

# 面向对象程序设计与实践

## 宠物小精灵

班级\_\_\_\_\_2016211306\_\_\_\_\_

学号\_\_\_\_\_2016211284\_\_\_\_\_

姓名\_\_\_\_\_黄浩强\_\_\_\_\_

指导老师\_\_\_\_\_双锴\_\_\_\_\_

目录

- 一、作业 1 宠物小精灵的加入 ..... 3
  - 1 程序功能 ..... 3
  - 2 程序结构及面向对象设计 ..... 3
    - 2.1 程序结构 ..... 3
    - 2.2 设计 ..... 4
    - 2.3 重要代码 ..... 5
  - 3 程序测试 ..... 8
- 二、作业 2、用户注册与平台登录 ..... 9
  - 1 程序功能 ..... 9
  - 2 程序结构及面向对象设计 ..... 10
    - 2.1 程序结构 ..... 10
    - 2.2 设计 ..... 12
    - 2.3 重要代码 ..... 14
  - 3 程序测试 ..... 19
- 三、作业 3 游戏对战的设计 ..... 22
  - 1 程序功能 ..... 22
  - 2 程序结构及面向对象设计 ..... 23
    - 2.1 程序结构 ..... 23
    - 2.2 设计 ..... 26
    - 2.3 重要代码 ..... 27
  - 3 程序测试 ..... 34

## 一、作业 1 宠物小精灵的加入

### 1 程序功能

(1) 要求设计宠物小精灵的类，属性包括包括种类（力量型、肉盾型、防御型、敏捷型，共四种）、名字、等级（上限 15，升级时增强其他属性）、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）

(2) 进一步要求：上述 4 种类型可以进一步深入划分，比如力量型又可以在细分为：沙瓦朗、火爆猴、腕力等。

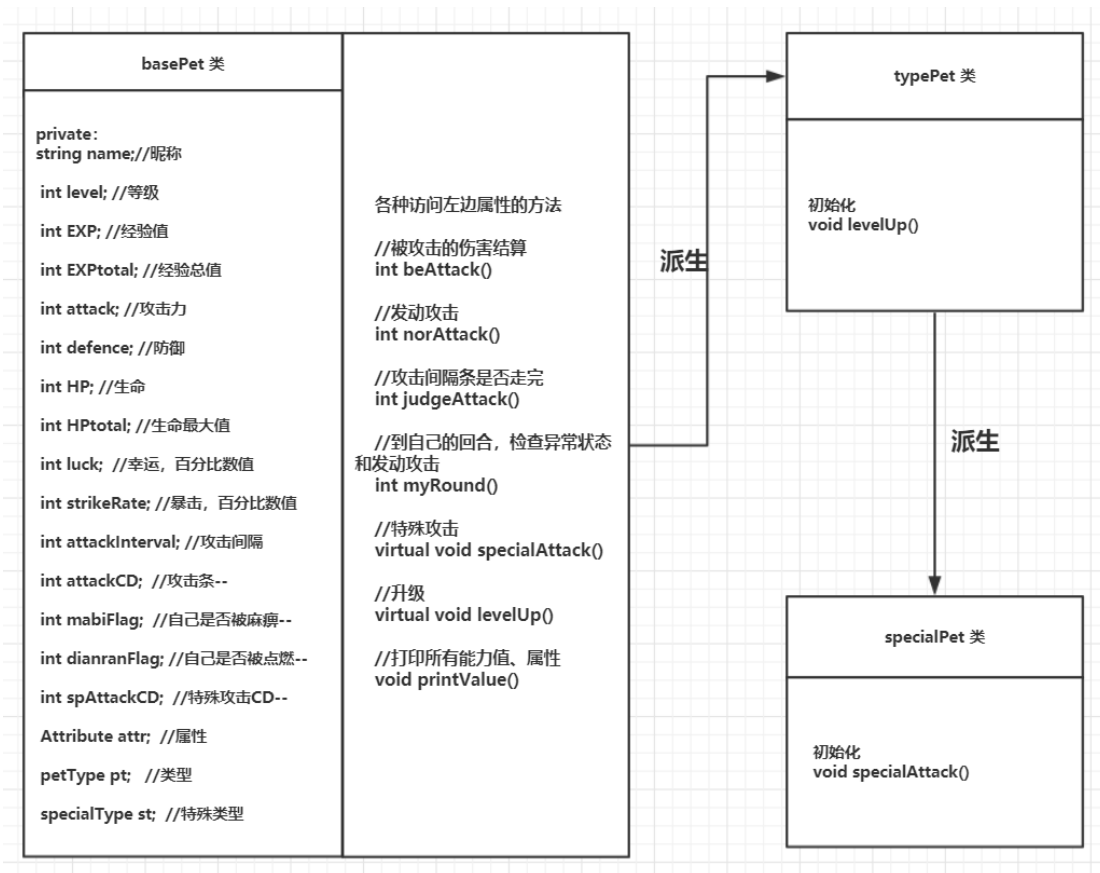
(3) 每个精灵有自己独特的攻击方式，如“闪电攻击”，“火焰攻击”等等，请设计一个精灵的基类，并将精灵的攻击方法设为虚方法以方便子类重写

(4) 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

### 2 程序结构及面向对象设计

#### 2.1 程序结构

如图所示，程序中主要包含三个类，分别是 basePet 类、typePet 类、specialPet 类。



### 2.1.1 basePet 类

包含了小精灵所需要的各种基本能力值和异常状态，如攻击、经验、点燃剩余等，和访问、修改他们的方法；以及作为战斗系统基础的几个函数。

### 2.2.2 typePet 类

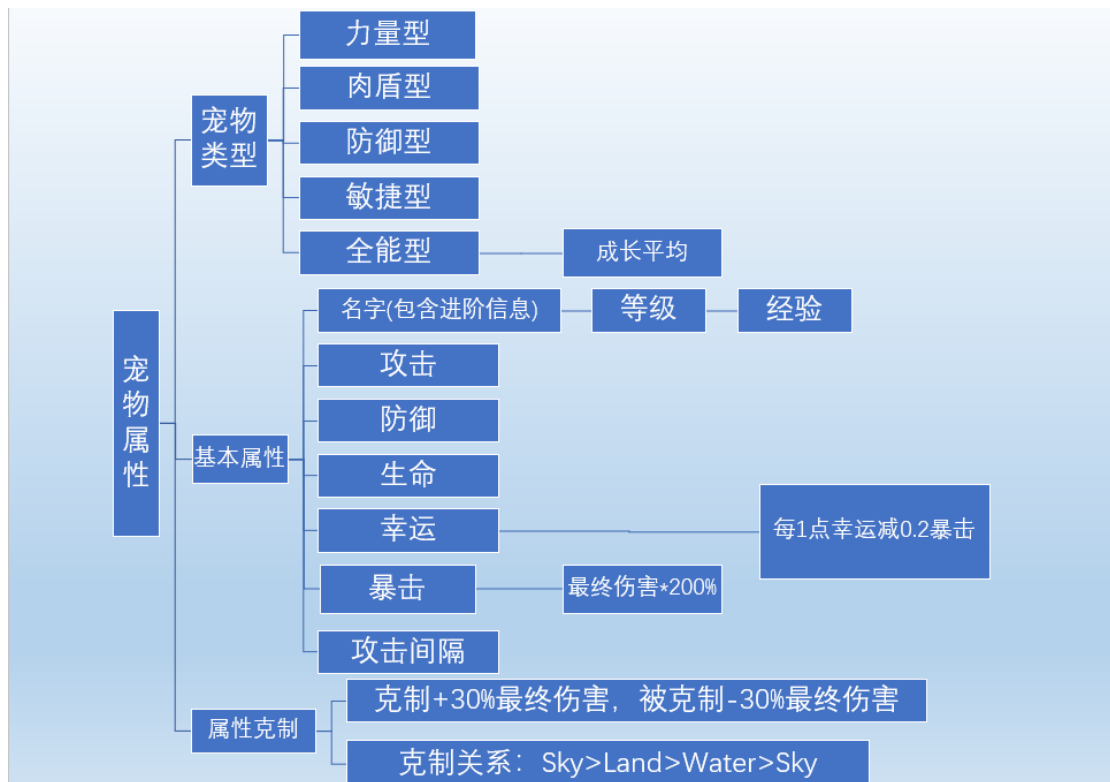
继承 basePet 类, 初始化函数将会设置五种基本类型的初始能力值和宠物类型 petType; levelUP 函数完成升级功能，对精灵的部分能力值进行提高。

### 2.2.3 specialPet 类

继承 specialPet 类，初始化函数将设置精灵的属性 attr（涉及属性相克）。重写 specialAttack()函数，完成特殊攻击的功能。

## 2.2 设计

精灵系统设计如下：



## 2.3 重要代码

### 2.3.1 伤害计算：

公式：最终伤害 = (int)((攻击-防御) \* (暴击-幸运\*0.2)) \* 属性克制

代码：

```

int beAttack(const pet& Att)
{
    int tAttack = Att.getAttack();

    int tStrikeRate = Att.getStrikeRate();

    Attribute tAttr = Att.getAttr();

    int restrainFlag = 0; //-1为本精灵克制对方，0为同属性，1为对方克制本精灵

    int hurt = (int)(tAttack - this->defence);

    if (hurt < 0) {
        cout << "攻击太低啦没有伤害2333" << endl;

        return 0;
    }
}
  
