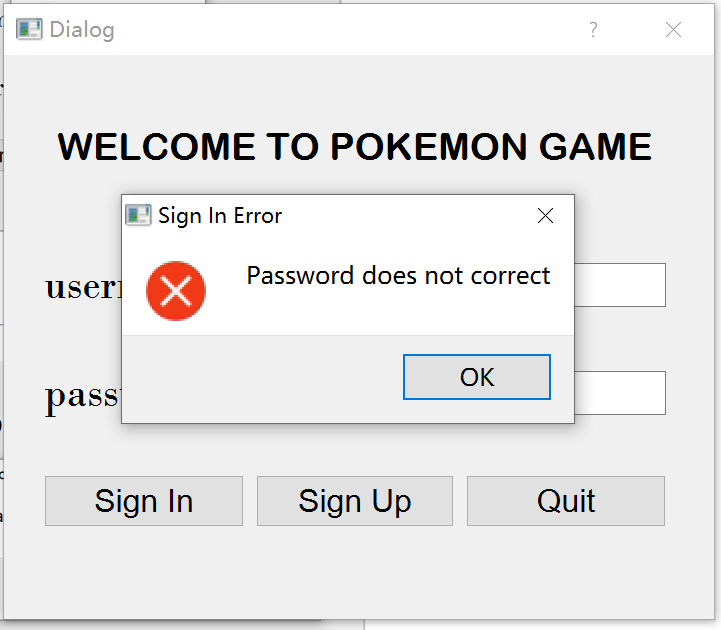
函数具体实现解释：

* 1. 使用delay(2)每回合停止2秒，模拟攻击间隔
  2. 本游戏为回合制游戏，但是我设置了speed属性，speed大的小精灵可以在一次回合中发动多次攻击
  3. 使用随机种子，0.3的几率使出技能，0.7的几率使出普通攻击
  4. 如果双方为同类型的小精灵，使出的伤害低（0.8倍），否则伤害为1.3倍
  5. 如果使出defense\_attack的技能，为自己加血
  6. 在判断完是否使出技能后，设置暴击和闪避机制，0.2的暴击几率，对方有0.1的几率闪避
  7. 设置血条，可以实时看到自己和对方的血条（见实验结果）

