C++程序设计实验报告-宠物小精灵对战系统

318-薛锦隆-2018210990

一、 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。

二、 运行环境

1. 系统：win10
2. 编程：5.14.2
3. 数据库：sqlite3

三、 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

四、 具体实现

1. 参数设计

```
1.     enum KIND   //pokemon 的所有种类
2. {
3.     HIGH_ATTACK,
4.     HIGH_BLOOD,
5.     HIGH_DEFENSE,
6.     HIGH_SPEED
7. };
```

```
enum SKILL  //pokemon 的特殊技能
{
 fire_attack,
 water_attack,
 defense_attack,
 double_attack,
 normal_attack   //所有精灵都有
}
```

```
const QList<QString> POKEMONNAME= {  //所有 pokemon 的名字（特殊技能；属性）
"Charmander",   //小火龙
 "Charmeleon",   //火恐龙
 "Charizard",    //喷火龙(fire_attack; high_attack)
 "Squirtle",     //杰尼龟
 "Wartortle",    //卡咪龟
 "Blastoise",    //水箭龟(water_attack; high_blood)
 "Diglett",      //地鼠
 "Dugtrio",      //三地鼠
 "Snorlax",      //卡比兽(defense_attack; high_defense)
 "Dodrio",       //嘟嘟利
 "Duduo",        //嘟嘟
 "Pikachu",      //皮卡丘(double_attack; high_speed)
}