```

```

    }

    srand((unsigned)time(NULL));

    int rate = rand() % 100;

    if ((int)(tStrikeRate - (this->luck)*0.2) > rate / 10) {

        cout << "暴击! " << endl;

        hurt = hurt * 2;

    }

    if (this->attr == tAttr)

        ;

    else if ((attr == Sky && tAttr == Land) || (attr == Land && tAttr == Water) || (attr ==
Water && tAttr == Sky))

        restrainFlag = -1;

    else

        restrainFlag = 1;

    hurt = (int) hurt * (1 + restrainFlag*0.3);

    if (hurt > 0) {

        this->HP -= hurt;

    }

    if (HP < 0) {

        HP = 0;

        return 1;

    }

    return 0;

};

```

### 2.3.2 升级（以全能型为例）

每打败一只小精灵，对方当前等级下经验最大值的 20%作为自己获得经验。

```

void levelUp()

{

    int EXP = this->getEXP() - this->getEXPtotal();

```

```

int EXPt = this->getEXPtotal() + 100;

chEXP(EXP);

chEXPtotal(EXPt);

chAttack(getAttack() + 20);

chDefence(getDefence() + 15);

chHPtotal(getHPtotal() + 100);

chHP(getHPtotal() + 100);

int lu = getLuck() + 5;

if (lu > 100)          //幸运溢出

    chLuck(100);

else

    chLuck(lu);

chStrikeRate(getStrikeRate() + 3);

}

```

### 3. 重写的特殊攻击（以皮卡丘为例）

//雷暴，攻击力额外提高20%，持续1回合；同时麻痹对方1回合

```

int spAttack(pet &BeAtt)

{

    BeAtt.chMabiFlag(1);

    cout << "您发动雷暴，攻击力提高20%，持续1回合；对方麻痹1回合" << endl;

    int tA = this->getAttack();

    chAttack((int)tA*1.2);

    int res = norAttack(BeAtt);

    chAttack((int)tA);

    this->chSpAttackCD(4);

    return res;

}

```

### 3 程序测试

经测试，程序能正常实现功能。

#### 3.1 攻击函数测试

如图所示，攻击函数能完成指定目标，特殊攻击正常发挥作用。

```
获得一只水箭龟！请输入昵称：我是水箭龟
获得一只皮卡丘！请输入昵称：我是皮卡丘
回合CD: 2      昵称：我是水箭龟
回合CD: 1      昵称：我是皮卡丘
水箭龟HP: 2000/2000
皮卡丘HP: 1000/1000
回合CD: 1      昵称：我是水箭龟
回合CD: 0      昵称：我是皮卡丘
现在我【我是皮卡丘】的回合！
特殊攻击CD: 0. 处于CD中时，不允许特殊攻击，否则视为无效输入
输入1普通攻击，输入2特殊攻击: 2
您发动雷暴，攻击力提高20%，持续1回合；对方麻痹1回合
暴击！
水箭龟HP: 1636/2000
皮卡丘HP: 1000/1000
回合CD: 0      昵称：我是水箭龟
现在我【我是水箭龟】的回合！
哎呀我正在麻痹中，麻痹剩余0回合
回合CD: 1      昵称：我是皮卡丘
水箭龟HP: 1636/2000
皮卡丘HP: 1000/1000
回合CD: 2      昵称：我是水箭龟
回合CD: 0      昵称：我是皮卡丘
现在我【我是皮卡丘】的回合！
特殊攻击CD: 3. 处于CD中时，不允许特殊攻击，否则视为无效输入
输入1普通攻击，输入2特殊攻击: _
```

#### 3.2 升级函数测试

如图所示，升级函数能完成指定目标的升级。



```
暴击！
恭喜【我是皮卡丘】获得胜利！
水箭龟HP：0/2000
皮卡丘HP：582/1000
请按任意键继续. . .
属性表：
昵称：我是皮卡丘
等级：1
经验：47
经验总值：234
攻击力：200
防御：75
HP：582
HPtotal：1000
幸运：40
暴击率：30
攻击间隔：2
麻痹剩余：0
点燃剩余：0
特殊攻击cd：0
属性：Land
类型：全能型
演示P升级：
已升级！
属性表：
昵称：我是皮卡丘
等级：1
经验：47
经验总值：334
攻击力：220
防御：94
HP：1200
HPtotal：1100
幸运：44
暴击率：32
攻击间隔：2
麻痹剩余：0
点燃剩余：0
特殊攻击cd：0
属性：Land
类型：全能型
请按任意键继续. . .
```

## 二、作业 2、用户注册与平台登录

### 1 程序功能

实现用户账号的登录、注册、登出功能，不允许有两个相同账号同时登录、不允许有两个同名账号。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（文件存储或数据库均可，数据库有额外加分）。同时，用户可以查看注册成功用户拥有的精灵和当前在线的用户。

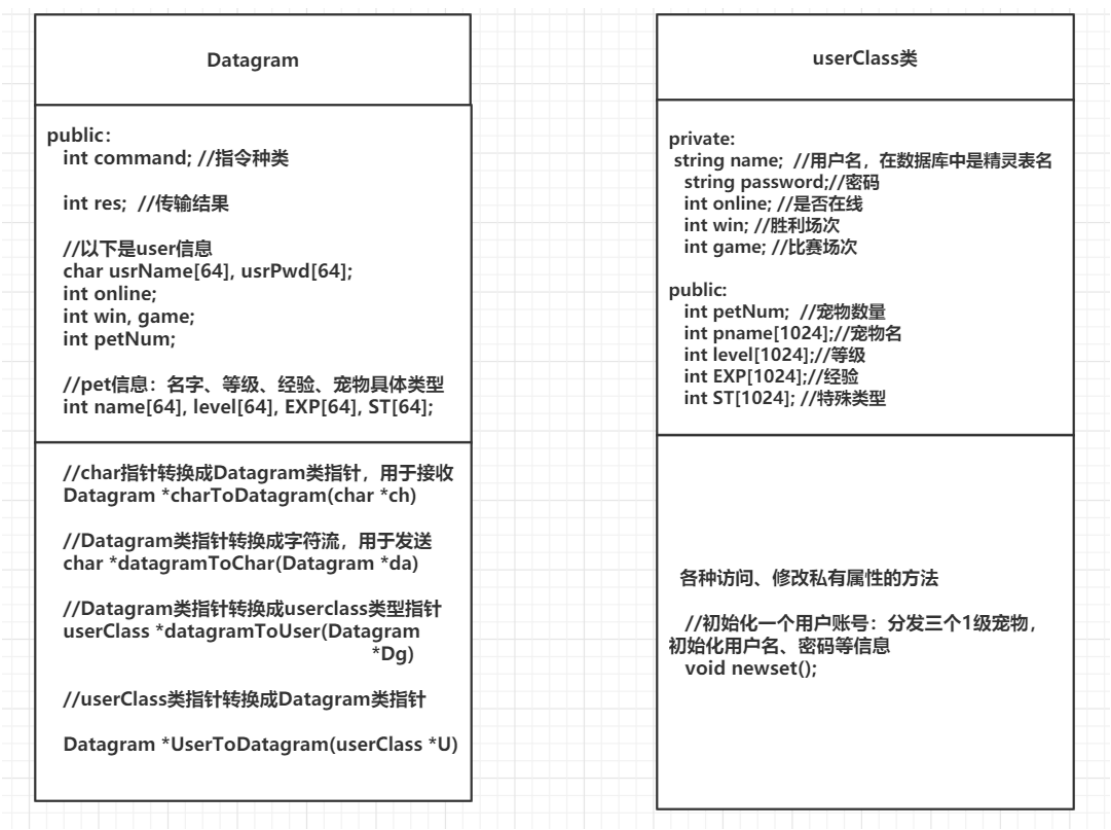
每个用户拥有：用户名、拥有的精灵，两个属性。 用户注册成功时，系统自动随机分发三个 1 级精灵给用户