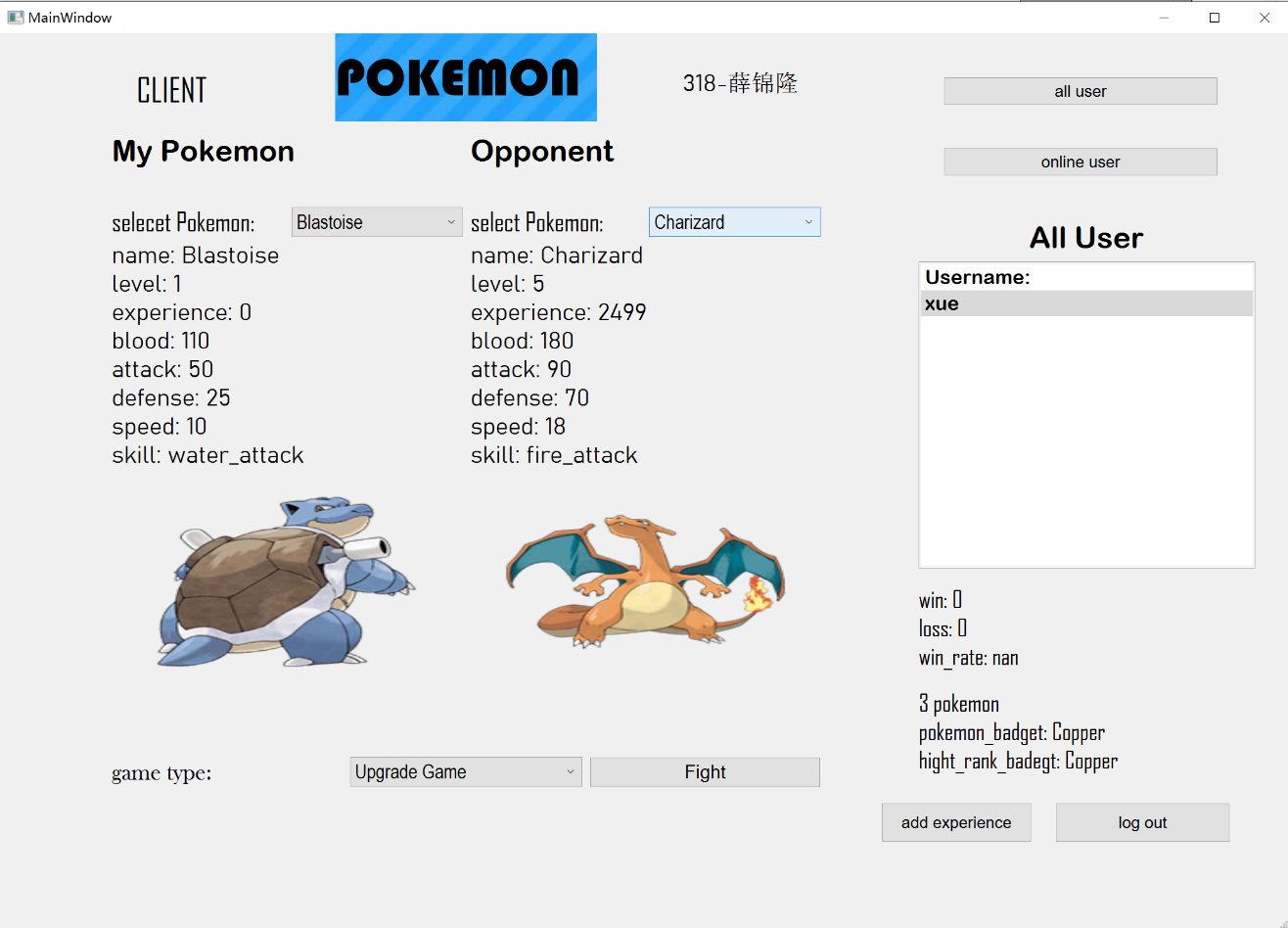
1. 实验结果
2. 服务器端
   * 1. 主界面
     2. 查看所有用户数据（点击all user）
     3. 查看所有小精灵数据（点击all pokemon）
3. 客户端
   1. 登录界面
   2. 注册成功
   3. 登陆成功
   4. 注册失败

