C++ 程序设计—宠物小精灵实验报告

学号-2014211281-黄智伟

2016 年 12 月 27 日

目录

1 设计任务的描述

应用面向对象的设计方法来设计一款平台类对战游戏。

2 运行环境

1. 系统：kali
2. 编程环境：qt5.7
3. 数据库：mysql

3 功能需求说明

3.1 第一阶段

实现小精灵类的设计，包括小精灵种类，各种属性及攻击方式设计，同时编写测试函数进行是测试。

3.2 第二阶段

1. 每个用户需要注册一个账号，用户名全局唯一，不能有任何两个用户名相同，要考虑注册失败的场景时的反馈。
2. 实现注册、登录、登出功能，均采用 C/S 模式，客户端和服务端用 socket 进行通信，服务端保存所有用户的信息。
3. 每个用户拥有：用户名、拥有的精灵，两个属性。用户注册成功时，系统自动随机分发三个 1 级精灵给用户。
4. 用户可以查看所有成功注册用户拥有的精灵，也可以查看所有当前在线的用户。

3.3 第三阶段

1. 实现用户小精灵与系统进行升级赛，升级可获得经验，进而升级。
2. 实现用户小精灵与系统进行决斗赛，战胜可获得此精灵，战败则系统从用户的精灵中随机选三个（不够三个精灵的情况就选择他所有的精灵），然后由用户选一个送出。
3. 用户如果没有精灵（比如总是失败，已经全部送出去），则系统会随机放给给他一个初级精灵。
4. 实现用户可查看某一用户胜率的功能。
5. 用户增加新属性：宠物个数徽章（金银铜）和高级宠物徽章（金银铜），分别根据拥有的宠物个数的多少和拥有高级宠物（15 级）个数的多少颁发。

4 具体实现

4.1 第一阶段

4.1.1 基类设计

主要数据类型:

```
enum KIND//四种类型
{
    HIGH_ATTACK,HIGH_BLOOD,HIGH_DEFENSE,HIGH_SPEED
};
const QList<QString> POKEMONNAME={
    "Feraligatr",//大力鳄
    "Charizard",//喷火龙
    "Noivern",//音波龙
    "Bagon",//宝贝龙
    "Glaceon",//冰伊布
    "Pikachu",//皮卡丘
    "Wynaut",//小果然
    "Doduo",//嘟嘟
};//pokemon名字

enum SKILL//技能
{
    HydroCannon,//加农水炮
    Cut,//居合斩
    LeechLife,//吸血
    FocusEnergy,//聚气
    Barrier,//屏障
    LightScreen,//光墙
    Counter,//双倍奉还
    TriAttack,//三重攻击
    NormalAttack,//普通攻击
};

enum QUALIFICATION{
    S,SS,SSS
};//属性等级

class Pokemon : public QObject
{
    Q_OBJECT
```

```
public:
    Pokemon():level(1),experience(0){}
    virtual ~Pokemon(){}

    virtual uint Attack(){return 0;}//精灵技能
    void ExperienceUp(uint value);
    virtual void LevelUP(){}

    QString getName();
    uint getLevel();
    QString getInformation();
protected:
    QString name;       //名字
    uint level;         //等级
    uint experience;    //经验
    uint attack;        //攻击属性
    uint blood;         //生命属性
    uint currentBlood;//实时血量，对决时使用
    uint defense;       //防御属性
    uint speed;         //敏捷属性
    uint kind;          //种类
    uint attr;          //稀有度
    SKILL skill;         //技能

    void setValues(uint baseAttack, uint baseBlood, uint baseDefense,
                   uint baseSpeed);

signals:

public slots:
};
```

主要函数:

```
void Pokemon::ExperienceUp(uint value)//增加经验函数

QString Pokemon::getName()//小精灵名字的get函数

uint Pokemon::getLevel()//小精灵等级的get函数

QString Pokemon::getInformation()//小精灵各种属性的get函数
```

```
void Pokemon::setValues(uint baseAttack, uint baseBlood, uint baseDefense,
                        uint baseSpeed)//小精灵各种属性的set函数
```

4.1.2 子类实现

以 HighAttack 子类为例：

```
uint HighAttack::Attack()//攻击函数
{
    qsrand (QTime::currentTime ().msec ());
    uint probability=qrand ()%10;
    if(probability<8)//普通攻击
        return NormalAttack;
    else              //20%几率使用技能
        return this->skill;
}

void HighAttack::LevelUP()//升级函数
{
    this->level++;

    this->attack+=100;
    this->blood+=50;
    this->currentBlood=this->blood;//升级后当前的血量回复满血
    this->defense+=50;
    this->speed+=50;
}
```

4.2 第二阶段

4.2.1 数据结构

1. UDP 数据包类型：

```
const uint SIGNUP=1;
const uint SIGNIN=2;
const uint NAMEEXIST=3;
const uint SIGNUPOK=4;
const uint NOSUCHUSER=5;
const uint PWDDIFF=6;
const uint SIGNINOK=7;
const uint PKMDATA=8;
const uint SIGNOUT=9;
const uint ONLINEUSERS=10;
```

```
const uint ALLUSERS=11;
```

2. mysql 中的 users 表（第二阶段只使用列 username 和 password，列 win 和 lose 为第三阶段使用）：

Field	Type	Null	Key	Default	Extra
username	char(32)	NO		NULL	
password	char(128)	NO		NULL	
win	int(10) unsigned	YES		0	
lose	int(10) unsigned	YES		0	