## 2 程序结构及面向对象设计

### 2.1 程序结构

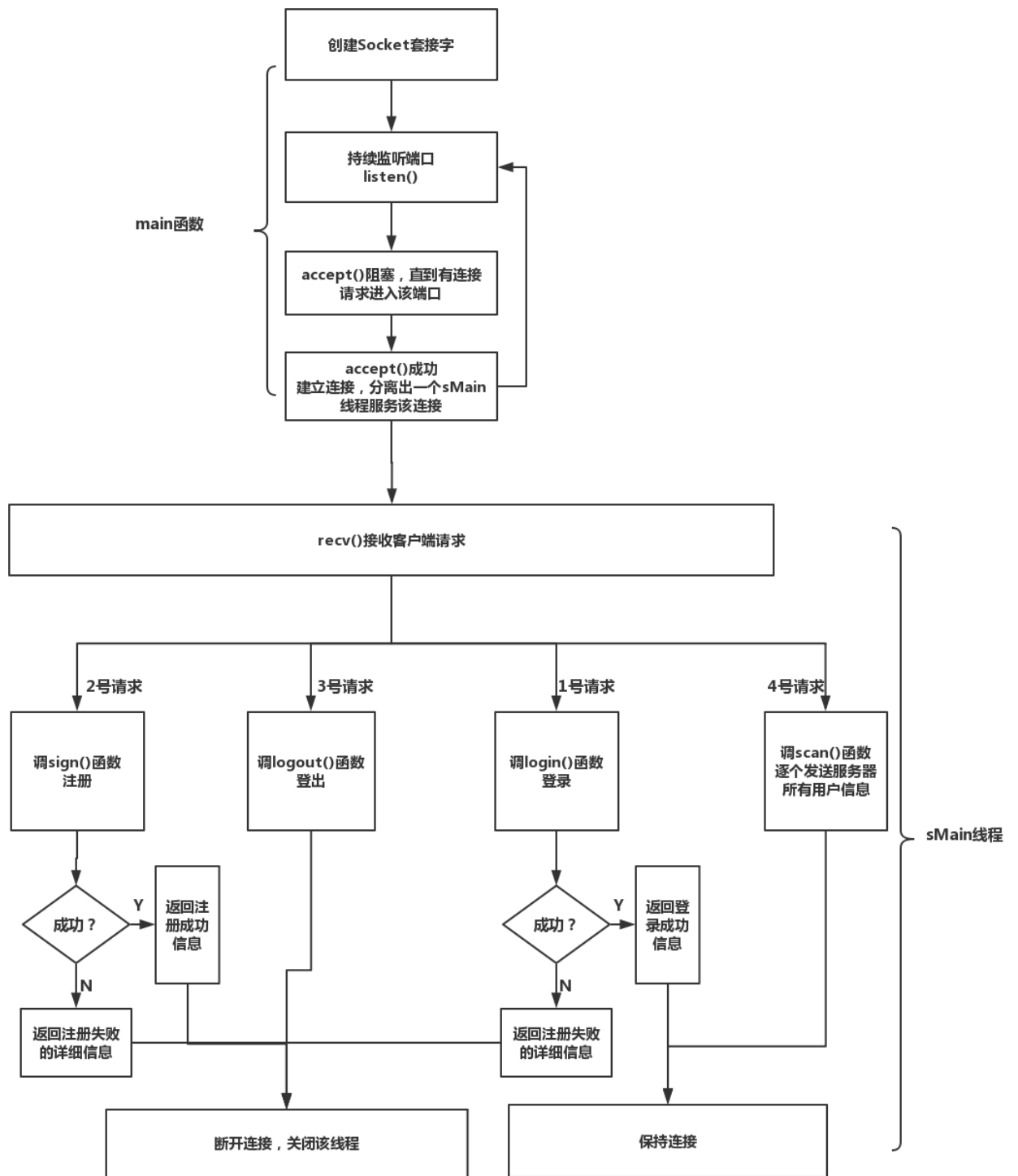
1. 从第一版的程序发展而成，分成服务端、客户端两个互相独立的程序，采用 C/S 模式。服务端采用 Visual Studio 编写，客户端采用 QT 编写。
2. 使用 socket 通信，相比第一版，增加了 Datagram、userClass 两个类。其构造如图所示：

其中，Datagram 用于传输，含有在发送接收时作数据转换的方法；userClass 类存储用户信息，便于传输通信和与 MySQL 结合使用，发送接收时与 Datagram 搭配使用。



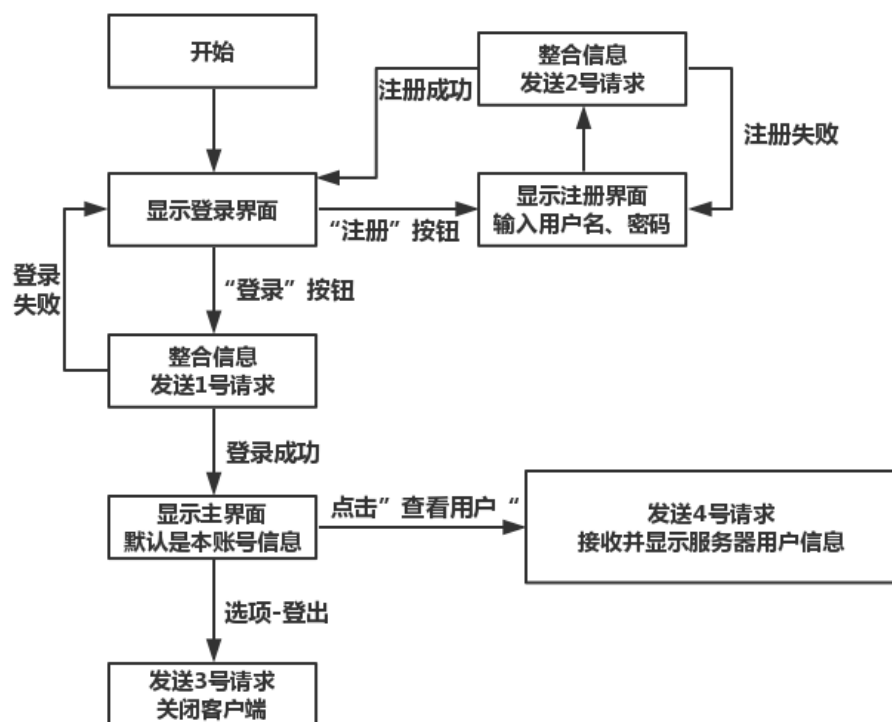
#### 2.1.1 服务端

处理请求和与数据库交互的流程如下



## 2.1.2 客户端

处理客户事件的流程如下：



## 2.2 设计

### 2.2.1 服务端

#### 1. 多客户端登录、运行

使用多线程（thread 类），accept()成功则创建一个 thread 并分离（detach()）。

另外，在 main.cpp 中维护一个全局变量 `vector<string> usrOnline`，记录当前登录的用户名字，若登录请求中的用户名在该 vector 中查找到，则不允许登录。

#### 2. 数据库（在第三版中有所改进）

使用 MySQL 数据库，在 main.cpp 中编写对数据库中变量的增删改查函数。对每位用户，在 user 表中存取用户信息，并对应的创建一张以用户名为名字的表，存取用户名下的小精灵信息。

数据库如下：

数据库过滤器 | 表过滤器 | 主机: 127.0.0.1 | 数据库: pet | 表: user | 数据 | 查询

petuser: 5 总记录 (大约)

name	password	petNum	win	game
123456	123	3	0	0
bbb	1	3	0	0
aaa	bbb	1	2	3
admin	12	5	0	0
123	1	3	3	7

Left sidebar shows database structure: localhost > information\_schema (160.0 KiB) > mysql > performance\_schema > pet (96.0 KiB) > 123 (16.0 KiB) > 123456 (16.0 KiB) > aaa (16.0 KiB) > admin (16.0 KiB) > bbb (16.0 KiB) > user (16.0 KiB)

### 3. 互斥

用两个互斥量 tMutex、oMutex。tMutex 作为访问数据库的互斥量，oMutex 作为访问 usrOnline 的互斥量，从而保证数据库和 usrOnline 的数据安全性和可信性。

## 2.2.2 客户端

### 1. 登录/注册的反馈信息

发送登录/注册信息并收到服务端的信息后，客户端根据 Datagram 中的 res 位判断结果，将会反馈用户：登录/注册成功；登录失败，用户已登录；登录失败，用户不存在或密码错误；注册失败，用户名已存在，这几种信息，以 QMessageBox:information 的方式提示用户。

### 2. 登出功能的实现

在登录未成功之前，服务器和客户端保持着“建立连接——发送请求——返回请求——断开连接”的方式通信。

在登录成功之后，服务器和客户端保持连接。通过在客户端主界面的析构函数中加入断开连接请求的方式，保证即使客户端不使用“登出”按钮，而是直接关闭客户端界面的情况下，也能够使服务器端正常登出。

### 3. 查看服务器其他用户功能的实现

客户端向服务器发送 4 号请求后，服务器根据查询结果会先返回一个 Datagram，其中包含用户的数量 n。客户端根据该信息建立循环，接收 n 位用户的信息，并显示在界面上。

### 4. 显示用户信息

在客户端维护全局变量 myAccount（userClass 类），用来显示用户信息。

### 5. 主界面

主界面用一个 stackedWidget 和三个按钮分成三部分，点击按钮跳到对应的 stackedWidget 内容，能达到类似“页面切换”的效果。

## 2.3 重要代码

### 2.3.1 服务端

#### 2.3.1.1 建立连接和分离线程

```
//创建套接字
WSAStartup(MAKEWORD(2, 2), &wsaData);
sListen = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(SERPORT);
serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);
memset(&(serverAddr.sin_zero), 0, sizeof(serverAddr.sin_zero));
res = ::bind(sListen, (SOCKADDR *)&serverAddr, sizeof(SOCKADDR));
if (res == SOCKET_ERROR)
{
    printf("bind failed!\n");
    return 3;
}
while (1) {
    //监听
    listen(sListen, 1);
    //建立连接
    clientAddrLen = sizeof(SOCKADDR);
    sClient = accept(sListen, (SOCKADDR *)&clientAddr, &clientAddrLen);
    if (sClient == INVALID_SOCKET) {        //连接建立失败
        printf("Accept failed!");
    }
    else {
        //接入线程
        printf("Accepted client: %s : %d\n", inet_ntoa(clientAddr.sin_addr),
ntohs(clientAddr.sin_port));
        thread task01(sMain, sClient);
        task01.detach();    //分离
    }
}
closesocket(sListen);

WSACleanup();
```

