C++程序设计实践实验报告

精灵宝可梦图鉴

学院：信息与通信工程学院

班级	姓名	学号
2018211129	骆禹松	2018210071
2018211129	李世超	2018210498

1 项目内容

2 运行环境

3 具体实现方案

3.1 宝可梦类的声明

根据项目所需，宝可梦类定义如下，其中包含了宝可梦属性值、名称、等级、能力加强类型等内容。

```
class Pokemon_Data: public QObject{
public:
    QString name;
    int name_number;
    int level;
    int attack;
    int defense;
    int hp;
    int speed;
    int species;
    int special_attack;
    int special_defense;
    QString GetName();
    QString GetHP();
    QString GetAttack();
    QString GetDefense();
    QString GetSpeed();
    QString GetSpecialAttack();
    QString GetSpecialDenfense();
    QString GetLevel();
    int GetLevelNum();
    Pokemon_Data():level(1){}
    virtual ~Pokemon_Data(){}
    virtual void Level_Up(){}
    void SetData(int Attack, int HP, int Defense, \
    int Speed, int Special_Attack, int Special_Defense);
};
```

3.2 函数功能说明

- GetName()：用于从宝可梦类中获取宝可梦名称，这里通过指针即可获取宝可梦类内对应成员的内容。
- GetLevel()：用于从宝可梦类中获取宝可梦等级，返回类型为字符串。
- GetLevelNum()：用于从宝可梦类中获取宝可梦等级，返回类型为整型。

- GetHP()：用于从宝可梦类中获取宝可梦血量数值。
- GetAttack()：用于从宝可梦类中获取宝可梦攻击数值。
- GetDefense()：用于从宝可梦类中获取宝可梦防御数值。
- GetSpeed()：用于从宝可梦类中获取宝可梦速度数值。
- GetSpecialAttack()：用于从宝可梦类中获取宝可梦特攻数值。
- GetSpecialDefense()：用于从宝可梦类中获取宝可梦特防数值。
- SetData(): 用于设置宝可梦属性值，即给宝可梦类中的变量赋值。
- Level_Up()：宝可梦升级规则设定。
- CreatePokemon()：通过随机数确定宝可梦名称和对应的能力加强类型，然后生成一只宝可梦。
- on_levelupbutton_clicked()：ui中“升级”按钮对应的函数
- on_changepokemonbutton_clicked()：ui中“下一只宝可梦”按钮对应的函数

3.3 获取宝可梦类中对应成员的内容

这里以GetLevelNum()函数为例来说明如何获取宝可梦类中对应成员的内容。

```
//获取宝可梦等级（整型）
int Pokemon_Data::GetLevelNum(){
    return this->level;
}
```

3.4 宝可梦属性值设定

每个宝可梦都有六个属性，分别为血量(HP)、攻击力(Attack)、防御(Defense)、速度(Speed)、特攻(Special Attack)和特防(Special Defence)。由于遗传因素影响，即使是同种宝可梦也存在个体值的差异，可能其中一个能力值会高一些，因此需要定义宝可梦能力加强类型。

```
// 物种包括高攻、高HP、高防、高速、高特攻、高特防六种
enum Species{
    Attack_Plus, HP_Plus, Defense_Plus, Speed_Plus, \
    Special_Attack_Plus, Special_Defense_Plus
};
```

下面以攻击增强型宝可梦为例来说明宝可梦属性值设定规则。首先定义普通宝可梦的初始属性计算公式为：

$$50 + qrand()\%50$$

攻击增强型宝可梦的初始属性计算公式则为:

$$100 + qrand()\%50$$

根据以上公式即可写出攻击增强类型的宝可梦属性设置函数。

```
//攻击加强宝可梦属性设置
AttackPlus::AttackPlus(){
    qsrand((unsigned)time(NULL));
    this->SetData(qrand()%Base_Data+Base_Data*2, qrand()%Base_Data+Base_Data, \
    qrand()%Base_Data+Base_Data, qrand()%Base_Data+Base_Data, \
    qrand()%Base_Data+Base_Data, qrand()%Base_Data+Base_Data);
}
```

其中setData()函数将输入的数值赋给宝可梦类中对应的变量，函数定义如下：

```
//设置宝可梦属性
void Pokemon_Data::SetData(int Attack, int HP, int Defense, \
int Speed, int Special_Attack, int Special_Defense){
    this->attack = Attack;
    this->hp = HP;
    this->defense = Defense;
    this->speed = Speed;
    this->special_attack = Special_Attack;
    this->special_defense = Special_Defense;
}
```