3. mysql 中的 pokemon 表：

Field	Type	Null	Key	Default	Extra
user	char(32)	NO		NULL	
name	char(32)	NO		NULL	
level	int(10) unsigned	NO		NULL	
experience	int(10) unsigned	NO		NULL	
attack	int(10) unsigned	NO		NULL	
blood	int(10) unsigned	NO		NULL	
defense	int(10) unsigned	NO		NULL	
speed	int(10) unsigned	NO		NULL	
kind	int(10) unsigned	NO		NULL	
attr	int(10) unsigned	NO		NULL	
skill	int(10) unsigned	NO		NULL	

4. 在线用户类（用于发送在线用户数据时的序列化）：

```
struct User//在线用户
{
    QString username;//用户名
    uint port;//端口号

    bool operator==(const User &user) const
    {
        return this->username==user.username&&this->port==user.port;
    }

    //友元声明，一对用于支持QDataStream 输入输出的函数
    friend QDataStream &operator<<(QDataStream &stream, const User &user);
    friend QDataStream &operator>>(QDataStream &stream, User &user);
};
```

5. 所有用户数据类（用于发送所有用户数据时的序列化）：

```
struct UserData
{
    QString username;//用户名
    QList<UDPPkm*> pkm;//小精灵

    ~UserData()
    {
        for(int i=0;i<3;i++)
            delete this->pkm[i];
    }

    //友元声明，一对用于支持QDataStream 输入输出的函数
    friend QDataStream &operator<<(QDataStream &stream,
                                   const UserData &userData);
    friend QDataStream &operator>>(QDataStream &stream, UserData &userData);
};
```

6. pokemon 数据包类（小精灵的数据）：

```
struct UDPPkm
{
    QString name;       //名字
    uint level;         //等级
    uint experience;    //经验
    uint attack;        //攻击属性
    uint blood;         //生命属性
    uint currentBlood;//实时血量，对决时使用
    uint defense;       //防御属性
    uint speed;         //敏捷属性
    uint kind;          //种类
    uint attr;          //稀有度
    uint skill;         //技能

    //友元声明，一对用于支持QDataStream 输入输出的函数
    friend QDataStream &operator<<(QDataStream &stream, const UDPPkm &pkm);
    friend QDataStream &operator>>(QDataStream &stream, UDPPkm &pkm);
};
```

4.2.2 服务器端实现

服务器监听 6665 端口，一有数据到达就调用 readPendingDatagrams() 函数分析客户端请求的种类并调用相应的函数处理。在线用户由服务器端用一个列表实现，用户注册及登录时将该用户加入列表中，登出从列表中移除。

1. 系统随机生成三个小精灵给用户
   void CreatePkm(QString user);//随机分配小精灵

2. 将生成的小精灵存入数据库
   void PutIntoSql(QString user, UDPPkm *pkm);

3. 发送小精灵数据
   void SentPkm(QDataStream &dsOut, QString user);//发送小精灵的数据

4. 发送在线用户给客户端
   void SentOnlineUsers(uint port);//发送在线用户

5. 发送所有用户数据给客户端
   void SentAllUsers(uint port);//发送注册用户

6. 分析客户端请求并处理

```
void MainWindow::readPendingDatagrams()
{
    ...
    if(loginKind==SIGNUP)
    {
        判断用户是否已存在，并进行处理，返回结果给请求的客户端
    }
    else if(loginKind==SIGNIN)
    {
        验证该用户是否存在及密码正确与否，进行处理，并返回结果给请求的客户端
    }
    else if(loginKind==SIGNOUT)
    {
        用户登出，清楚服务器端的在线用户列表
    }
    else if(loginKind==ONLINEUSERS)
    {
        发送在线用户给客户端
    }
    else if(loginKind==ALLUSERS)
    {
        发送所有用户给客户端
    }
}
```

4.2.3 客户端实现

客户端提供与用户交互的界面，并接受用户的输入，显示服务器返回的数据。客户端分为两个界面，一个为登录注册界面，另一个为主界面，显示小精灵数据，并提供用户查询在线用户及所有用户的操作界面。启动客户端时检查默认端口（6666）是否已被占用，若是则端口号加一。

1. 登出
   void on_BtnLoginOut_clicked();
2. 查询在线用户（发送 UDP 数据包查询）
   void on_BtnOnline_clicked();
3. 查询所有用户（发送 UDP 数据包查询）
   void on_BtnAllUsers_clicked();
4. 接收数据并进行相应处理
   void readPendingDatagrams();

4.3 第三阶段[1]

4.3.1 数据结构（详见第二阶段4.2.1）

第三阶段增加的 UDP 数据包种类为：

```
const uint GETADMINPKM=12;//获取系统小精灵
const uint UPGRADE=13;//升级赛
const uint DUEL=14;//决斗赛
const uint UPDATED=15;//更新后小精灵
const uint WINNINGRATE=16;//获胜率
```

4.3.2 服务器端

相较于第二阶段增加了更新数据库的操作，同时服务器端随机生成十个小精灵（3 个一级，3 个中级，2 个高级，2 个满级）保存在一个列表中，供用户升级赛及决斗赛挑战。