### 2.3.1.2 处理用户各种请求（以登录请求为例）

```
//处理登录请求
int login(SOCKET sClient, Datagram *Dg)
{
    int res;
    char *ch;
    char sql[1024];
    userClass *usrtmp;
    Datagram *datatmp;
    oMutex.lock();
    if (find(usrOnline.begin(), usrOnline.end(), Dg->usrName) != usrOnline.end()) {
        //当前用户在线
        oMutex.unlock();
        datatmp = new Datagram;
        datatmp->res = 2;
        ch = datagramToChar(datatmp);
        res = send(sClient, ch, 1023, 0);
        if (res == SOCKET_ERROR) { printf("send failed!1"); }
        return 0;
    }
    oMutex.unlock();
    //访问数据库，用户不存在或密码错误res位置0，否则置1并发送相关数据
    tMutex.lock();
    mysqlId.freeConnect(); mysqlId.connectPet();
    usrtmp = mysqlId.queUsr(Dg->usrName);
    if (usrtmp == NULL) { //用户不存在
        tMutex.unlock();
        datatmp = new Datagram;
        datatmp->res = 0;
        ch = datagramToChar(datatmp);
        res = send(sClient, ch, 1023, 0);
        if (res == SOCKET_ERROR) { printf("send failed!1"); }
        return 0;
    }
    else { //用户存在,回传数据，同时维护usrOnline
        mysqlId.freeConnect(); mysqlId.connectPet();
        datatmp = UserToDatagram(usrtmp); //传用户的小精灵数据
        datatmp->res = 1;
        ch = datagramToChar(datatmp);
        const char* chtmp = ch;
        res = send(sClient, chtmp, 1023, 0);
        if (res == SOCKET_ERROR) { printf("send failed!1"); return 0; }
    }
}
```

```

        else { usrOnline.push_back(usrtmp->getName()); }
    }
    tMutex.unlock();
    delete ch, datatmp;
    return 1;
}

```

### 2.3.1.3 与服务器的交互（以创建新用户为例）

```

//连接MySQL
void SQL::connectPet()
{
    pConn = mysql_init(NULL);
    if (NULL == mysql_init(pConn)) {
        cout << "初始化失败" << endl;
        return ;
    }
    //mysql_set_character_set(pConn, "utf8");
    //第2、3、4、5参数的意思分别是：服务器地址、用户名、密码、数据库名，第6个为mysql端口号（0为默认值3306）
    if (NULL == mysql_real_connect(pConn, "localhost", "root", "1", "pet", 3306, NULL, 0))
    {
        printf("无法连接数据库:%s", mysql_error(pConn));
        return ;
    }
}

//释放与MySQL的连接
void SQL::freeConnect()
{
    mysql_close(pConn);
}

//在创建新用户完成后，再调用此函数创建用户对应的宠物列表
int SQL::createNewPetTable(userClass U)
{
    //创建新表
    char sql[10000];
    int res = 0;
    string uName = (string)U.getName();
    sprintf(sql, "create table if not exists `%s`(\n
        name int(10),level int(10),EXP int(10),specialType int(10))charset=utf8", uName.c_str());
    res = mysql_query(pConn, sql);
    if (res){
        printf("创建失败:%s", mysql_error(pConn));
    }
}

```



```

    }
    //把初始化用户的精灵信息加入表中
    for (int i = 0; i < U.petNum; i++) {
        sprintf(sql, "insert into `%s` (name,level,EXP,specialType) values(%d,%d,%d,%d)",
            uName.c_str(),
            U.pname[i],
            U.level[i],
            U.EXP[i],
            U.ST[i]);
        if (mysql_query(pConn, sql)) {
            printf("新增表项失败:%s", mysql_error(pConn));
            return 0;
        }
    }
    return res;
}

//在数据库插入新用户
int SQL::insertNewUsr(userClass U)
{
    char sql[10000];
    string uName = U.getName();
    string uPwd = U.getPwd();
    //首先判断该用户是否已经存在
    sprintf(sql, "select * from user where name = '%s'", uName.c_str());
    MYSQL_RES *m_res;
    mysql_query(pConn, sql);
    m_res = mysql_store_result(pConn);
    if (mysql_num_rows(m_res) > 0) { //用户存在
        printf("用户存在");
        mysql_free_result(m_res);
        return 0;
    }
    mysql_free_result(m_res);
    //用户不存在,在user表中生成新用户
    freeConnect();
    connectPet();
    sprintf(sql, "insert into user(name,password,petNum,win,game) values('%s','%s',%d,%d,%d)",
        uName.c_str(),
        uPwd.c_str(),
        U.petNum,
        U.getWin(),
        U.getGame());
    if (mysql_query(pConn, sql)) {
        printf("新增用户失败: %s", mysql_error(pConn));
    }
}

```

```

        return 0;
    }

    //新增用户的Pet表
    int res = createNewPetTable(U);
    return res;
}

```

## 2.3.2 客户端（函数实现方式大致一样，以登录和登出为例）

### 2.3.2.1 登录功能

```

//登录按钮
void login::on_loginBtn_clicked()
{
    if(!connectToServer()) return ;
    QString nametmp = ui->usrLine->text().trimmed();
    QString pwdtmp = ui->pwdLine->text();
    QByteArray qbtmp;
    Datagram *datatmp = new Datagram;
    char *ch;
    //const char* chtmp[1024];

    datatmp->command = 1;                //登录请求
    datatmp->res = 0;
    qbtmp = nametmp.toLatin1();          //用户名和密码
    strcpy(datatmp->usrName,qbtmp.data());
    qbtmp = pwdtmp.toLatin1();
    strcpy(datatmp->usrPwd,qbtmp.data());

    ch = datagramToChar(datatmp);
    send(*sclient,ch,1023,0);
    recv(*sclient,ch,1023,0);

    datatmp = charToDatagram(ch);
    if(datatmp->res == 1){                //登录成功，对 myAccount 进行初始化
        QMessageBox::information(NULL,"提示","登录成功! ",QMessageBox::Yes);
        delete myAccount;
        myAccount = datagramToUser(datatmp);
        accept();
    }
    else if(datatmp->res == 2){
        QMessageBox::information(NULL,"提示","该用户已登录! ",QMessageBox::Yes);
    }
}

```

```

        closesocket(*sclient);
        return ;
    }
    else{          //登录失败
        QMessageBox::information(NULL,"提示","登录失败! 请检查用户名和密码",QMessageBox::Yes);
        closesocket(*sclient);
        return ;
    }
}

```

### 2.3.3.2 登出功能的实现（析构函数中）

```

MainWindow::~~MainWindow()
{
    //登出
    Datagram *Dg = UserToDatagram(myAccount);
    Dg->command = 3;
    char *ch = datagramToChar(Dg);
    send(*sclient,ch,1023,0);
    closesocket(*sclient);
    delete Dg;
    delete ui;
}

