

宠物小精灵对战系统

目录

1. 题目
2. 开发环境以及使用技术
3. 程序框架设计介绍与代码细则
4. 使用流程说明
5. 实验总结和感想

1. 题目

题目一：
宠物小精灵的加入

要求：

设计宠物小精灵的类，为简化游戏设计，精灵的属性包括种类（力量型：高攻击； 肉盾型：高生命值； 防御型：高防御； 敏捷型：低攻击间隔，共四种）、名字、等级、经验值、攻击力、防御力、生命值、攻击间隔等（以上属性必须，其他属性可自行添加）（基本要求：本游戏中只有上面的4种类型。

进一步要求：上述4种类型可以进一步深入划分，比如力量型又可以再细分为：沙瓦朗、火爆猴、腕力等）每个精灵初始等级为1，满级15级，每当精灵升级的时候，宠物对应的属性值会有少量增加（主属性增加量相对较多）每个精灵有自己独特的攻击方式，如“闪电攻击”，“火焰攻击”等等，请设计一个精灵的基类，并将精灵的攻击方法设为虚方法以方便子类重写

请写一个测试程序对设计的精灵类的相关属性和方法（包括攻击函数，升级函数等）进行测试

题目主要考察点：类的继承，对象数据成员设计，成员函数设计

题目二：用户注册与平台登录

要求：

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

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

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

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

题目主要考察点：socket通信，交互场景反馈，用户信息存储方式，界面交互，其它合理的新颖设计。

题目三：游戏对战的设计

要求：

已经登录的在线用户可以和服务器进行虚拟决斗，决斗分两种：升级赛和决斗赛，两种比赛都能增长精灵宠物经验值。服务器上有一个虚拟精灵的列表，用户可以挑选其中任意一个进行比赛（升级赛或者决斗赛）。另外决斗赛中用户胜出可以直接获得该战胜的精灵，失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。

升级赛 只是用户用来增加精灵经验值，规则开发者自定；

累积多少经验值升一级，规则开发者自定；

决斗赛的上述规则同升级赛，只是额外还可以赢得一个宠物或失去一个宠物。

用户如果没有精灵（比如总是失败，已经全部送出去），则系统会随机放给他一个初级精灵。

请让你的系统自动模拟每场比赛的每次出招。另外，为了增加不确定性，可以加入概率闪避攻击和暴击伤害机制

比赛的过程和结果由系统根据上述规则自动模拟完成，要求结果具有一定的随机性。

用户增加新功能，可以查看某个用户的胜率

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），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

题目主要考察点：客户端与服务器数据交互（可采用多进程或异步通信或其他方法均可），并发请求处理，类的方法设计，伤害计算方法设计，界面交互，其它合理的新颖设计。

软件设计要求

如有必要的友元函数，要在报告（课程设计报告）和程序中说明每个友元函数的不可替代性，为什么一定要用友元才能实现。

自己编写的代码，除主函数和必要的友元函数外，不允许出现任何一个非类成员函数。

任何不改变对象状态（不改写自身对象数据成员值）的成员函数均需显式标注const。

代码规范性要求

代码需遵循课件提出的编码规范要求。

通过开发环境自动生成的界面类代码，全部数据成员和成员函数需在类声明时加以注释，函数体内的必要步骤要加以注释。

其他全部类代码的数据成员和成员函数的声明和实现均需加以注释，成员函数的必要步骤要加以注释。

2.开发环境以及使用技术

开发环境：win10 & QT5.12.9 & MinGW 64-bit

其他依赖：MySQL 8.0.21 x64（本地）& MySQL 5.5.62 x64（云端）

使用说明：

- 本程序客户端Client和服务器Server分开编写，因此需要分开运行两个软件。**本地先运行服务器再运行客户端。**为了方便起见本程序默认两个程序在本地运行，如需远程连接，请修改客户端源码中127.0.0.1为目标服务器IP地址。
- win10下运行请确保有QT本身需要的运行环境（EXE文件夹中有运行相关的DLL文件），**本地无对应的MySQL库请使用云端版本服务器。**
- 本程序支持多用户端

3.程序框架设计介绍与代码细则

数据库组件

数据均存在MySQL中users中

用户信息表：

ID	NAME	PASSWORD	ALLCOUNT	WINCOUNT
序号（自动递增）	名字	密码	总场数	胜利场数
int	string	string	int	int

精灵信息表：

TYPE	NAME	LEVEL	EXP	ATTACK	DEFENSE	HP	ATTINTERVAL
类型	名字	等级	经验	攻击力	防御力	血量	攻击间隔
string	string	int	int	int	int	int	int