1. 若用户的小精灵经验增加到足够升级，则进行升级同时更新数据库
   void UpdatePkm(QString username, uint index, uint adminIndex);//更新用户升级的小精灵
2. 用户决斗赛战胜，更新数据库
   void AddPkm(QString username, uint index);//用户增加小精灵，更新数据库
3. 用户决斗赛战败，送出小精灵，更新数据库
   void DeletePkm(QString username, uint index);//用户删除小精灵，更新数据库
4. 判断用户是否还有小精灵，若无则分配
   void UpdateUserPkm(QString username);//判断用户是否没有小精灵并分配

[1]由于第三阶段是在第二阶段的基础上编写的，故第二阶段存在的数据结构及函数就不再说明

5. 读取客户端请求并进行处理（只说明第三阶段增加的部分）

```
void MainWindow::readPendingDatagrams()
{
    ...
    else if(loginKind==GETADMINPKM)
    {
        随机生成十个小精灵并发送给客户端
    }
    else if(loginKind==UPGRADE)
    {
        接受用户端升级赛的战斗结果并相应地更新数据库
    }
    else if(loginKind==DUEL)
    {
        接受客户端决斗赛的结果并相应地更新数据库
    }
    else if(loginKind==WINNINGRATE)
    {
        发送用户端的战胜及失败场数（由此可计算出用户的胜率）
    }
}
```

4.3.3 客户端实现

新增了两个类 Fight（实现战斗的类）和 SelectPkm（实现用户选择一个小精灵送出）。战斗部分由客户端实现，客户端先获取十个系统小精灵的数据，并选择一个进行升级赛或决斗赛。战斗过程为半即时，由小精灵的 speed 属性决定攻击间隙。小精灵进行攻击时有 20% 几率使用技能，70% 几率普通攻击，同时还有 10% 几率被对手闪避。

1. 获取系统小精灵（小精灵每次请求时重新随机生成）
   void on_BtnUpgrade_clicked();

2. 获取系统小精灵（小精灵每次请求时重新随机生成）
   void on_BtnDuel_clicked();

3. 调用 Fight 类进行战斗
   void on_BtnConfirm_clicked();

4. 用户小精灵攻击（20% 几率使用技能，70% 几率普通攻击，同时还有 10% 几率被对手闪避）
   void selfAttack();//用户攻击

5. 系统小精灵攻击（20% 几率使用技能，70% 几率普通攻击，同时还有 10% 几率被对手闪避）
   void oppoAttack();//对手攻击

6. 传递用户选择的小精灵给主界面（此函数在 SelectPkm 类中）
   void sentSelectResult(uint index);//用户选择结果

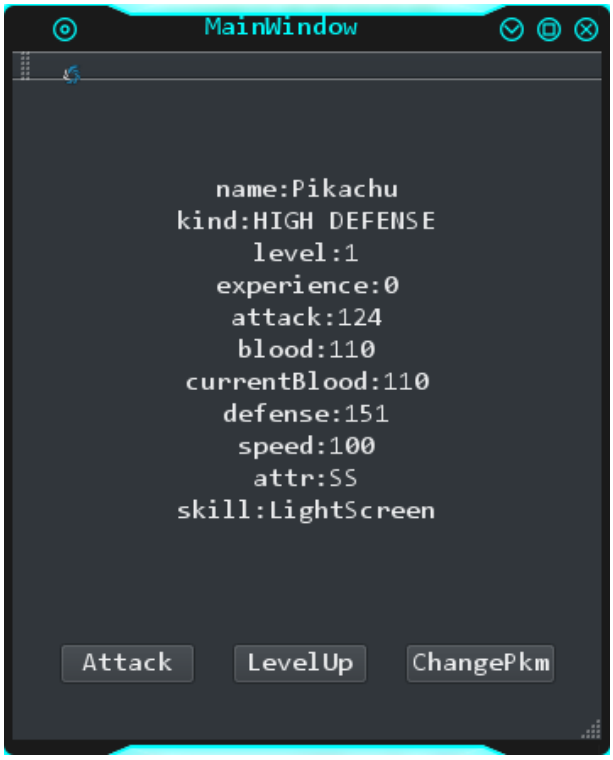
7. 从 Fight 类获得战斗结果（槽函数）
   void recvResult(bool isWin);//接受战斗结果

8. 获取用户选择结果并发送给服务器，更新数据库
   void recvSelectResult(uint index);//接受用户选择的小精灵

5 使用说明

5.1 第一阶段

主界面：



普通攻击：

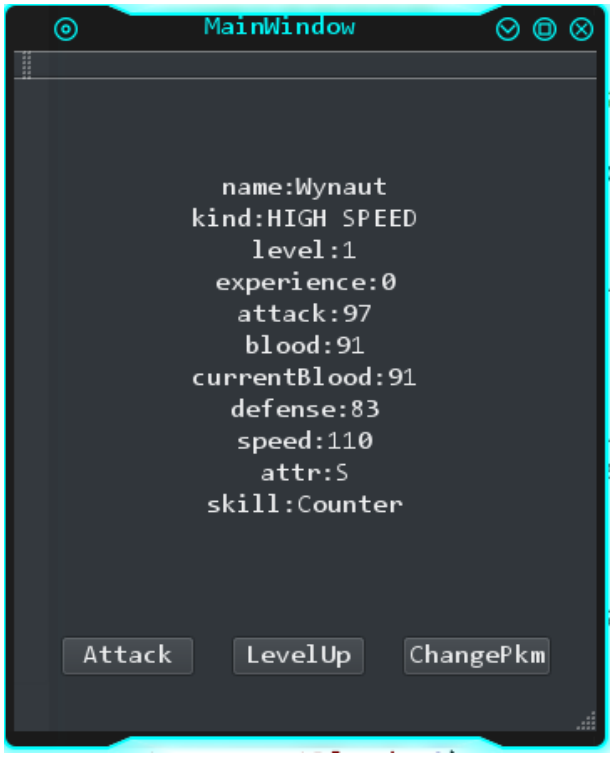

MainWindow
name:Pikachu
kind:HIGH DEFENSE
level:1
experience:0
attack:124
blood:110
currentBlood:110
defense:151
speed:100
attr:SS
skill:LightScreen
Attack
LevelUp
ChangePkm
Pikachu using Attack