enum LOG_TYPE  {SIGN_IN, SIGN_UP, NAME_EXIST, SIGN_UP_SUCCESS, USER_NONE, PWD_ER
ROR, SIGN_IN_SUCESS, SIGN_OUT, ALL_USER, ONLINE_USER,VIRTUAL_PKM, PKM_DATA}; //
不同的发送的种类
```

```
#define MAX_LEVEL 15  //最大等级
#define BASE 100      //基础值（不同种类的 pokemon 的初始化的基础值）
```

2. 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）

```
    class Pokemon : public QObject  //pokeomon 基类
{
    Q_OBJECT

protected:
    QString name;             //pokemon 名字
    unsigned int level;       //等级
```

```
    unsigned int experience;   //经验值
    unsigned int attack;       //攻击力
    unsigned int blood;        //血量
    unsigned int current_blood;  //当前血量
    unsigned int defense;        //防御力
    unsigned int speed;          //攻击速度
    unsigned int kind;           //类型
    SKILL skill;                 //特殊技能
    void setValue(unsigned int base_attack, unsigned int base_blood, unsigned int base_defense, unsigned int base_speed);

public:
    Pokemon(){};
    virtual unsigned int Attack(){return 0;}
     //每种精灵攻击方法不一样，为虚函数，需重写
    void experienceUp(unsigned int value);   //增加精灵经验值
    QString getSkill(){return ATTACKKIND[skill];}
    QString getAllValue();     //返回精灵所有信息
    virtual void levelUp(){};   //每种精灵升级的情况不一样，为虚函数，需重写
    void setLevel(unsigned int set_level);  //升级到指定等级
    Pkm* getAttr();
    virtual ~Pokemon() {}

signals:
public slots:
};
```

```
   class high_attack : public Pokemon  //子类：如：high_attack 属性
{
    Q_OBJECT
public:
    high_attack(SKILL skill);
    unsigned int Attack();  //重写攻击
    void levelUp();     //重写不同种类的精灵的升级方式
    ~high_attack(){};
};
```

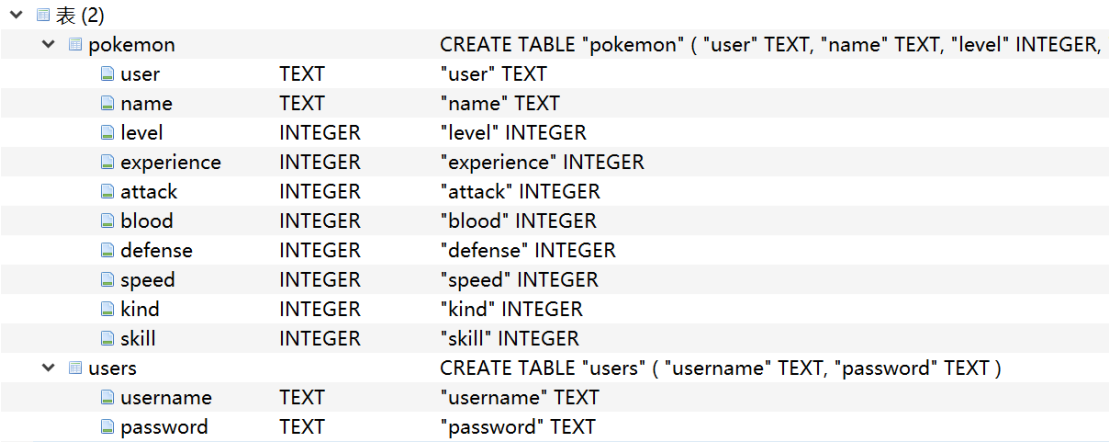
3. 数据库设计

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。



4. 用户类的设计（为节省空间，把不重要的函数省去）

```
1.    class User : public QObject  //用户类，定义了用户的各种操作
2. {
3.      Q_OBJECT
4. public:
5.     User();
6.     void setUser(QDataStream& dsIn);     //根据传来的数据载入到用户中
7.     void appendPkm(Pkm* pkm){allPkmAttr.append(pkm);};  //得到一个精灵
