一，宠物小精灵类设计

1.1 宠物小精灵基类

class pet

{

private:

petKind kind;//宠物的属性

int level;//宠物的等级

int xp;//宠物的经验值

int xpMax;//当前级别下宠物的经验值上限

int hp;//宠物的生命值

int attack;//宠物的攻击力

int define;//宠物的防御力

int attackInterval;//宠物的攻击间隔

int missRate;//宠物攻击的失误率

int hpNow;//在战斗过程中宠物的生命值

int attackNow;//在战斗过程中宠物的攻击力

int defineNow;//在战斗过程中宠物的防御力

int attackIntervalNow;//在战斗过程中宠物的攻击间隔

int missRateNow;//当前战斗中宠物的失误率

int attackTime;

public:

pet(petKind \_kind, QString \_name, int \_lv);

pet(QByteArray &b , int &pos);

virtual ~***pet***();

petKind **getKind**() const

{

return kind;

};//获得宠物的属性

int **getLevel**() const

{

return level;

};//获得宠物的等级

QString **getName**() const

{

return name;

};//获得宠物的名称

int **getHp**() const

{

return hp;

};//获得宠物的生命值

int **getXp**() const

{

return xp;

};//获得宠物的经验值

int **getXpMax**() const

{

return xpMax;

};//获得宠物当前等级下经验值的上限

int **getDefine**() const

{

return define;

};//获得宠物的防御力

int **getAttack**() const

{

return attack;

};//获得宠物的攻击力

int **getAttackInterval**() const

{

return attackInterval;

};//获得宠物的攻击间隔

int **getHpNow**() const

{

return hpNow;

};//获得宠物当前生命值

int **getAttackNow**() const

{

return attackNow;

};//获得宠物当前攻击力

int **getDefineNow**() const

{

return defineNow;

};//获得宠物当前防御力

int **getAttackIntervalNow**() const

{

return attackIntervalNow;

};//获得宠物当前攻击间隔

int **getMissRateNow**() const

{

return missRateNow;

};//获得宠物在战斗中的失误率

int **getAttackTime**() const

{

return attackTime;

};

void **changeDefineNow**(int \_def)

{

defineNow = \_def;

};//更改宠物当前的防御力

void **changeAttackNow**(int \_atk)

{

attackNow = \_atk;

};//更改宠物当前的攻击力

void **changeAttackIntervalNow**(int \_atkInterval)

{

attackIntervalNow = \_atkInterval;

};//更改宠物当前的攻击间隔

void **changeHpNow**(int \_hp)

{

hpNow = \_hp;

};//更改宠物当前的生命值

void **levelUp**();//宠物升级

void **levelInitialize**();//宠物级别初始化

void **beAttacked**(QTextBrowser \*p , const pet &attacker);

void **fightStart**();//为战斗开始做准备

void **fightEnd**(int experience);//战斗结束，根据战斗的结果为宠物添加经验值

void **print**(QPainter &p ,int x ,int y);

QString **toQString**();

void **getPetInfo**(QByteArray &b , int &pos);

virtual void ***performAbilities***(QTextBrowser \*p, pet &attacked) = 0;

//特殊攻击，虚函数，用于子类的重写

protected:

QString name;

};

1.2 小精灵派生类

class smallFireDragon: public pet

{

public:

smallFireDragon(petKind , QString , int);

smallFireDragon(QByteArray &b , int &pos);

virtual void ***performAbilities***(QTextBrowser \*p , pet &);//特殊攻击，使用虚方法，对父类进行重载

};

class pikachu: public pet

{

public:

pikachu(petKind , QString , int);

pikachu(QByteArray &b , int &pos);

virtual void ***performAbilities***(QTextBrowser \*p, pet &attacked);

};

class squirtle: public pet

{

public:

squirtle(petKind , QString , int);

squirtle(QByteArray &b , int &pos);

virtual void ***performAbilities***(QTextBrowser \*p , pet &);

};

class bulbasaur: public pet

{

public:

bulbasaur(petKind , QString , int);

bulbasaur(QByteArray &b , int &pos);

virtual void ***performAbilities***(QTextBrowser \*p , pet &);

};

1.3 主界面类

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

bool **fighting**();

void **showInformation**();

bool **hasIllegal**(QString s);

private slots:

void **socket\_disconnected**();//断开通心连接

void **on\_loginIn\_clicked**();//登录按钮

void **on\_registe\_clicked**();//注册按钮

void **readData**();//读取从服务器发来的数据

void **on\_pushButton\_next\_1\_clicked**();

void **on\_pushButton\_next\_2\_clicked**();

void **on\_pushButton\_loginOut\_clicked**();//退出登录按钮

void **on\_pushButton\_lvUpBattle\_clicked**();//升级赛按钮

void **on\_pushButton\_battle\_clicked**();//决斗赛按钮

void **on\_pushButton\_throwName1\_clicked**();//选择丢弃按钮1

void **on\_pushButton\_throwName2\_clicked**();//选择丢弃按钮2

void **on\_pushButton\_throwName3\_clicked**();//选择丢弃按钮3

void **on\_pushButton\_lvUp\_clicked**();//升级按钮

QByteArray **send**(QByteArray &s,QByteArray &name, QByteArray &password)