使用技能：


MainWindow
name:Pikachu
kind:HIGH DEFENSE
level:1
experience:0
attack:124
blood:110
currentBlood:110
defense:151
speed:100
attr:SS
skill:LightScreen
Attack
LevelUp
ChangePkm
Pikachu using LightScreen

升级（升到满级 15 级后，Level Up 按钮将被禁用）：

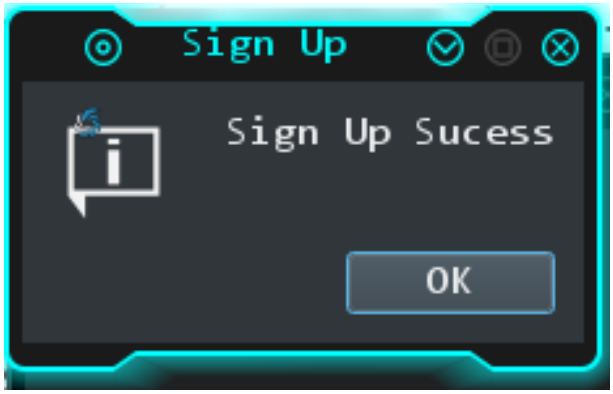
更换小精灵：



5.2 第二阶段

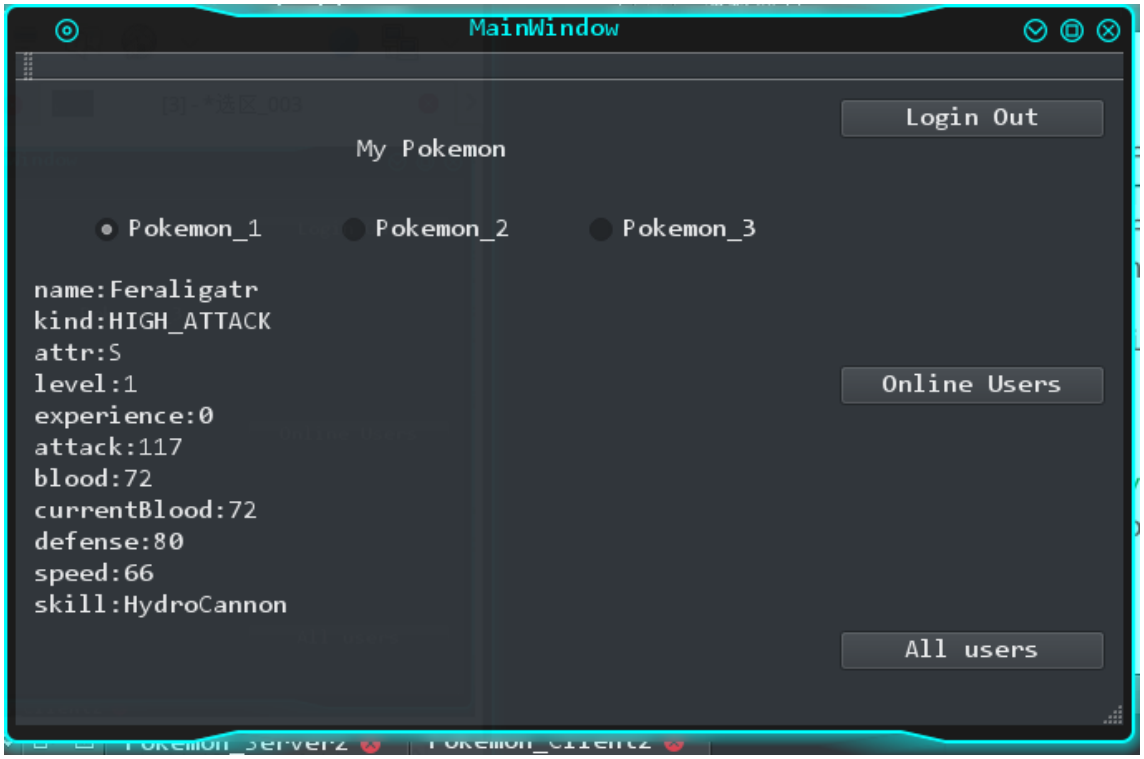