本地服务器实现本地MySQL连接：

```
//数据库连接
QSqlDatabase db = QSqlDatabase::addDatabase("QODBC");
db.setHostName("127.0.0.1");
db.setPort(3306);
db.setDatabaseName("MyServer");
db.setUserName("root");
db.setPassword("lky1161");

bool ok = db.open();
if (ok)
{
    QMessageBox::information(this, "infor", "数据库登录成功");
    qDebug() << "数据库登录成功";
}
else
{
    QMessageBox::information(this, "infor", "数据库登录失败");
    qDebug() << "数据库登录失败" << db.lastError().text();
}
```

云端服务器实现云端MySQL连接：

```
//数据库连接
QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");
db.setHostName("140.143.142.112");
db.setPort(3306);
db.setDatabaseName("pokemon");
db.setUserName("pokemon");
db.setPassword("poke1161");

bool ok = db.open();
if (ok)
```

```

{
    QMessageBox::information(this, "infor", "数据库登录成功");
    qDebug() << "数据库登录成功";
}
else
{
    QMessageBox::information(this, "infor", "数据库登录失败");
    qDebug()<<"数据库登录失败"<<db.lastError().text();
}

```

与数据库的交互:

例 注册成功时候:

```

query.exec("use pokemon");
query.exec( QString("select * from users where name = '%1';").arg(namestring));
while(query.next())
{
    temname = query.value("name").toString();
}
//如果不存在, 即注册成功
if(temname == NULL)
{
    query.exec( QString("insert into users(name, password,allcount,wincount,"
        "havePowerPoke,haveTankPoke,haveDefensePoke,haveQuickPoke) "
        "values('%1', '%2',%3,%4,%5,%6,%7,%8);")
        .arg(namestring).arg(passwordstring).arg(0).arg(0).arg(0).arg(0).arg(0).arg(0));
    ui->textRead->append("用户: " + namestring + "注册成功! ");
}

```

通信组件

为了实现客户端和服务器的交流, 我自定义了所有消息的报文格式; 为了防止TCP黏包, 一次发送一整条完整信息, 再拆开分解。

服务器端:

socket通信:

###监听: ###

```

#include <QTcpServer> //监听socket
//监听套接字, 指定父对象自动回收空间
tcpServer = new QTcpServer(this);

tcpServer->listen(QHostAddress::Any, 8888);

//建立新连接触发槽函数
connect(tcpServer, &QTcpServer::newConnection,
        this, &Widget::newconnection);

```

```

//新用户连接

```

```

void widget::newconnection()
{

qDebug() << "新客户端登陆:";
QTcpSocket *newClient;

//得到新进来的socket, 用于标识刚刚连接的新用户
newClient = tcpServer->nextPendingConnection();
//连接标志
clientJoinUp_check = true;

//在客户端列表最后添加新的socket
clients_checks[currentClient].client_socket = newClient;
currentClient++;
tcpSocket = newClient;

//在对话框显示谁和我连接
//获取对方IP和端口
QString ip = tcpSocket->peerAddress().toString();
qint16 port = tcpSocket->peerPort();
QString temp = QString("[IP:%1:Port%2]:成功连接").arg(ip).arg(port);

ui->textRead->append(temp);

//接收消息
connect(tcpSocket,&QTcpSocket::readyRead,this,&widget::readmessage);

}

```

为了实现多用户连接，服务器端类中有一个储存所有连接用户的结构体，通过它来寻找用户名和TCPSOCKET之间的关系：

```

//储存所有连接过的用户
struct clients_check
{
    QTcpSocket *client_socket;
    QString namestring = "NULL";
    bool issignIn = false;
}clients_checks[MAXCONNECT];

```

通信：

来自客户端不同消息的处理：

```

//处理信息细分
//注册信息 0
void dealSignUp(QString str,QTcpSocket *client_tcp);
//登录信息 1
void dealSignIn(QString str,QTcpSocket *client_tcp);
//断开信息 2
void currentClientChangeToFlase(QTcpSocket *client_tcp);//注销后改变目标用户数组的登录状态
//申领精灵 3
void dealGetPoke(QTcpSocket *client_tcp);
//查看用户 4

```

```

void dealGetUser(QTcpSocket *client_tcp);
//查看徽章 5
void dealGetBadge(QTcpSocket *client_tcp);
//查看精灵 6
void dealCheakPoke(QString str,QTcpSocket *client_tcp);
//查看出战精灵 7
void dealCheakFightPoke(QTcpSocket *client_tcp);
//处理战斗后结果 8
void dealFightResult(QString str,QTcpSocket *client_tcp);