8.     void sendAllPkmAttr(QDataStream & dsOut);   //更新用户的所有信息
9.     Pkm* getPkmByIndex(unsigned int index){return allPkmAttr[index];};
10.    void setUser(Pkm** allPkm, unsigned int pkmNum);
11.      //从 login 界面中传递的数据，加载到用户中
12.    int attackOpponent(unsigned int index);
13.      //用户的一个精灵发出攻击，返回攻击伤害
14.    void addExperience(unsigned int index, unsigned int opLevel);
15.      //用户的 pokemon 增加经验值（根据对方精灵的等级）
16.    void levelUp(unsigned int index);  //用户的一个精灵升级
17.    void popPkmByIndex(unsigned int index){allPkmAttr.removeAt(index);};
18.      //失去一个 pokemon
19.    QString getUserBadget();    //返回用户徽章的情况
20.    unsigned int win;   //赢场
21.    unsigned int lose;  //败场
22.    ~User();
23.
24. private:
25.    QList<Pkm*> allPkmAttr; //用户的所有 pokemon 信息
26.    QString username;   //用户名
27.    unsigned int pkmNum;    //pokemon 数量
28. };
```

5. 存放所有小精灵的数据结构

```
    struct Pkm  //pokemon 信息，与 pokemon 表一致
{
    QString name;
    unsigned int level;