登录界面：

注册：

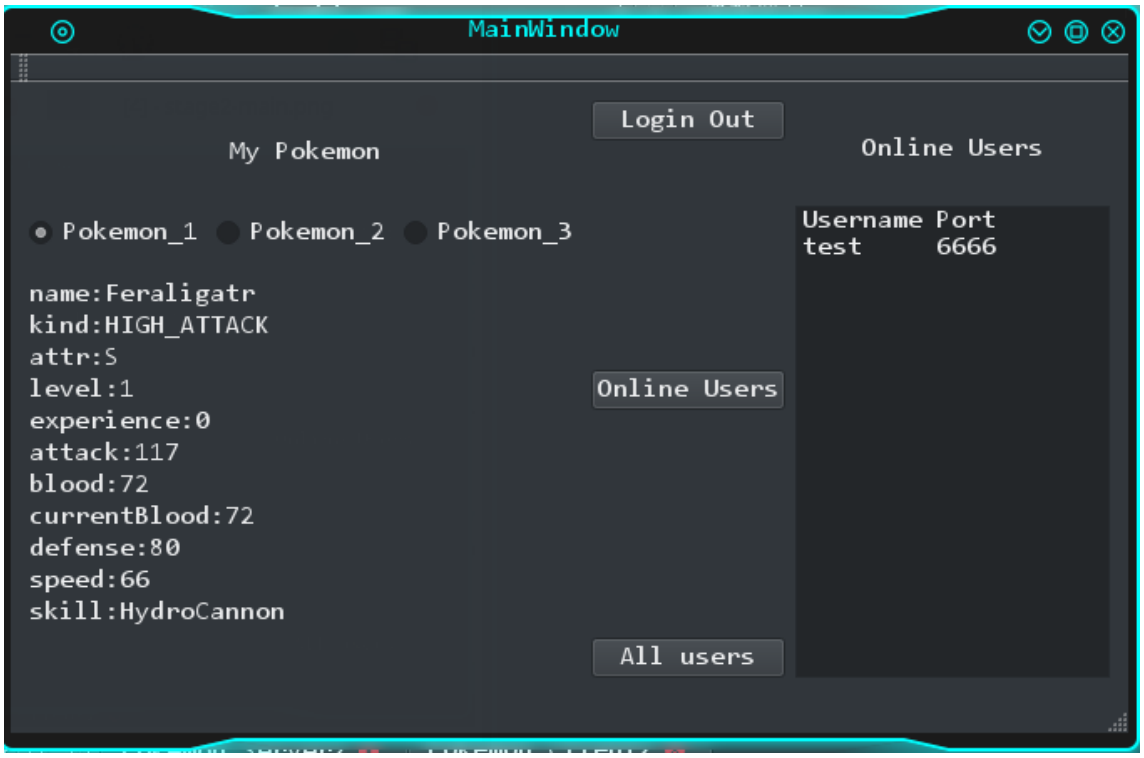


登录：

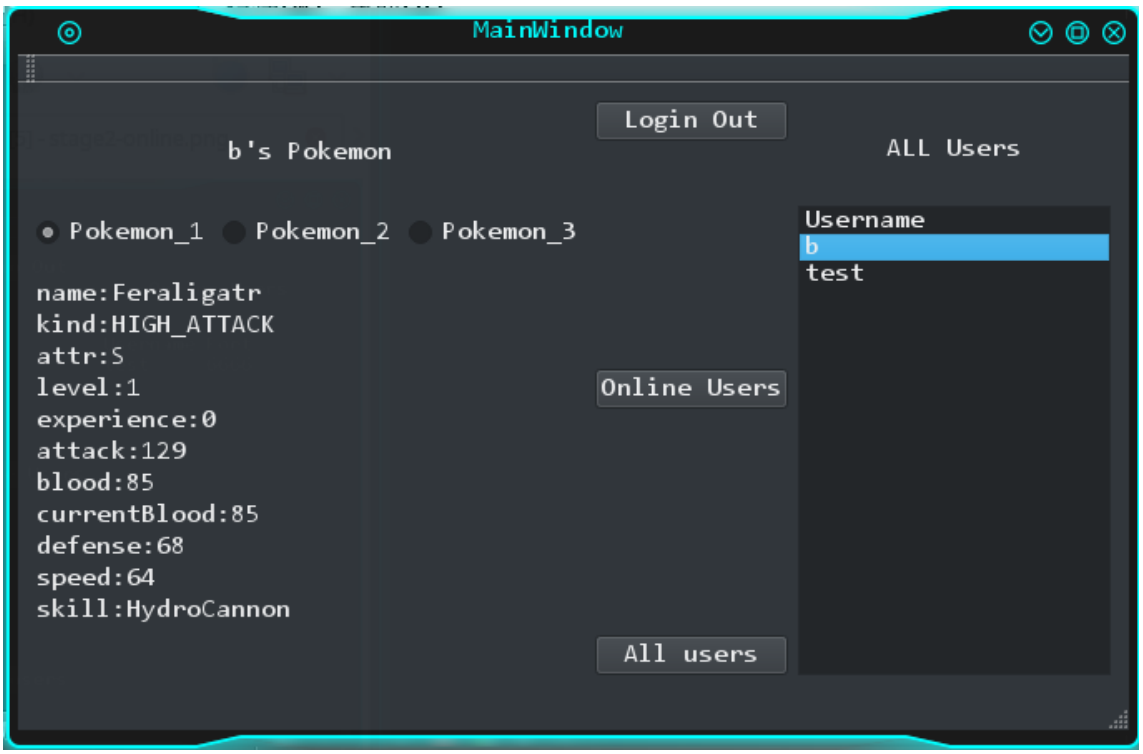
主界面：


MainWindow
Login Out
My Pokemon
Pokemon_1
Pokemon_2
Pokemon_3
name:Feraligatr
kind:HIGH_ATTACK
attr:S
level:1
experience:0
attack:117
blood:72
currentBlood:72
defense:80
speed:66
skill:HydroCannon
Online Users
All users