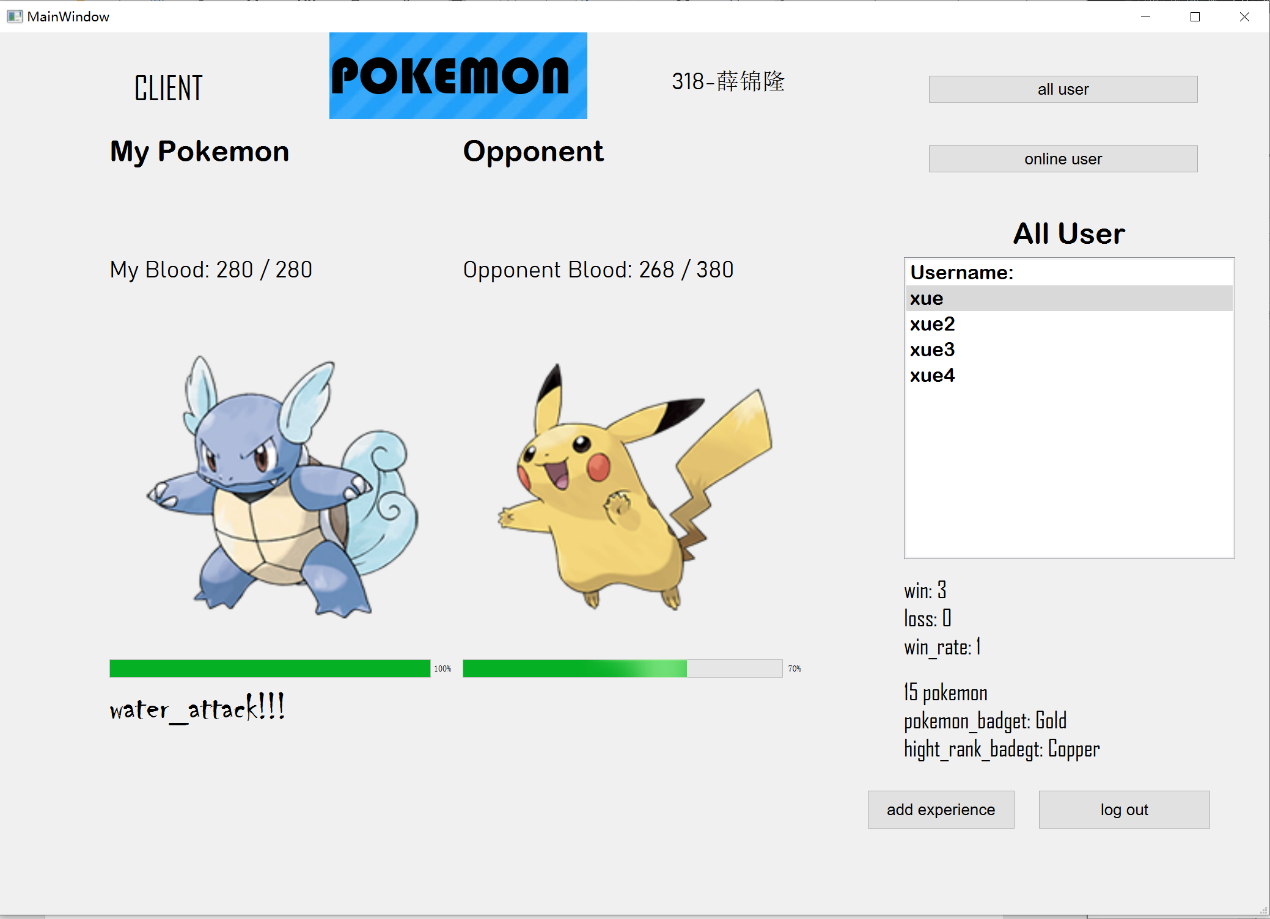


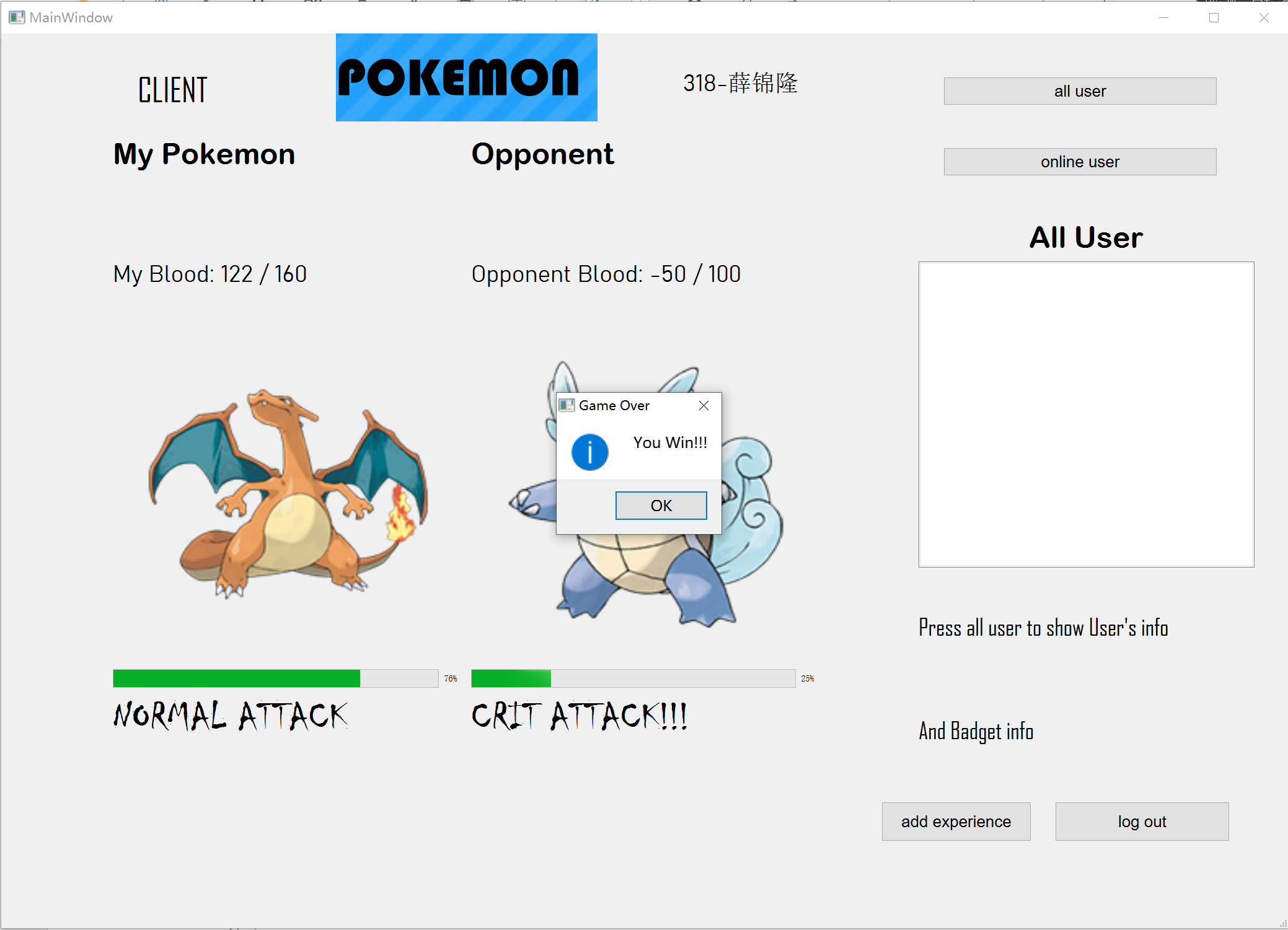
* 1. 密码错误



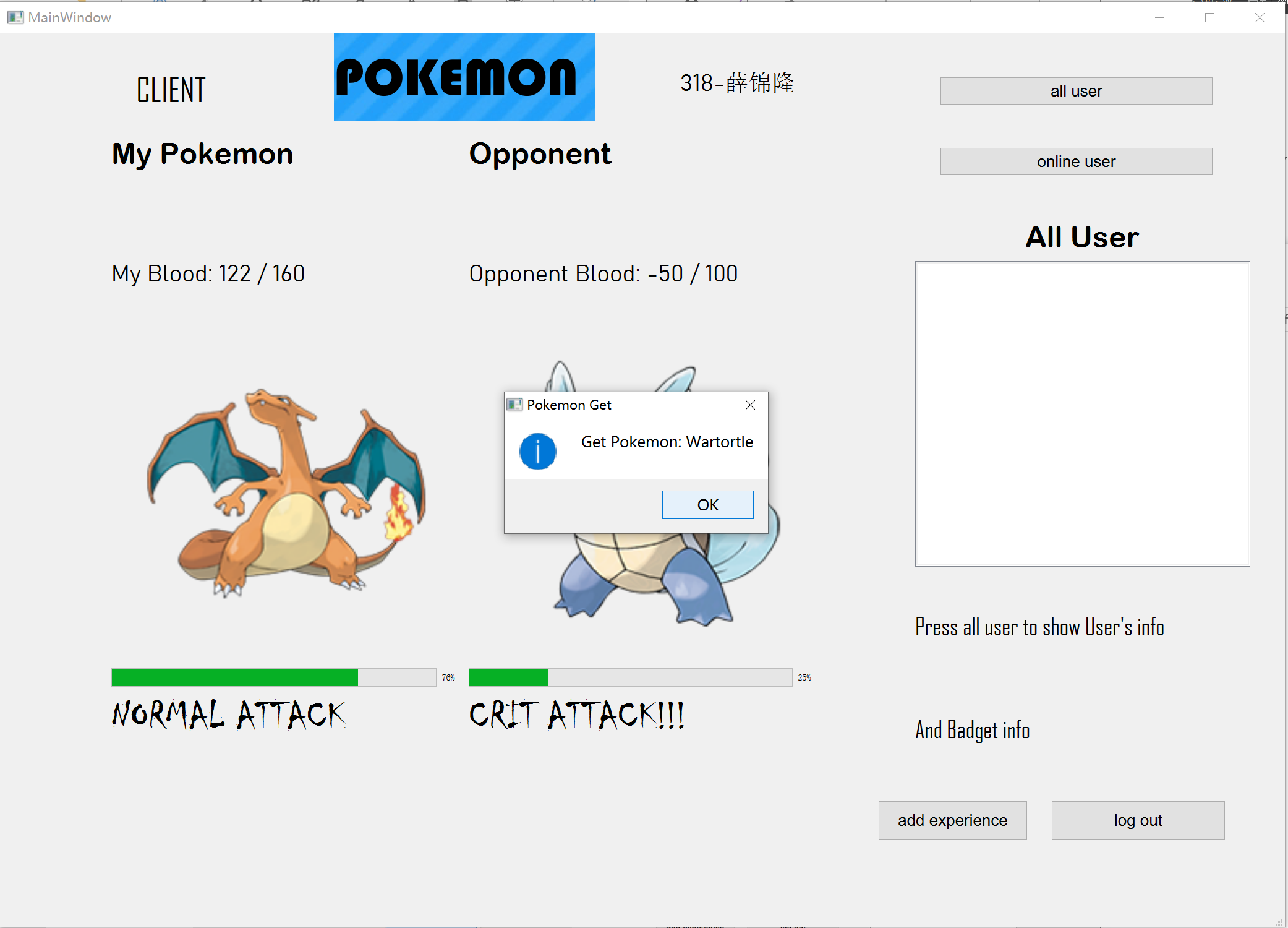
* 1. 主界面