    unsigned int experience;
    unsigned int attack;
    unsigned int blood;
    unsigned int current_blood;
    unsigned int defense;
    unsigned int speed;
    unsigned int kind;
    unsigned int skill;
};
```

6. 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：

```
void createPokemon(QString username, unsigned int create_level);
//生成 pokemon，写入 pokemon 数据库
void sendPekemon(QDataStream& dsOut, QString user);
//发送 user 的所有 pokemon 的信息
void sendAllUser(unsigned int port);
//从数据库返回并发送所有用户信息（用户名，所有精灵）
void setSql(QString user, Pkm* pokemon);
//设置 pokemon 数据库的内容
void sendOnlineUser(unsigned int port);
//从 online_user 中返回数据，并发送
void appendUserFromSql(QString username);
//在线用户增加，新增一个 online_user（登录或注册+1，退出-1）
void updateUserPkm(QDataStream& dsIn, QString username);
//更新用户在数据库的所有 pokemon 信息
```

7. 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 userPkmInde
    x, User* opponent, unsigned int opPkmIndex); //开始比赛
11. bool fightProgress(User* user, unsigned int userPkmIndex, User* opponent, un
    signed 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
```

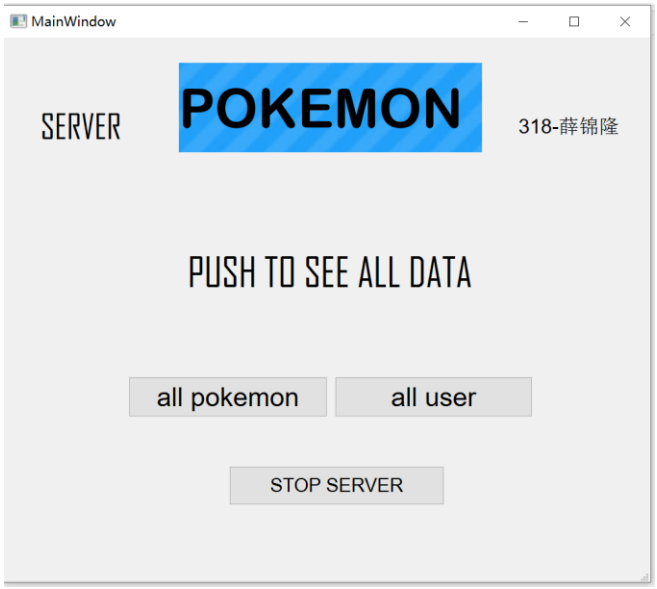
函数具体实现解释：

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）

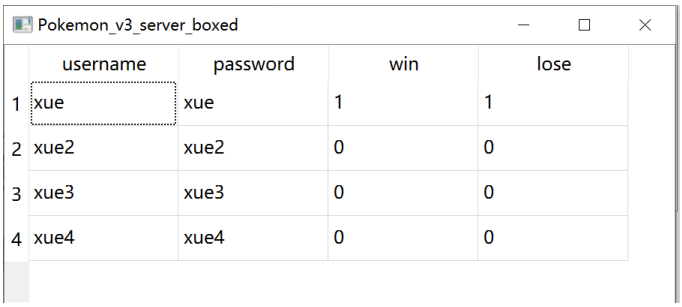
五、 实验结果

1. 服务器端

（1） 主界面

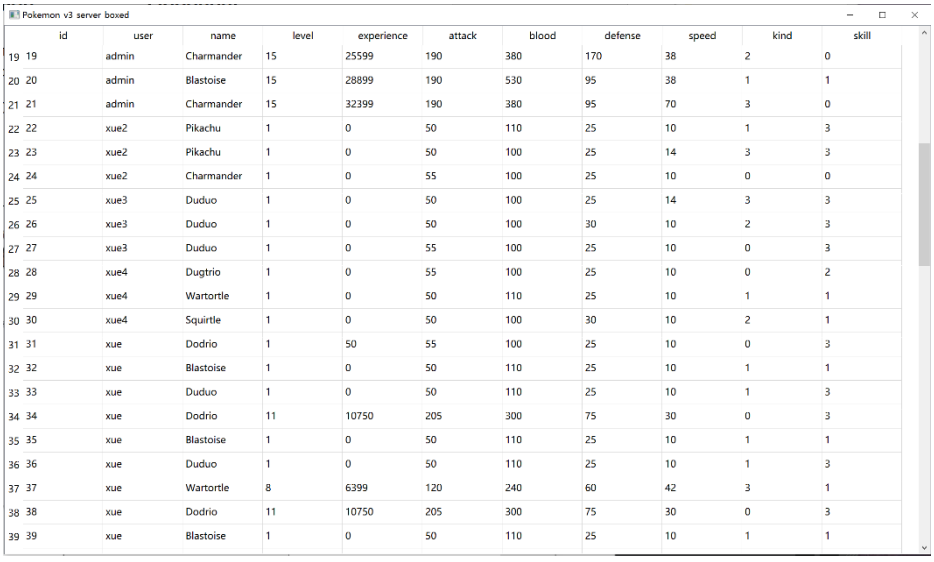


（2） 查看所有用户数据（点击 all user）



	username	password	win	lose
1	xue	xue	1	1
2	xue2	xue2	0	0
3	xue3	xue3	0	0
4	xue4	xue4	0	0

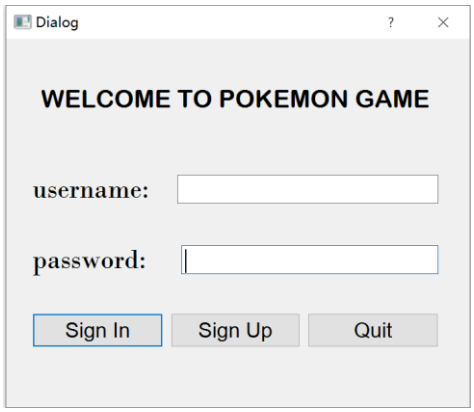
（3） 查看所有小精灵数据（点击 all pokemon）



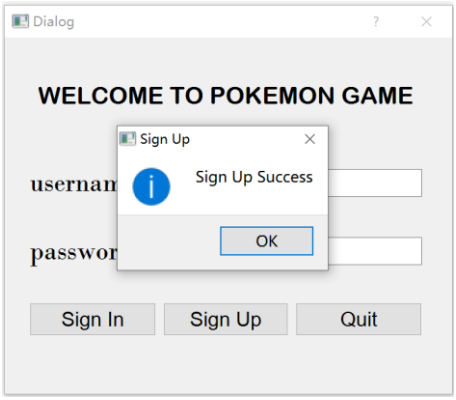
	id	user	name	level	experience	attack	blood	defense	speed	kind	skill
19	19	admin	Charmander	15	25599	190	380	170	38	2	0
20	20	admin	Blastoise	15	28899	190	530	95	38	1	1
21	21	admin	Charmander	15	32399	190	380	95	70	3	0
22	22	xue2	Pikachu	1	0	50	110	25	10	1	3
23	23	xue2	Pikachu	1	0	50	100	25	14	3	3
24	24	xue2	Charmander	1	0	55	100	25	10	0	0
25	25	xue3	Duduo	1	0	50	100	25	14	3	3