```

例子：查看用户 4 void dealGetUser(QTcpSocket *client_tcp);

```

void widget::dealGetUser(QTcpSocket *client_tcp)
{
    QString str;
    QString ratestr;
    for(int i = 0;i < currentClient;i++)
    {
        str = "4 user:" + clients_checks[i].namestring + " is ";
        if(clients_checks[i].isSignIn == false)
        {
            str += "offline ";
        }
        else
            str += "online ";
        str += "rate:" +
        QString::number(getRateFromDatabase(clients_checks[i].namestring) * 100,'f',1) +
        ";";
        sendMessageToClient(str,client_tcp);
    }
}

```

客户端：

与服务器通信框架：

```

//*****服务器通信
*****//
void connectToServer(); //连接服务器
void sendMessageToServer(QString message); //向服务器发送信息
void receiveMessageFromServer(); //从服务器收到信息的函数
void dealMessage(QString str); //处理服务器传来的信息
//处理返回信息细分
void dealSignUpBack(QString str); //注册信息返回处理 0
void dealSignInBack(QString str); //登录信息返回处理 1
void dealGetPokeBack(QString str); //申领信息返回处理 3
void dealGetUserListBack(QString str); //查看用户信息返回处理 4
void dealGetMyBadgeBack(QString str); //查看徽章信息返回处理 5
void dealCheakMyPokeBack(QString str); //查看精灵信息返回处理 6
void dealCheakFightPokeBack(QString str); //查看征战精灵信息返回处理 7

```

```

//处理发送信息细分
void enterGameWidget();           //登陆成功进入游戏界面
void closeMainGameDeal();        //游戏界面关闭注销账号 2
void getPokeDeal();              //游戏界面领取精灵处理 3
void getUserList();              //游戏界面查看用户列表 4
void getMyBadge();               //游戏界面查看我的徽章 5
void getMyPoke();                //游戏界面查看我的精灵 6

```

例子1：处理从服务器返回的注册信息

```

void MainWidget::dealSignupBack(QString str)
{
    switch (str[1].toLatin1() - '0')
    {
        case 1:           //注册成功
            QMessageBox::information(this,"好啊","注册成功!");
            qDebug() << "注册成功";
            break;
        case 0:           //注册失败
            QMessageBox::information(this,"不好","用户名已存在，注册失败");
            qDebug() << "注册失败";
            break;
    }
}

```

例子2：本地比赛胜利或失败信息传给服务器 注意自定义报文格式

//发送信息格式：

//胜利8 isWin whichgame isLevel isGet enemyNum

//失败8 isWin whichgame 0 isLose myNum

```

void MainWidget::dealWinOrLose(bool iswin,int myNum)
{
    int isLevelUp = 0;
    int isGet = 0;
    int isLose = 0;
    QString str;
    //发送信息格式:
    //胜利8 iswin whichgame isLevel isGet enemyNum
    //失败8 iswin whichgame 0 isLose myNum
    if(iswin)//战斗胜利
    {
        if(myPoke[myNum].getExp() >= 50)//能升级
        {
            isLevelUp = 1;
            battlewidget.setText("精灵" + myPoke[myNum].getName() + "升级啦!");
        }
        if(whichgame == 0)//决斗赛
        {
            if((enemyPoke.getType() == powerPoke && myPoke[0].getHave() == 0) ||
                (enemyPoke.getType() == tankPoke && myPoke[1].getHave() == 0) ||
                (enemyPoke.getType() == defensePoke && myPoke[2].getHave() == 0) ||
                (enemyPoke.getType() == quickPoke && myPoke[3].getHave() == 0)
            )
            {

```

```

        isGet = 1;
        battlewidget.setText("获得敌方力量型精灵!");
    }
    else
        battlewidget.setText("已经拥有对方精灵, 无法获得!");
}
str = "8 1 " + QString::number(whichgame) + " " + QString::number(isLevelUp)
+
    " " + QString::number(isGet) + " " + enemyPoke.getType();
sendMessageToServer(str);
}
else//战斗失败
{
    if(whichgame == 0)//决斗赛
    {
        isLose = 1;
        battlewidget.setText("失去当前精灵!");
    }
    str = "8 0 " + QString::number(whichgame) + " 0 " + QString::number(isLose)
+ " " + myPoke[myNum].getType();
    sendMessageToServer(str);
}
}
}