```

## 3 程序测试

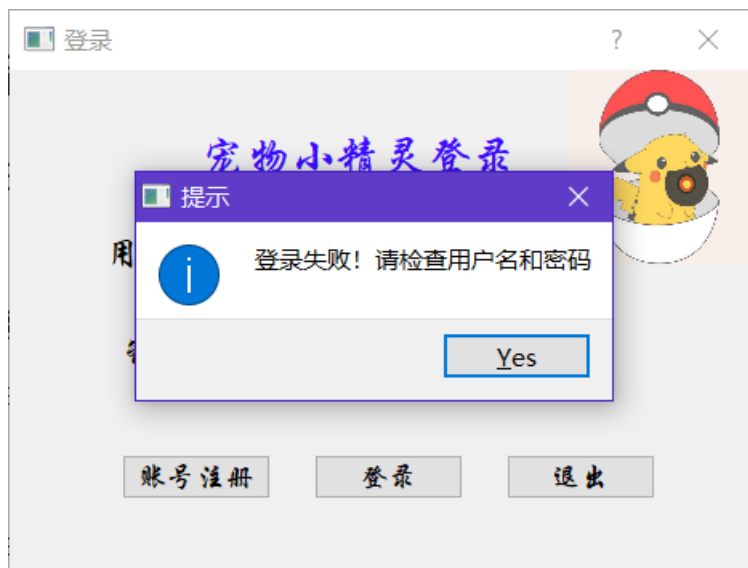
经测试，程序能实现上述功能。

### 3.1 正常登录

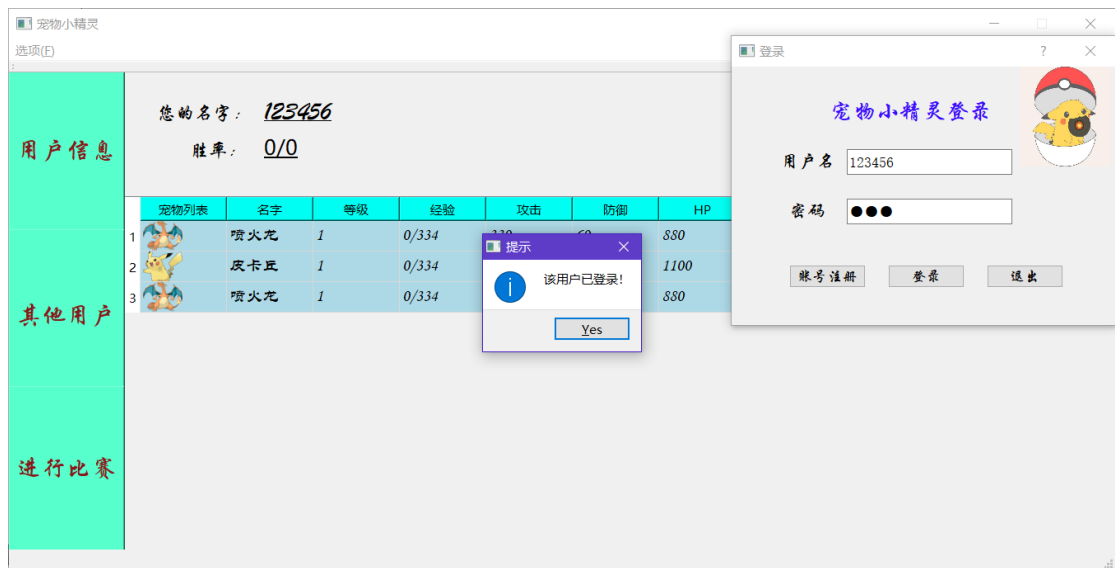


### 3.2 登录的几种异常情况

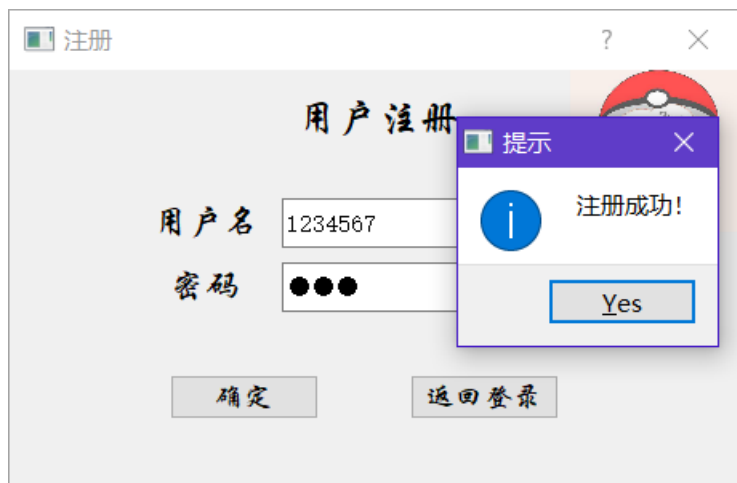
用户名/密码错误



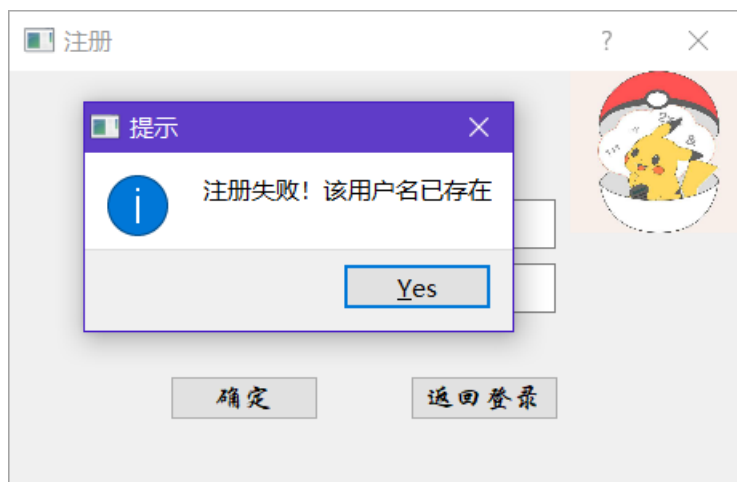
用户已登录



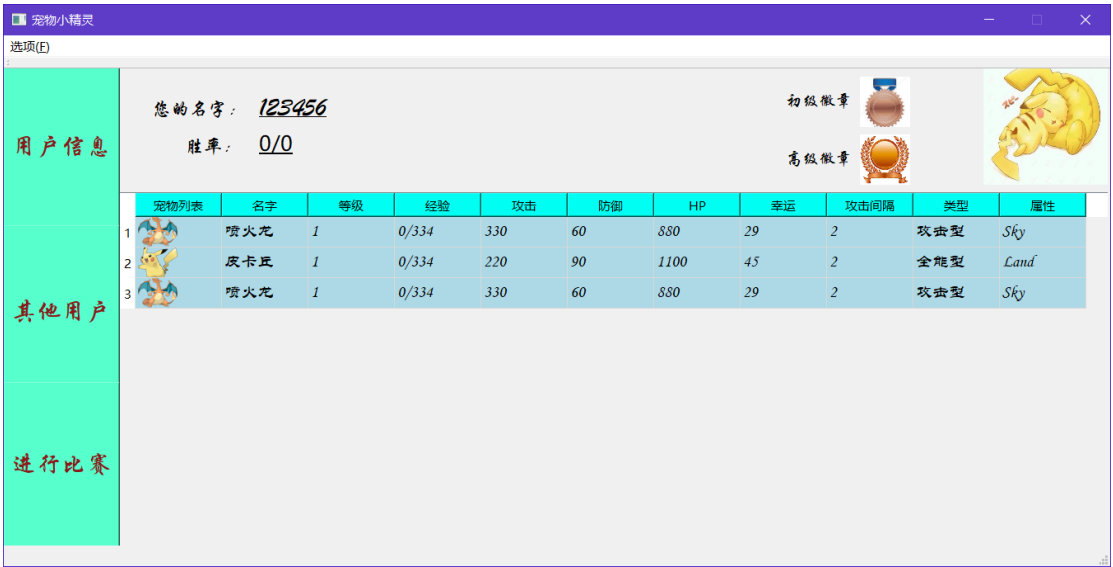
### 3.3 正常注册



### 3.4 注册异常：用户名存在

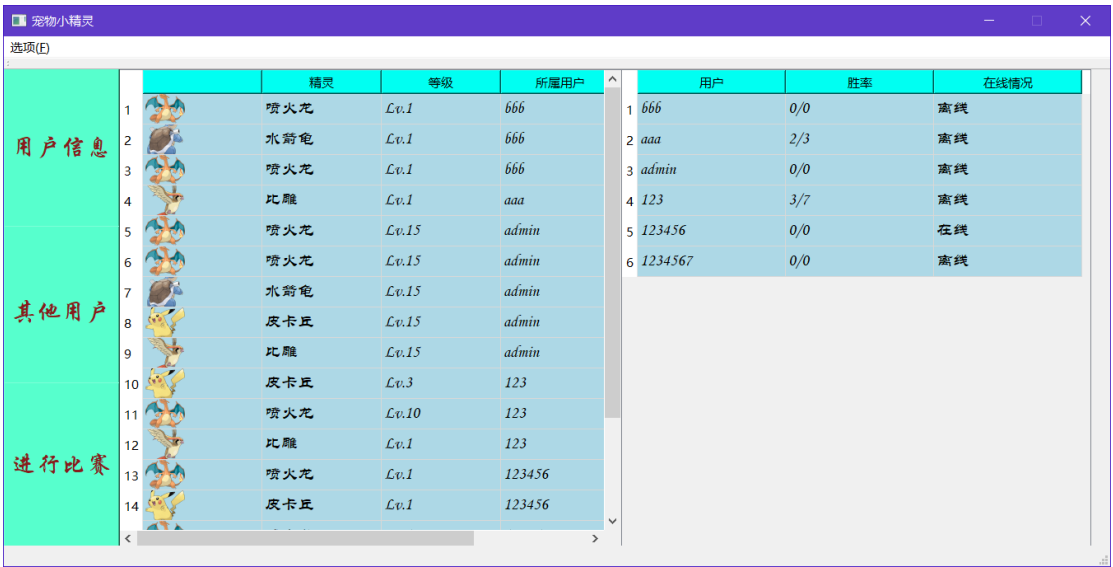


### 3.5 用户信息显示



### 3.6 查看其他用户

包含用户信息、在线情况。



## 三、作业 3 游戏对战的设计

### 1 程序功能

实现游戏对战：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。每打败一只小精灵，对方当前等级下经验最大值的 20%作为自己获得经验。

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。

比赛为回合制，攻击间隔根据宠物的不同类型而定，有暴击、异常状态机制。

用户增加新属性，可以查看用户胜率、徽章。（宠物个数、高级宠物徽章）。

## 2 程序结构及面向对象设计

### 2.1 程序结构

相对于第二版，增加了两个与游戏对战有关的客户请求；数据库的组织方式由为每个用户创建一张精灵信息表，变成只用一张单独的表存所有用户的精灵信息，由用户名+精灵名的方式唯一确定一只精灵，这样能使数据库效率更高。如图所示：

pete.user: 3 总记录 (大约)

name	password	petNum	win	game
123	1	3	0	0
aaa	bbb	3	1	1
admin	aaa	8	11	13

pete.monster: 14 总记录 (大约)

uname	name	level	EXP	specialType
123	20,011	1	0	0
123	29,903	1	0	0
123	2,859	1	0	0
aaa	0	1	0	1
aaa	8,140	1	67	0
aaa	9,919	1	0	1
admin	2,696	15	347	2
admin	26,849	15	0	0
admin	0	15	693	1
admin	7,433	15	0	3
admin	378	15	0	1
admin	48	15	207	2
admin	11,362	15	0	0
admin	9,270	15	0	0