26	26	xue3	Duduo	1	0	50	100	30	10	2	3
27	27	xue3	Duduo	1	0	55	100	25	10	0	3
28	28	xue4	Dugtrio	1	0	55	100	25	10	0	2
29	29	xue4	Wartortle	1	0	50	110	25	10	1	1
30	30	xue4	Squirtle	1	0	50	100	30	10	2	1
31	31	xue	Dodrio	1	50	55	100	25	10	0	3
32	32	xue	Blastoise	1	0	50	110	25	10	1	1
33	33	xue	Duduo	1	0	50	110	25	10	1	3
34	34	xue	Dodrio	11	10750	205	300	75	30	0	3
35	35	xue	Blastoise	1	0	50	110	25	10	1	1
36	36	xue	Duduo	1	0	50	110	25	10	1	3
37	37	xue	Wartortle	8	6399	120	240	60	42	3	1
38	38	xue	Dodrio	11	10750	205	300	75	30	0	3
39	39	xue	Blastoise	1	0	50	110	25	10	1	1

2. 客户端

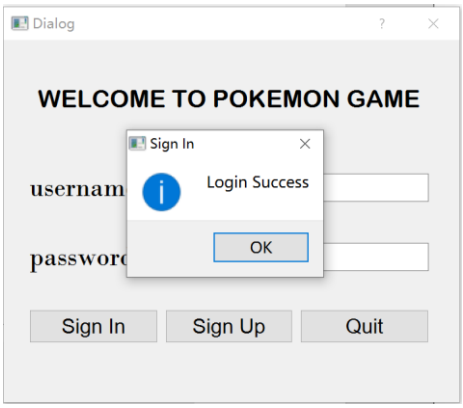
（1） 登录界面



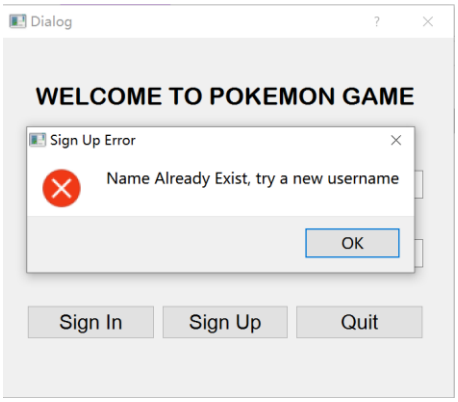
（2） 注册成功



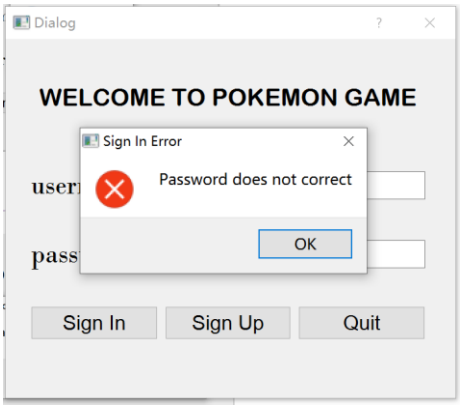
（3） 登陆成功



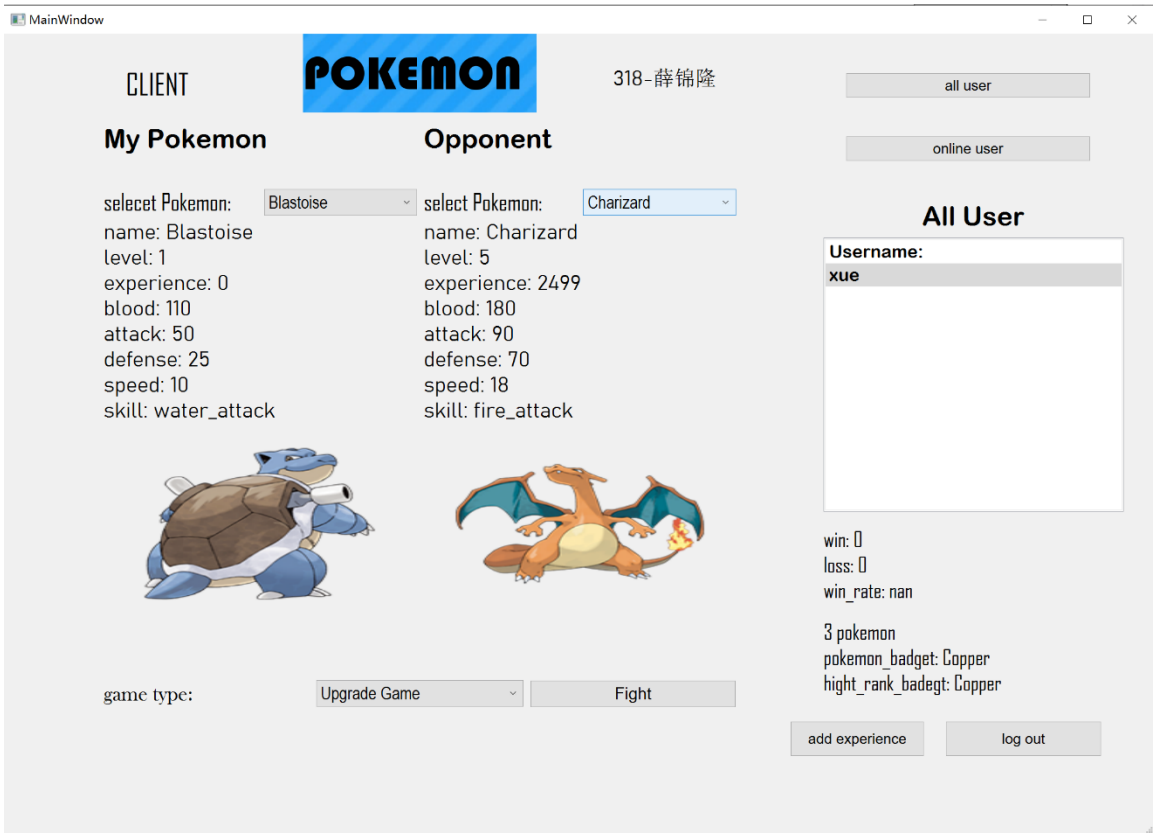
（4） 注册失败


Dialog
WELCOME TO POKEMON GAME
Sign Up Error
Name Already Exist, try a new username
OK
Sign In
Sign Up
Quit

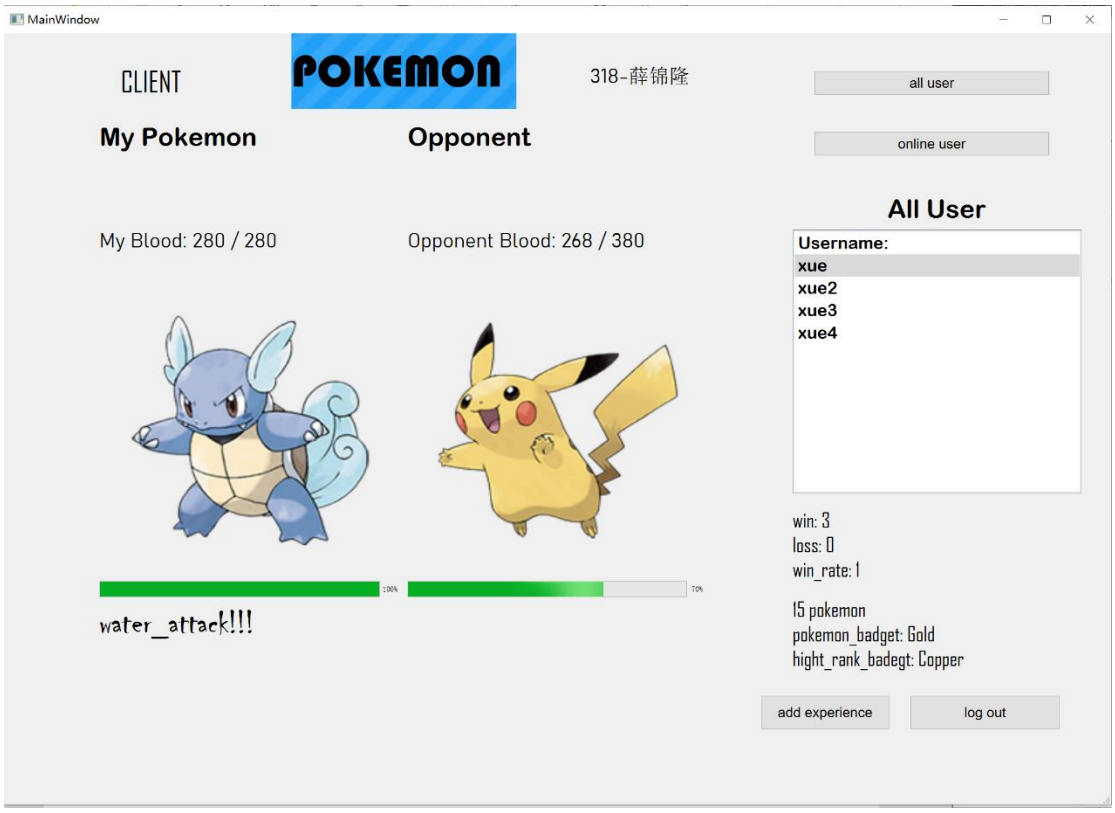
（5） 密码错误


Dialog
WELCOME TO POKEMON GAME