```

小精灵类设计

基类PokemonBase设计:

```

class PokemonBase
{

public:
    PokemonBase();
    //重载构造函数
    //自定义数值构造
    PokemonBase(QString m_name, int m_level, int m_exp, int m_attack, int m_defense, int
    m_hp, int m_attInterval);
    //初始数值构造
    PokemonBase(QString m_name, pokemonType m_type);
    ~PokemonBase();

public:
    //查看接口
    QString getName();
    pokemonType getType();
    int getHave();
    int getLevel();
    int getAttack();
    int getDefense();
    int getHp();
    int getAttInterval();
    int getExp();

    //设置接口
    void setName(QString newName);
    void setHave(int have);
    void setType(pokemonType newtype);
}

```



```

void setLevel(int newLevel);
void setAttack(int newAttack);
void setDefense(int newDefense);
void setHp(int newHp);
void setAttInterval(int newAttInterval);
void setExp(int newexp);
//增加接口
void addLevel();
void addExp(int addexp);
//这四类要根据类型决定
void addAttack();
void addDefense();
void addHp();
void addAttInterval();

//独特的战斗技能函数
virtual int fightFight(PokemonBase *){};

//战斗函数
bool fightwithPoke(PokemonBase *);
//私有属性
private:
QString name_poke;      //名称
pokemonType type_poke;  //类别
int isHave = 0;         //是否拥有
int level;              //等级
int exp;                //经验值
int attack;             //攻击力
int defense;            //防御力
int hp;                 //生命值
int attInterval;        //攻击间隔
};

```

四种不同子类设计，其中重写了父类中虚函数特殊攻击

```

class PokemonPower : public PokemonBase
{
public:
    //力量型特殊攻击 大力出奇迹
    //造成1.5倍攻击力
    PokemonPower();
    PokemonPower(QString m_name,int m_level,int m_exp,int m_attack,int
m_defense,int m_hp,int m_attInterval);
    int fightFight(PokemonBase *);
};

class PokemonTank : public PokemonBase
{
public:
    //肉盾型特殊攻击 肉蛋冲击
    //造成当前攻击力+血量三分之一
    int fightFight(PokemonBase *);
    PokemonTank();
    PokemonTank(QString m_name,int m_level,int m_exp,int m_attack,int
m_defense,int m_hp,int m_attInterval);
};

```

```

class PokemonDefense : public PokemonBase
{
public:
    //防御型特殊攻击 攻防合一
    //造成当前攻击力+防御力
    int fightFight(PokemonBase *);
    PokemonDefense();
    PokemonDefense(QString m_name,int m_level,int m_exp,int m_attack,int
m_defense,int m_hp,int m_attInterval);

};

class PokemonQuick : public PokemonBase
{
public:
    //敏捷型特殊攻击 连打
    //打两下 造成两倍伤害
    int fightFight(PokemonBase *);
    PokemonQuick();
    PokemonQuick(QString m_name,int m_level,int m_exp,int m_attack,int
m_defense,int m_hp,int m_attInterval);

};

```


战斗时的概率暴击或闪避请见后面战斗页面。

客户端页面介绍

MainWidget:

本界面负责与服务器通信以及登录注册，登录成功隐藏，在后台默默处理一切通信信息，构造时初始化其他所有页面。

登录



Name

lucien

Password

●●●●●●

☐ 显示密码

ip

127.0.0.1

端口

8888

登录

注册

MainGame:

本页面负责登录后游戏大厅

```
class MainGame : public QWidget
{
Q_OBJECT

public:
explicit MainGame(QWidget *parent = nullptr);
~MainGame();
QString getLineName(); // 获取名字输入行信息

signals:
void closeMainGame(); // 关闭窗口信号
void cheakGetPokeSignal(); // 按钮->获取精灵信号
void cheakUserListSignal(); // 按钮->查看用户信号
void cheakMyBadgeSignal(); // 按钮->查看我的徽章
void cheakMyPokeSignal(); // 按钮->查看我的精灵
void enterDuelSignal(int which_game); // 按钮->进入决斗赛 0
void enterUpgradeSignal(int which_game); // 按钮->进入升级赛 1
```

```
protected:
void closeEvent(QCloseEvent *event); //游戏窗口关闭时触发的函数
//重写绘图虚函数
void paintEvent(QPaintEvent *);
private slots:
```

```
void on_buttonGetPoke_clicked();

void on_buttonUserList_clicked();

void on_buttonMyBadge_clicked();

void on_buttonMyPoke_clicked();

void on_buttonDuel_clicked();

void on_buttonUpgrade_clicked();

private:
Ui::MainGame *ui;
};
```



PokeList:

本页面用于查看目标用户拥有的精灵

UserList:

本页面用于查看当前在线用户以及他们的胜率

BadgeWidget:

本页面用于查看我的勋章

ChooseWidget:

本页面用于点击比赛后选择出战精灵



BattleWidget:

本页面用于查看战斗结果

其中暴击与闪避在这里处理，通过产生随机数来决定是否暴击或者闪避

其中case 1为使用技能，代码过于繁琐暂不列出，可去源码看

```
time= QTime::currentTime();
qrand(time.msec()+time.second()*1000 + round);
n = qrand() % 10;    //产生0-9随机数
```

```
switch (n)
{
case 0://暴击
    temHp = (myPoke->getAttack() - enemyPoke->getDefense()) * 2;
    enemyPokeHp -= temHp;
    ui->textBrowser->append(myPoke->getName() + "触发暴击，造成了" +
QString::number(temHp) + "点伤害！" + enemyPoke->getName() +
"还剩下" + QString::number(enemyPokeHp) + "点血量");
    break;
case 2://闪避情况
    temHp = 0;
    enemyPokeHp -= temHp;
    ui->textBrowser->append(myPoke->getName() + "攻击被闪避，造成了" +
QString::number(temHp) + "点伤害！" + enemyPoke->getName() +
"还剩下" + QString::number(enemyPokeHp) + "点血量");
    break;
default://正常情况
    temHp = myPoke->getAttack() - enemyPoke->getDefense();
    enemyPokeHp -= temHp;
    ui->textBrowser->append(myPoke->getName() + "造成了" +
QString::number(temHp) + "点伤害！" + enemyPoke->getName() +
"还剩下" + QString::number(enemyPokeHp) + "点血量");
```

```
break;
```

Form

我方精灵信息:
名字: Mango
类别: 肉盾型
等级: 15 经验值:0 攻击力:30 防御力:0 生命值:120 攻击间隔:50
敌方精灵信息:
名字: enemy
类别: 肉盾型
等级: 5 经验值:0 攻击力:35 防御力:8 生命值:100 攻击间隔:50
Mango触发暴击, 造成了44点伤害! enemy还剩下56点血量
enemy触发暴击, 造成了70点伤害! Mango还剩下50点血量
Mango造成了22点伤害! enemy还剩下34点血量
enemy造成了35点伤害! Mango还剩下15点血量
Mango造成了22点伤害! enemy还剩下12点血量
enemy造成了35点伤害! Mango还剩下-20点血量
敌方精灵胜利!

4.使用流程说明

- 首先打开服务器端，本地有对应Mysql的pokemon库就使用PokemonServer版本的客户端；若本地无则使用云端Mysql版本的cloudServer。
- 打开显示连接数据库成功后打开客户端PokemonClient，可以注册和登录，其中有注册登录字符限制和失败提醒。
- 登录成功后进入页面，在查看精灵下面有一个输入栏，在不输入任何字符的情况下点击查看精灵会默认查看自己的精灵；输入目标用户的名字后会查询目标用户的精灵。
- 点击我的徽章可以查看自己的徽章情况，分为等级和数量徽章，徽章要求符合题目。
- 点击用户列表可以查看当前所有在线用户，并且会显示他的胜率。
- 点击领取宠物会在你没有宠物的时候随机选一只送给你，在有宠物的情况下会提示失败。
- 点击升级赛和决斗赛会进入选择页面，选择完自己的精灵和目标精灵后即可战斗。
- 战斗页面会显示所有信息。
- 本程序支持多用户登录，可以多次打开一个服务器端和多个用户端。如果要在两台电脑上分别打开客户端和服务端请修改客户端源码中服务器地址，因为客户端只和服务端交互，无法直接和数据库交互，必须需要目标服务器。

##5.实验总结和感想

在本次实验中，我独立编写代码，锻炼了C++的编程能力、QT库的编程能力，对面向对象有了更深刻的了解，了解了什么是类、虚函数、重载等等面向对象特有功能；对TCP\SOCKET通信有了更深刻的了解，可以类似本程序的建立一个TCP的聊天室；运用了Mysql数据库，对数据储存有了更多的了解；还使用了云服务器储存mysql，对云上的架构有了了解。

此外在编写代码过程中，遇到了许许多多的困难，例如QT的qmysql驱动需要自己去库中编译出dll，又比如使用sql语言时候不能有一丝错误。这些具有难度的工作都让我受益匪浅，通过这一次的实验我学习了比目标更多的知识，在编程方面也更加熟练，总的来说是一次圆满又具有意义的实验。