相对于第一版，修改了战斗相关函数，把一部分函数移到战斗界面中，方便进行调用。但是特殊攻击、升级等函数还是在宠物类里的。

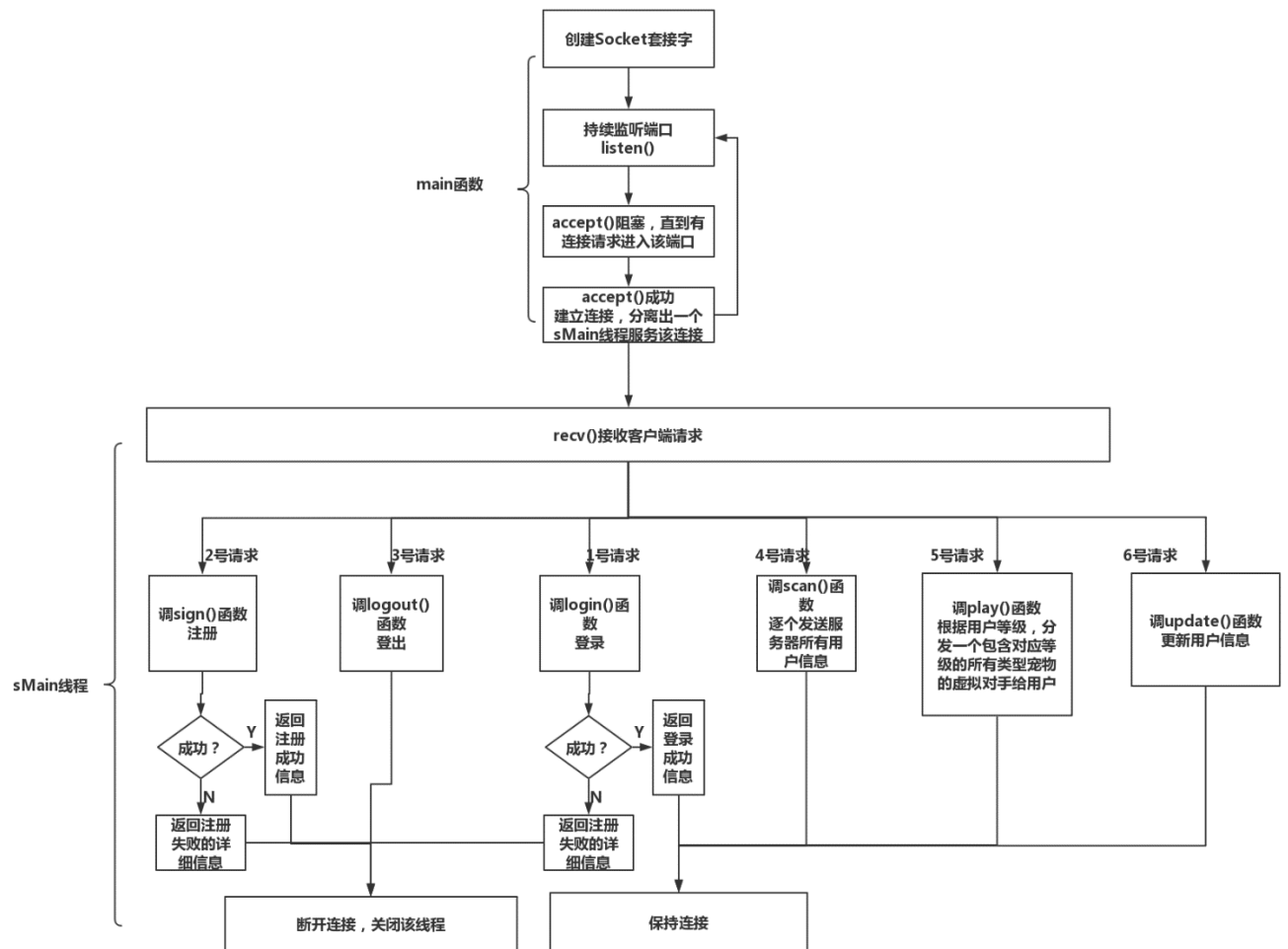
### 2.1.1 服务端

为了更灵活，把与数据库的交互抽象成一个类，如下：



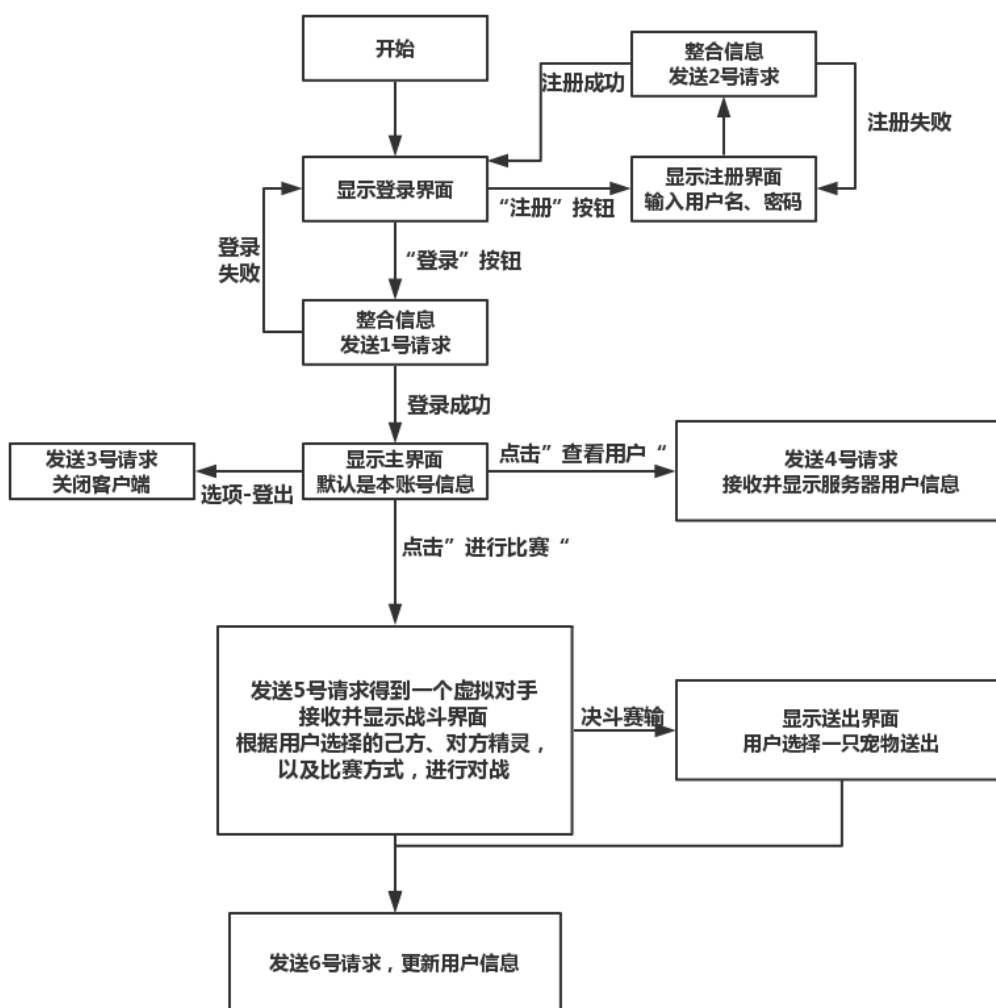
处理请求和与数据库交互的流程如下





## 2.1.2 客户端

处理客户事件的流程如下：



## 2.2 设计

### 2.2.1 服务端

#### 1. 服务用户的对战请求

收到 5 号请求，则根据用户等级，创建一个拥有对应等级的所有种类宠物的虚拟对手，发送给客户端。客户端比赛结束后回送 6 号请求，服务端调用函数更新用户信息。

### 2.2.2 客户端

#### 1. 比赛制

比赛采用回合制，每回合双方分别判断自己攻击间隔条是否走完，若走完则发送攻击。攻击方式由计算机随机，特殊攻击有 CD。

#### 2. 精灵的选择（对战、送出）

用 QComboBox 实现精灵选择。

### 3. 虚拟对手的维护

维护一个全局变量 serverAccount (userClass 类), 里面存有服务器发来的虚拟对手的信息。

## 2.3 重要代码

### 2.3.1 服务端

#### 2.3.1.1 服务用户对战请求

```
void play(SOCKET sClient, Datagram *Dg)
{
    int level_high = 1, level_low = 1;
    //选取用户等级最高和最低的小精灵
    for (int i = 0; i < Dg->petNum; i++) {
        if (level_high < Dg->level[i]) {
            level_high = Dg->level[i];
        }
        else if (level_low > Dg->level[i]) {
            level_low = Dg->level[i];
        }
    }
    int level_choose = (int)(level_high + level_low) / 2;
    //分发一个datagram, 包含虚拟精灵及相关信息
    Datagram *datatmp = new Datagram;
    datatmp->res = 1;
    datatmp->petNum = 4;
    for (int i = 0; i < datatmp->petNum; i++) {
        datatmp->level[i] = userVoid.level[level_choose * 4 + i];
        datatmp->ST[i] = userVoid.ST[level_choose * 4 + i];
    }

    int res = send(sClient, datagramToChar(datatmp), 1023, 0);
    if (res == SOCKET_ERROR) { printf("send failed!5"); }
}
```

#### 2.3.1.2 服务用户更新账户信息请求

```
void update(SOCKET sClient, Datagram *Dg)
{
    userClass *usrtmp = datagramToUser(Dg);
}
```

```

mysqlId.chUsr(*usrtmp);
delete usrtmp;
}