查询在线用户：

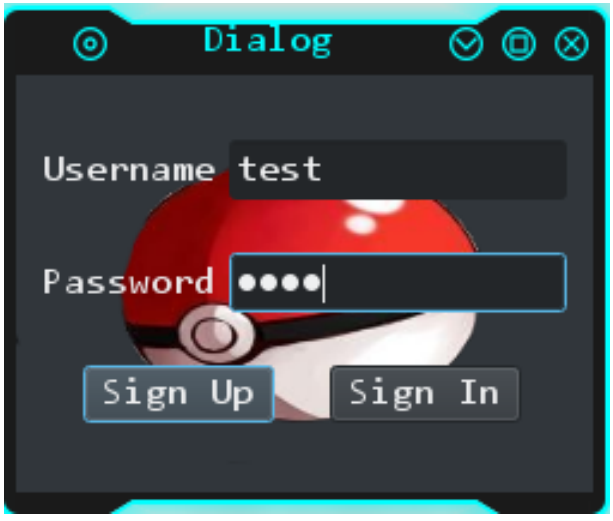

MainWindow
Login Out
My Pokemon
Online Users
Pokemon_1
Pokemon_2
Pokemon_3
Username Port
test 6666
name:Feraligatr
kind:HIGH_ATTACK
attr:S
level:1
experience:0
attack:117
blood:72
currentBlood:72
defense:80
speed:66
skill:HydroCannon
Online Users
All users