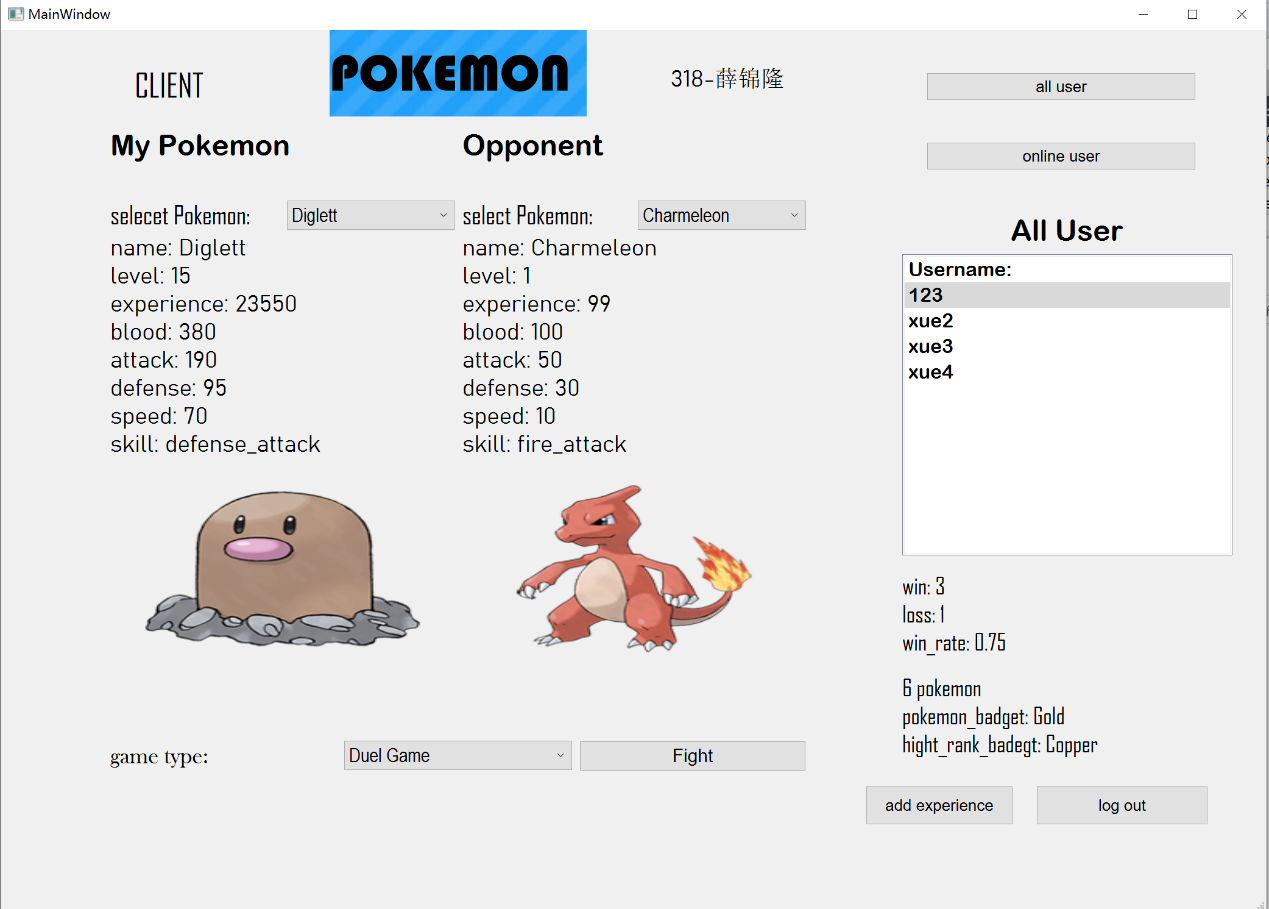


* 1. 战斗界面
  2. 在比赛中获胜

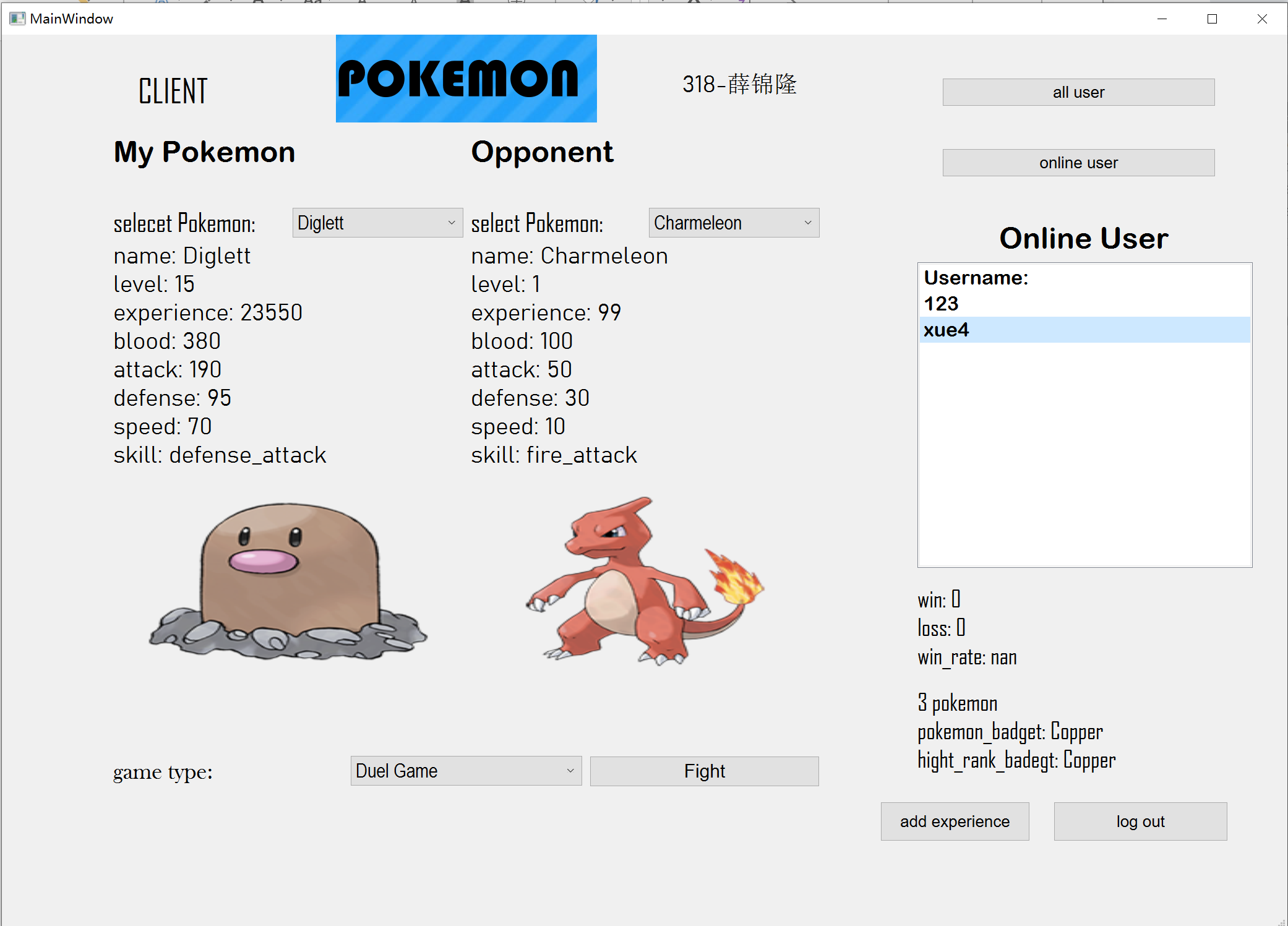


* 1. 升级赛中，pokemon升级
  2. 决斗赛中，赢得比赛，获得新的pokemon



* 1. 决斗赛中，输掉比赛，选择送出一个pokemon
  2. 查看徽章、胜负情况（分别点击all user、 online user）

注：显示所有用户的信息（all user）



注：显示在线用户的信息（online user）

1. 实验总结

在此次面向对象的小精灵对战系统编程中，我学会了使用C++的面向对象编程、数据库SQLITE3的使用（SQL语言）、QT界面编程、SOCKET编程，理解了客户端和服务器的数据交换的过程，并能够使用QT实现界面交互。同时，我加深了对C++类的继承、成员函数设计的理解。

最重要的是，在此次课程设计中，我的编程水平、数据结构和类设计得到了极大的提升。我从完全没有接触过QT，到理解其中的机制，并能设计创造出我想要实现的效果。从没有接触过面向对象编程，到可以合理使用类的设计，大大简化工作量，专注于对象的操作。

总之，我在本课程中收获很大，学到了面向对象编程的思想，理解了它的强大之处，并且，我的编程设计水平有很大提升。