```

## 2.3.2 客户端

### 2.3.2.1 对战主函数

```

void game::on_startBtn_clicked()
{
    if(gameMode == 2){
        QMessageBox::information(NULL,"提示","请选择比赛模式! ");
        return ;
    }
    //开始对战
    srand((unsigned int)time(NULL));
    Round = 0; //回合数
    int res = 0, loseFlag = 0;
    QEventLoop loop;

    while(1){
        Round ++;
        ui->roundLabel->setText("Round"+QString::number(Round));

        //用户精灵
        usrPet->chAttackCD(usrPet->getAttackCD()-1);
        ui->usrCD->setValue(usrPet->getAttackCD()); //设置 CD
        if(usrPet->getAttackCD() == 0){
            usrPet->chAttackCD(usrPet->getAttackInterval());
            res = usrPlay();
        }
        else{
            QString qsusr = "攻击正在 CD 中, CD 剩余: " + QString::number(usrPet->getAttackCD());
            if(usrPet->getDianran() != 0){
                usrPet->chDianranFlag(usrPet->getDianran()-1);
                qsusr += "\n 点燃损失 50HP\n 点燃剩余: " + QString::number(usrPet->getDianran());
                usrPet->chHP(usrPet->getHP()-50);
                if(usrPet->getHP() <= 0){ //被烧死了
                    res = 2;
                }
            }
        }
        ui->gameLabelUsr->setText(qsusr);
    }
}

```

```

//用户结果
int iii = serPet->getHP();
if(serPet->getHP() < 0){           //设置血量
    ui->serProg->setValue(0);
}
else{
    ui->serProg->setValue(serPet->getHP());
}
if(res == 1){
    QMessageBox::information(NULL,"比赛结束","You Win!");
    myAccount->chWin(myAccount->getWin()+1);
    //升级, etc
    int i = 0.2*serPet->getEXPtotal();
    usrPet->chEXP(usrPet->getEXP() + i);
    if(usrPet->getEXP() > usrPet->getEXPtotal()){           //升级
        QMessageBox::information(NULL,"提示","升级! ");
        usrPet->levelUp();
    }
    if(gameMode == 1){           //决斗赛, 获得精灵
        QMessageBox::information(NULL,"提示","恭喜获得精灵! ");
        int petLast = myAccount->petNum;
        myAccount->ST[petLast] = serPet->getST();
        myAccount->level[petLast] = serPet->getLevel();
        myAccount->EXP[petLast] = 0;
        myAccount->pname[petLast] = rand() % 65534;
        myAccount->petNum ++;
    }
    myAccount->level[usrPetOrder] = usrPet->getLevel();
    myAccount->EXP[usrPetOrder] = usrPet->getEXP();
    break ;
}
else if(res == 2){
    QMessageBox::information(NULL,"比赛结束","You Lose!");
    //决斗赛时, 失去一只精灵
    if(gameMode == 1){
        LosePet();
        loseFlag = 1;
    }
    break ;
}
}
QTimer::singleShot(1000, &loop, SLOT(quit()));
loop.exec();

//服务器精灵

```

```

serPet->chAttackCD(serPet->getAttackCD()-1);
ui->serCD->setValue(serPet->getAttackCD());
if(serPet->getAttackCD() == 0){
    serPet->chAttackCD(serPet->getAttackInterval());
    res = serPlay();
}
else{
    QString qsser = "攻击正在 CD 中, CD 剩余: " + QString::number(usrPet->getAttackCD());
    ui->gameLabelServer->setText(qsser);
}
//服务器结果
if(usrPet->getHP() < 0){
    ui->usrProg->setValue(0);
}
else{
    ui->usrProg->setValue(usrPet->getHP());
}

if(res == 2){
    QMessageBox::information(NULL,"比赛结束","You Win!");
    myAccount->chWin(myAccount->getWin()+1);
    //升级, etc
    int i = 0.2*serPet->getEXPtotal();
    usrPet->chEXP(usrPet->getEXP() + i);
    if(usrPet->getEXP() > usrPet->getEXPtotal()){ //升级
        QMessageBox::information(NULL,"提示","升级! ");
        usrPet->levelUp();
    }
    if(gameMode == 1){ //决斗赛, 获得精灵
        QMessageBox::information(NULL,"提示","恭喜获得精灵! ");
        int petLast = myAccount->petNum;
        myAccount->ST[petLast] = serPet->getST();
        myAccount->level[petLast] = serPet->getLevel();
        myAccount->EXP[petLast] = 0;
        myAccount->pname[petLast] = rand() % 65534;
        myAccount->petNum ++;
    }
    myAccount->level[usrPetOrder] = usrPet->getLevel();
    myAccount->EXP[usrPetOrder] = usrPet->getEXP();
    break ;
}
else if(res == 1){
    QMessageBox::information(NULL,"比赛结束","You Lose!");
    //决斗赛时, 失去一只精灵

```

```

        if(gameMode == 1){
            LosePet();
            loseFlag = 1;
        }
        break ;
    }
    QTimer::singleShot(2000, &loop, SLOT(quit()));
    loop.exec();
}

//比赛结束，更新服务器上的账户信息
if(loseFlag == 1){
    return ;
}
myAccount->chGame(myAccount->getGame()+1);
Datagram *Dg = UserToDatagram(myAccount);
Dg->command = 6;
int kkk = send(*sclient,datagramToChar(Dg),1023,0);
if(!kkk){
    QMessageBox::information(NULL,"警告","用户更新发送失败");
}
}
}

```

### 2.3.2.3 战斗伤害结算函数（以用户方为例）

```

/* 返回 1：对方死亡 *
 * 返回 2：己方死亡 *
 * 返回 0：双方存活 */
int game::usrPlay()
{
    QString qstmp;        //输出内容
    srand((unsigned int)time(NULL));
    //首先检查异常状态
    if (usrPet->getMabi() > 0) {
        usrPet->chMabiFlag(usrPet->getMabi()-1);
        qstmp = "哎呀您正在麻痹中\n 不得动弹\n 麻痹剩余" + QString::number(usrPet->getMabi()) + "
回合\n";
        ui->gameLabelUsr->setText(qstmp);
        return 0;
    }
    if (usrPet->getDianran() > 0) {
        usrPet->chDianranFlag(usrPet->getDianran()-1);
        usrPet->chHP(usrPet->getHP()-50);
        if (usrPet->getHP() <= 0) {

```

```

        qstmp = "您被烧死了! ";
        ui->gameLabelUsr->setText(qstmp);
        return 2;
    }
    qstmp = "点燃扣 50HP, 点燃剩余" + QString::number(usrPet->getDianran()) + "回合\n";
}
//开始攻击
int res = 0;
int spJud = rand() % 2;    //一半可能发动特殊攻击, 一半可能发动普通攻击
if(usrPet->getSpCD() != 0 || spJud){    //特殊攻击尚未冷却完成或随机结果为普通攻击
    res = serPet->beAttack(*usrPet);
    qstmp += "您发动了普通攻击! \n";
    usrPet->chSpAttackCD(usrPet->getSpCD()-1);
}
else{    //发动特殊攻击
    res = usrPet->spAttack(*serPet,qstmp);
}
ui->gameLabelUsr->setText(qstmp);
return res;
}

```

### 2.3.2.4 送出精灵-显示精灵（随机 3 只/所有）

```

loseCho::loseCho(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::loseCho)
{
    ui->setupUi(this);

    ui->comboBox->addItem("[空]");
    QString qstmp;
    if(myAccount->petNum <= 3){    //只剩下 3 只
        petNumber = myAccount->petNum;
        for(int i = 0; i < myAccount->petNum; i++){
            switch(myAccount->ST[i]){
                case 0:
                    qstmp = "水箭龟";break;
                case 1:
                    qstmp = "皮卡丘";break;
                case 2:
                    qstmp = "喷火龙";break;
                case 3:
                    qstmp = "比雕";break;
                default:

```



```

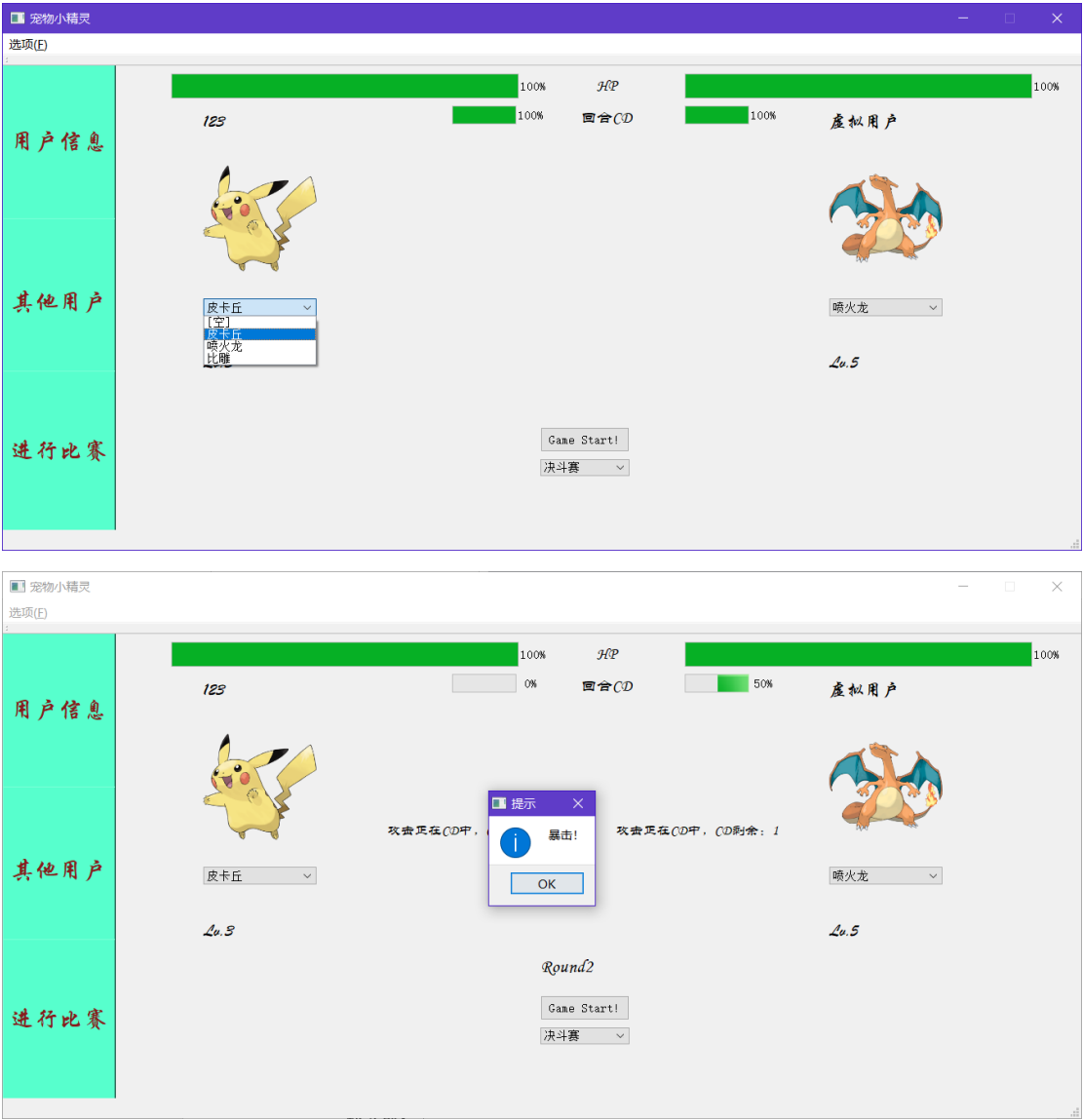
        qstmp = "???";break;
    }
    ui->comboBox->addItem(qstmp);
}
option1 = 0; option2 = 1; option3 = 2;
}
else{
    //剩下超过 3 只
    srand((unsigned int)time(NULL));
    option1 = option2 = option3 = -1;
    petNumber = 3;
    for(int i = 0; i < 3; i++){
        int k = rand() % myAccount->petNum;
        if(option1 == k || option2 == k || option3 == k){
            i--;
            continue;
        }
        else if(option1 == -1){
            option1 = k;
        }
        else if(option2 == -1){
            option2 = k;
        }
        else if(option3 == -1){
            option3 = k;
        }
        switch(myAccount->ST[k]){
            case 0:
                qstmp = "水箭龟";break;
            case 1:
                qstmp = "皮卡丘";break;
            case 2:
                qstmp = "喷火龙";break;
            case 3:
                qstmp = "比雕";break;
            default:
                qstmp = "???";break;
        }
        ui->comboBox->addItem(qstmp);
    }
}
connect(ui->comboBox,SIGNAL(currentIndexChanged(QString)),this,SLOT(printPetImage()));
ui->comboBox->setCurrentIndex(0);
}

```

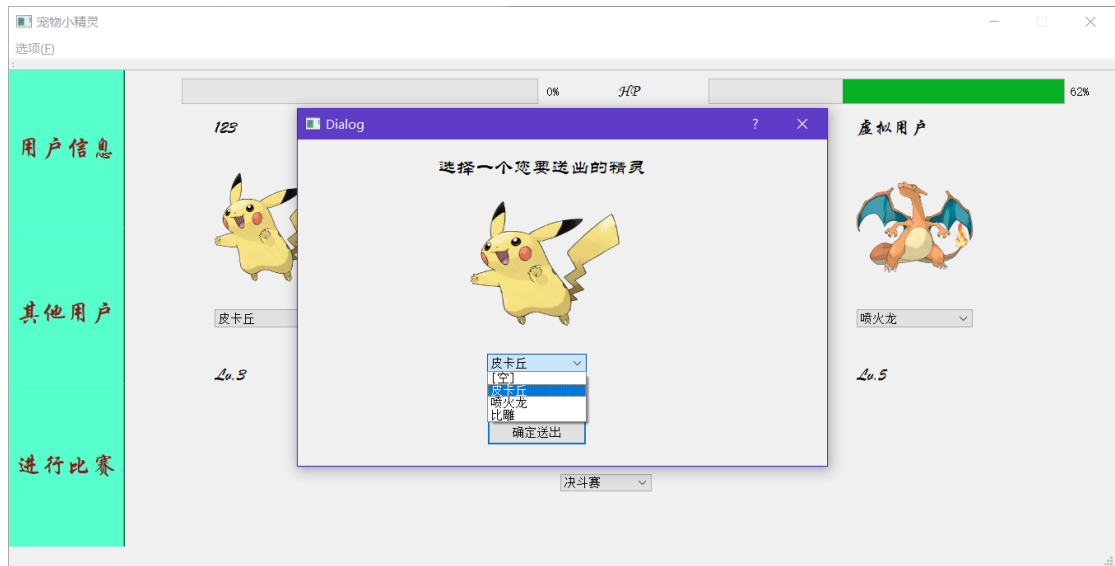
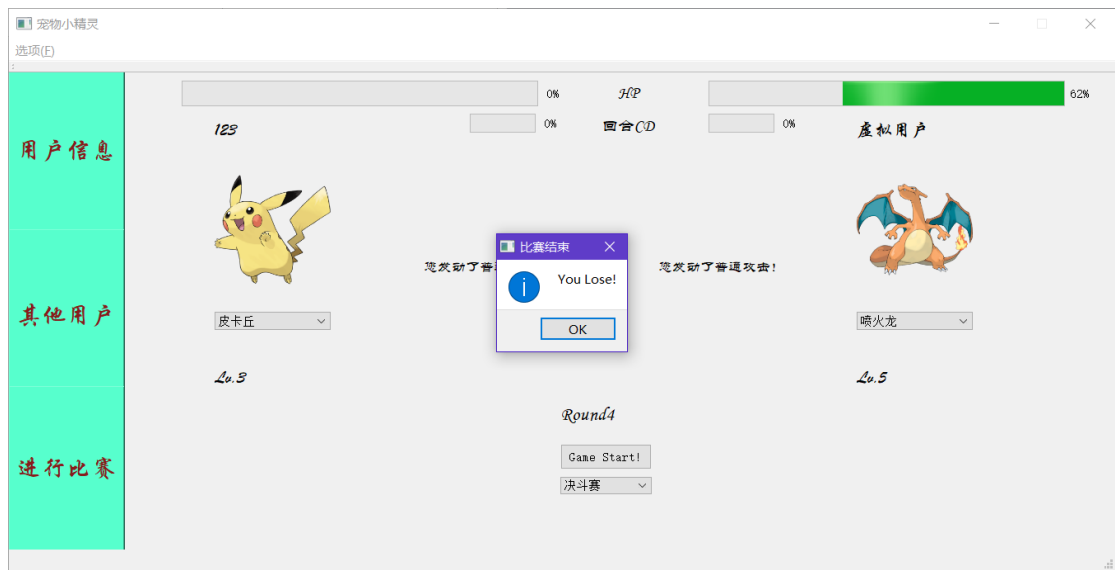
### 3 程序测试

经测试，程序能正常实现上述功能

#### 3.1 对战界面及精灵的选择



#### 3.2 决斗赛败北后精灵的送出



### 3.3 战斗后用户信息的更新

战斗前：

宠物小精灵


选项(E)

用户信息



您的名字: 123  
胜率: 3/7

初级徽章

高级徽章



其他用户

	宠物列表	名字	等级	经验	攻击	防御	HP	幸运	攻击间隔	类型	属性
1		皮卡丘	3	293/534	260	120	1300	55	2	全能型	Land
2		喷火龙	10	147/1234	600	150	1600	254	2	攻击型	Sky
3		比雕	1	32/334	130	26	650	140	1	敏捷型	Sky

进行比赛

宠物小精灵

选项(E)

用户信息

精灵

等级

所属用户

其他用户

	用户	胜率	在线情况
1	bbb	0/0	离线
2	aaa	2/3	离线
3	admin	0/0	离线
4	123456	0/0	离线
5	1234567	0/0	离线
6	123	3/7	在线

进行比赛

战斗后：（决斗赛败北）

宠物小精灵


选项(E)

用户信息



您的名字: 123  
胜率: 3/8

初级徽章

高级徽章

















其他用户

	宠物列表	名字	等级	经验	攻击	防御	HP	幸运	攻击间隔	类型	属性
1		比雕	1	32/334	130	26	650	140	1	敏捷型	Sky
2		喷火龙	10	147/1234	600	150	1600	254	2	攻击型	Sky

进行比赛

宠物小精灵

选项(E)

用户信息	1		喷火龙	Lv.1	666	1	666	0/0	离线
	2		水箭龟	Lv.1	666	2	aaa	2/3	离线
	3		喷火龙	Lv.1	666	3	admin	0/0	离线
	4		比雕	Lv.1	aaa	4	123456	0/0	离线
其他用户	5		喷火龙	Lv.15	admin	5	1234567	0/0	离线
	6		喷火龙	Lv.15	admin	6	123	3/8	在线
	7		水箭龟	Lv.15	admin				
	8		皮卡丘	Lv.15	admin				
	9		比雕	Lv.15	admin				
进行比赛	10		喷火龙	Lv.1	123456				
	11		皮卡丘	Lv.1	123456				
	12		喷火龙	Lv.1	123456				
	13		喷火龙	Lv.1	1234567				
	14		皮卡丘	Lv.1	1234567				