{

s.append(name);

s.append(' ');

s.append(password);

return s;

}

void **self\_show**(pet &P );

void **oppose\_show**(pet &P);

private:

Ui::MainWindow \*ui;

QTcpSocket \*socket;

QString IP;

int port;

int select1,select2,select3;

int nowPet;

player user;

pet \*p;

};

二.通信协议设计

客户端：

登录时，客户端发送的消息的首字母为l(login缩写)；注册时，客户端发送的消息的首字母为r(register缩写)；登出时，客户端发送的消息的首字母为o(logout缩写)。登录和注册时还包括相应用户的用户名和密码；登出时包括用户登出时所有的状态信息。

服务器端：

登录成功时，服务器端发送“loginsuccessful”；

注册用户名已经存在时，服务器端发送“usernameisin”；

登录失败时，服务器端发送“loginfailed”；

注册成功时，服务器端发送“registersuccessful”；

登录成功时，在上述信息发送之后还应该+空格+当前用户信息+“#”+其他用户的信息，每个用户之间使用“#”分隔

\*用户信息格式：用户名 密码 获胜次数 失败次数 小精灵数量 高级小精灵数量 每个小精灵信息

\*小精灵信息格式：名称 种类 等级 当前经验值

三.小精灵相应数值设计以及计算方法

3.1 小精灵基本属性计算方式：

attack = level \* attackRate\_level ;

define = level \* defineRate\_level ;

hp = level \* hpRate\_level ;

attackInterval = 1000 - level \* intervalRate\_level;

missRate = 10 + int(level/5)+1;

3.2不同种类的小精灵的特殊增强属性：

switch(kind)

{

case ATTACK:

attack =attack\*2;

break;

case DEFINE:

define =define\*2;

break;

case HP:

hp =hp\*2;

break;

default:

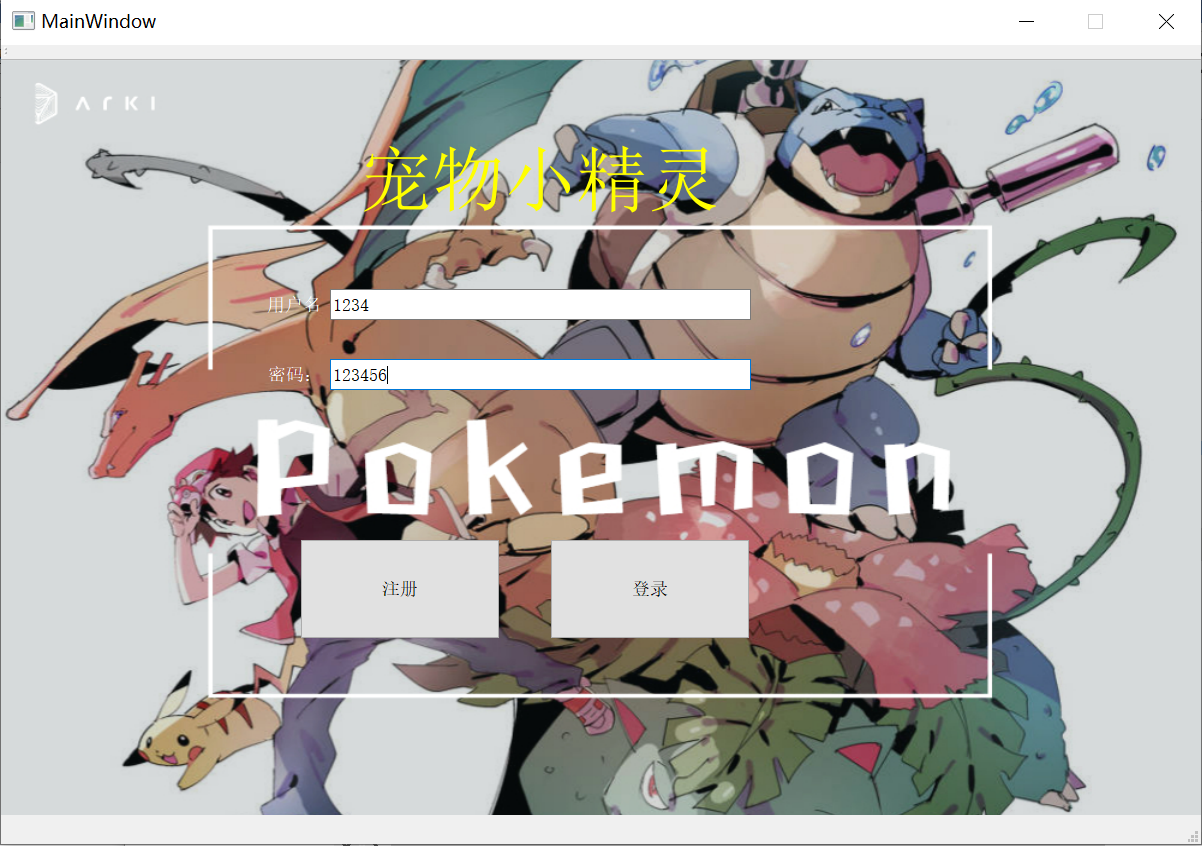
attackInterval = attackInterval - level \* intervalRate\_level;

break;

}

四.运行时截图

登录界面：



主界面：