查询所有用户：

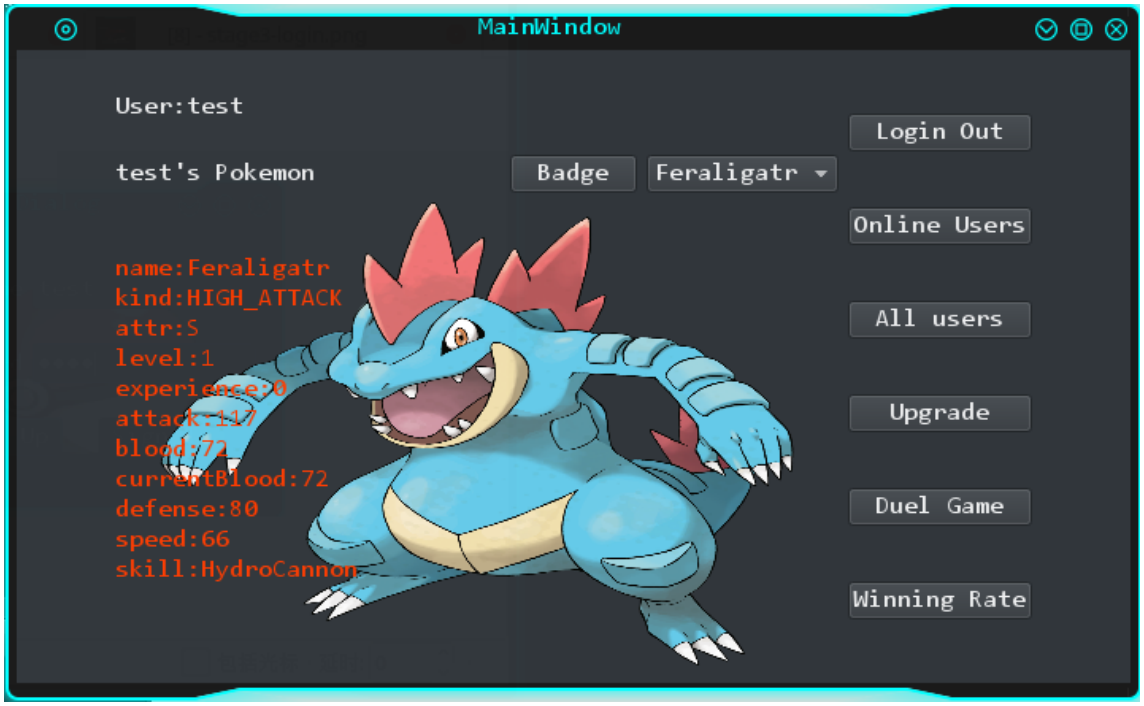


5.3 第三阶段

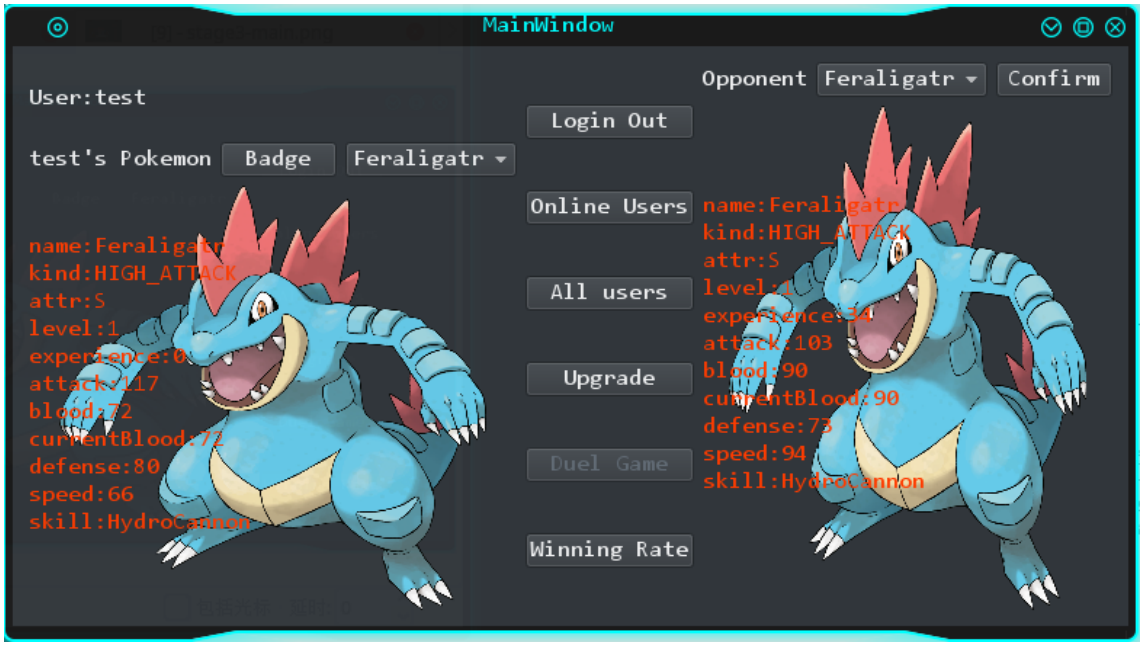
登录界面（第二阶段已截取注册界面，故不再截取）：



主界面：



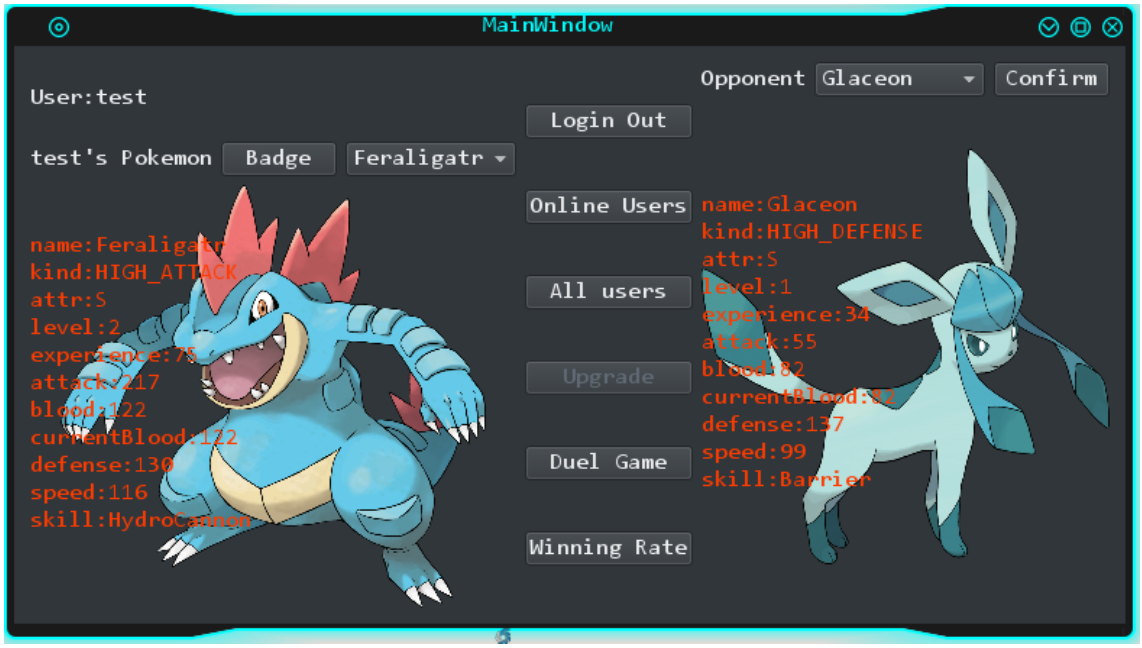
升级赛选择小精灵（决斗赛与此类似，故不再截图）：



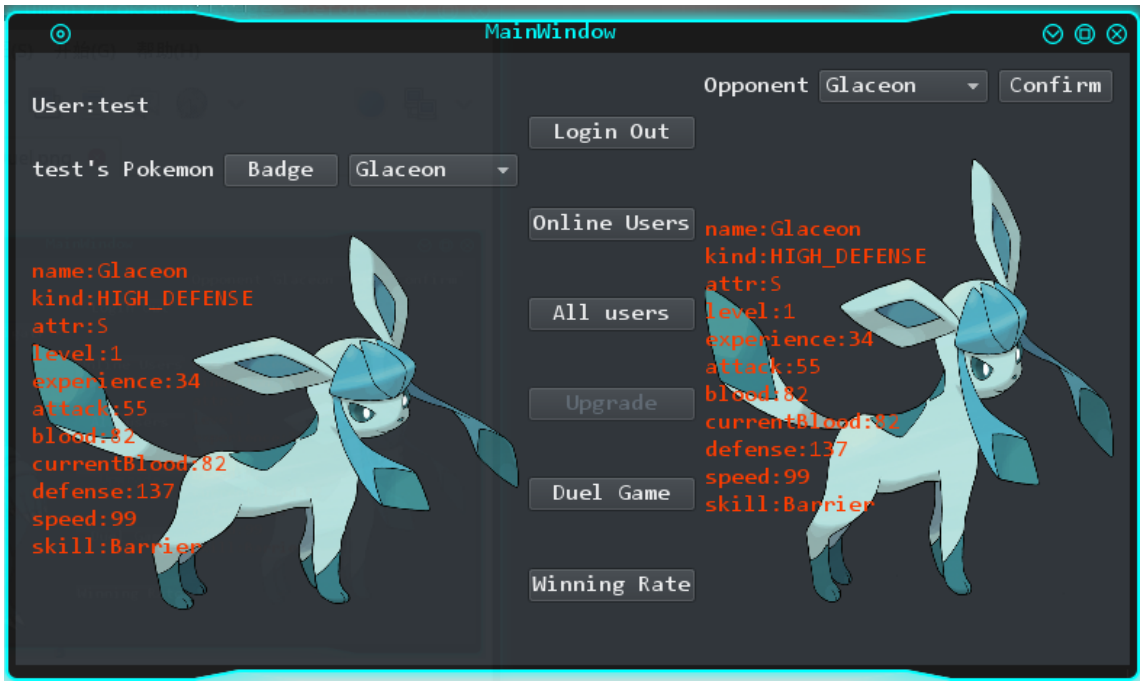
战斗（左边小精灵使用普通攻击，右边小精灵的攻击则被闪避）：

胜利（从下图可看出小精灵战胜后经验增加，且已升级）：

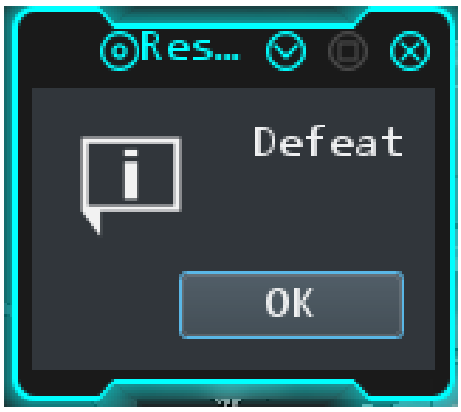