Sign In Error
Password does not correct
OK
Sign In
Sign Up
Quit

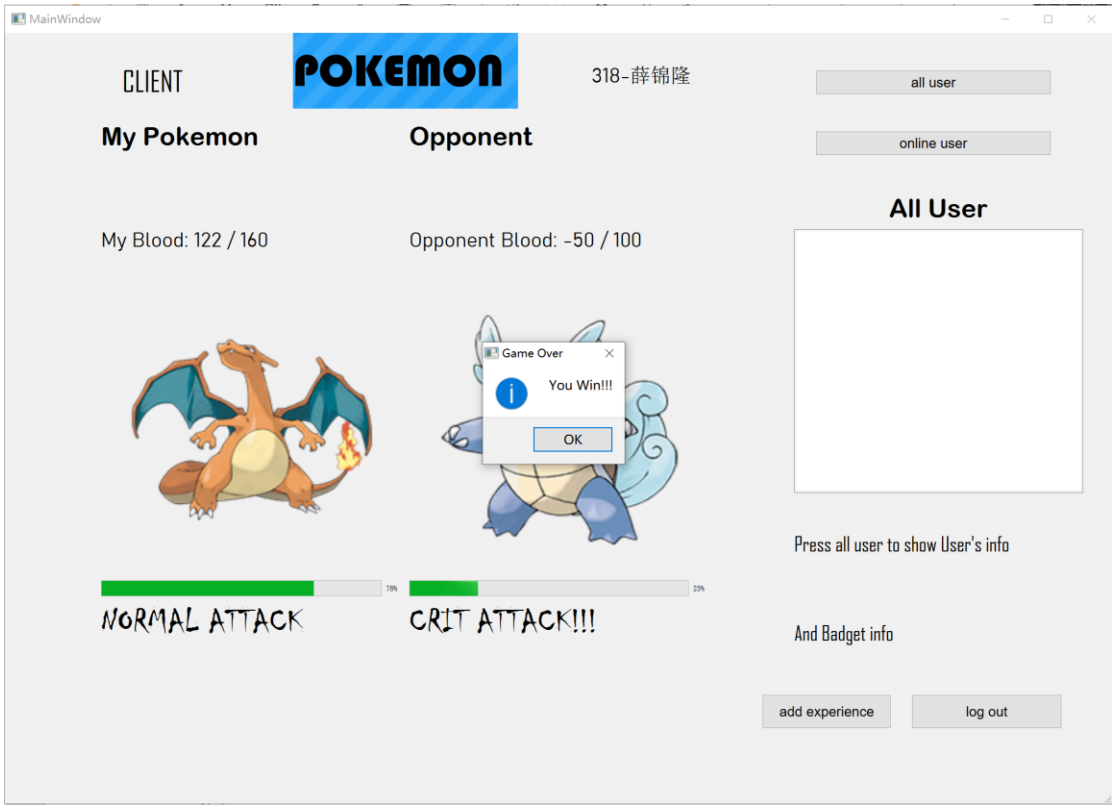
（6） 主界面


MainWindow
CLIENT
POKEMON
318-薛锦隆
all user
online user
My Pokemon
Opponent
selecet Pokemon:
Blastoise
name: Blastoise
level: 1
experience: 0
blood: 110
attack: 50
defense: 25
speed: 10
skill: water_attack
select Pokemon:
Charizard
name: Charizard
level: 5
experience: 2499
blood: 180
attack: 90
defense: 70
speed: 18
skill: fire_attack
All User
Username:
xue
win: 0
loss: 0
win_rate: nan
3 pokemon
pokemon_badget: Copper
hight_rank_badegt: Copper
game type:
Upgrade Game
Fight
add experience
log out

（7） 战斗界面

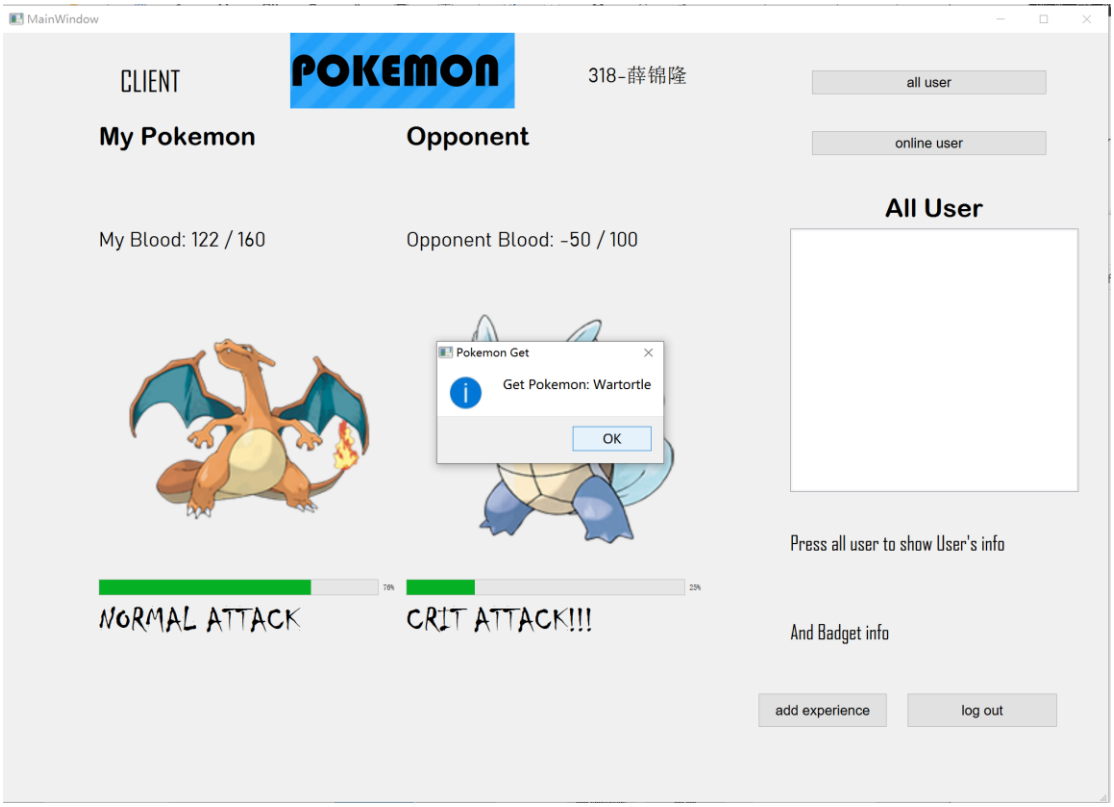


（8） 在比赛中获胜



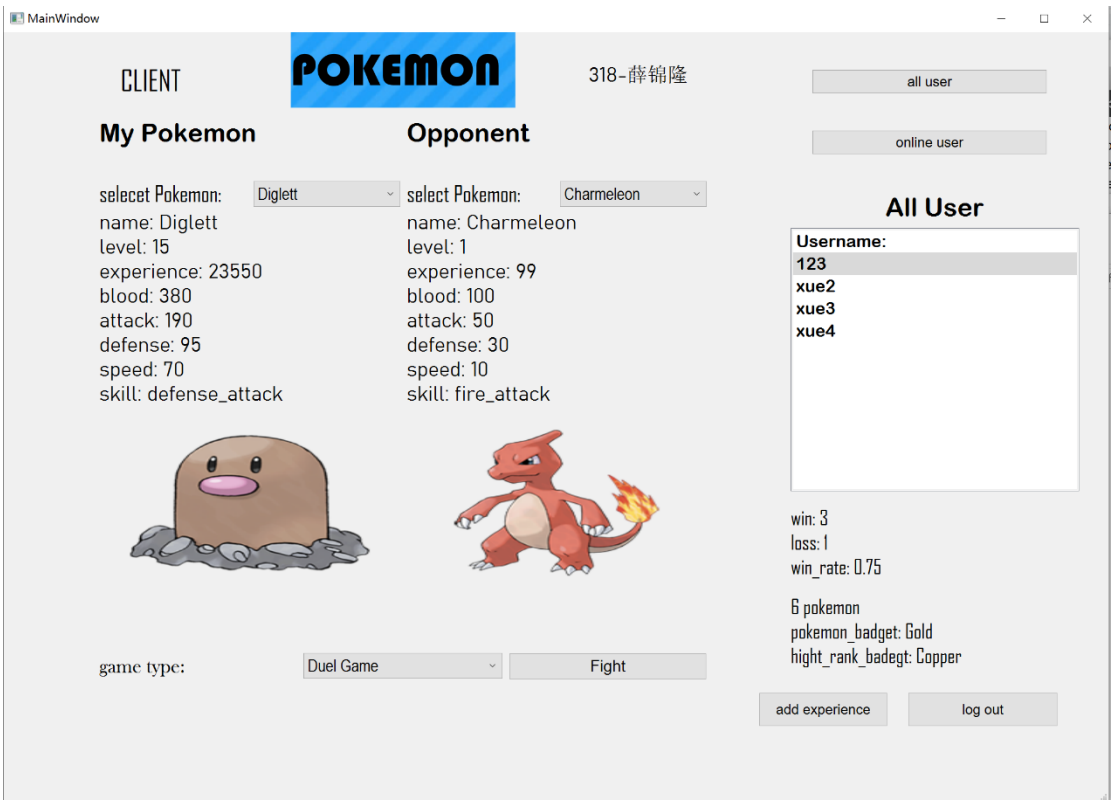
（9） 升级赛中，pokemon 升级

（10） 决斗赛中，赢得比赛，获得新的 pokemon

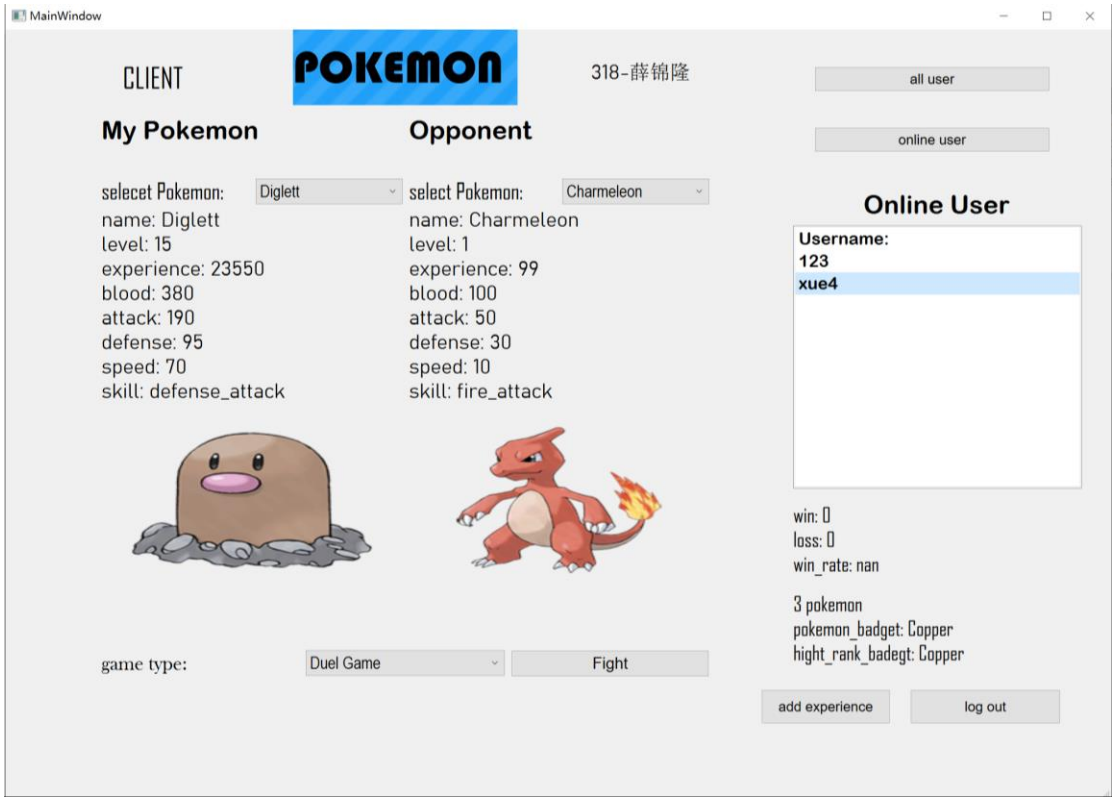


（11） 决斗赛中，输掉比赛，选择送出一个 pokemon

（12） 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）

六、 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。