决斗赛前（战胜后可获得右边小精灵）：



决斗赛后（已战胜，且获得右边小精灵）：

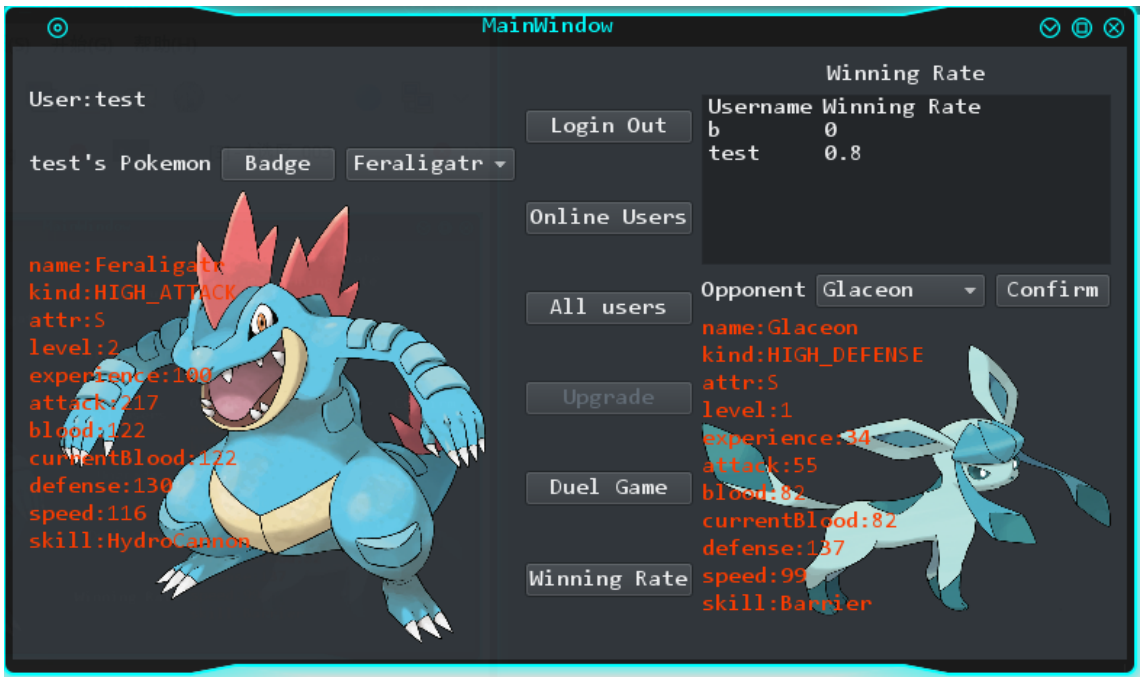


战败：



战败选择一个小精灵送出：

查看胜率：



查看徽章：

6 实验总结

本次程序设计实践采用 C++ 语言，并综合运用了 qt 的 GUI 编程以及 socket 编程，巩固复习了 C++ 的基础知识，对类的继承，多态以及封装有了更深刻的认识。此次程序设计使用了 socket 编程，实际使用了编程来控制网络传输数据，对网络编程也加深了理解。尤其是网络传输数据时的数据序列化，对

网络数据的传输起着至关重要的作用，可以极大简化数据的封装与读取。此次的 C++ 程序设计实践锻炼了实际编程的能力，把编程知识实际运用到了现实中，增加了编程的信心。