3.5 宝可梦升级规则设定

宝可梦升级后不仅各个属性值会有增长，还能学得更多的技能，先对升级规则规定如下：

- 点击升级按钮可使宝可梦等级+1。
- 宝可梦等级最高为16，达到满级后升级按钮不可点击。
- 宝可梦达到4级、8级和12级时会分别学习一项技能。
- 每升一级，除对应的增强属性+50外，其余属性均+25。

根据上述规则，可以写出宝可梦升级函数，这里依然以攻击增强类型宝可梦为例。

```
//攻击加强宝可梦升级后属性设置
void AttackPlus::Level_Up(){
    this->level++;
    this->attack+=Base_Levelup*2;
    this->hp+=Base_Levelup;
    this->defense+=Base_Levelup;
    this->speed+=Base_Levelup;
    this->special_attack+=Base_Levelup;
    this->special_defense+=Base_Levelup;
}
```

当宝可梦达到满级后，升级按钮不可点击，这里通过"setDisabled(true)"实现。

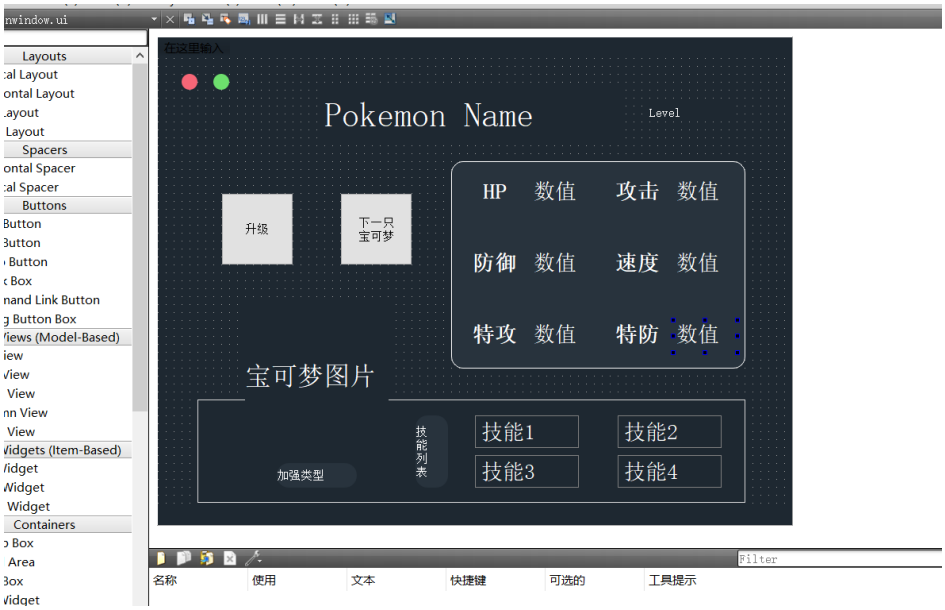
```
//满级16级后设置升级按钮点击无效
    if(pokemon_data->GetLevelNum()>=level_max){
        ui->levelupbutton->setDisabled(true);
        ui->level->setText("已升至满级");
    }
```

根据等级学习技能可以通过if判断语句来实现，其中Skill是QList类型，本次项目共收录了36个宝可梦，故Skill中的元素数量为4*16，其中第$4i$到第$4i+3$个元素分别对应第i个宝可梦的4个技能（这里以0为计数起点）。

```
//每升四级学一个技能
int levelnow = pokemon_data->GetLevelNum();
if (levelnow/4==0){
    ui->skill_1->setText(Skill.at(name_num*4));
    ui->skill_1->setAlignment(Qt::AlignCenter);
}
if (levelnow/4==1){
    ui->skill_2->setText(Skill.at(name_num*4+1));
    ui->skill_2->setAlignment(Qt::AlignCenter);
}
if (levelnow/4==2){
    ui->skill_3->setText(Skill.at(name_num*4+2));
    ui->skill_3->setAlignment(Qt::AlignCenter);
}
if (levelnow/4==3){
    ui->skill_4->setText(Skill.at(name_num*4+3));
    ui->skill_4->setAlignment(Qt::AlignCenter);
}
```

3.6 UI设计

本项目采用Qt来对程序进行可视化，整体UI设计如下图所示：



为了使页面更加美观，我们还对页面进行了如下处理：

- 去掉了程序的顶栏，同时在左上角增加了两个按钮分别实现关闭窗口和最小化窗口的功能。

```
//设置隐藏窗口状态栏、窗口透明度、禁用拖放窗口大小
this->setWindowFlags(Qt::FramelessWindowHint);
this->setFixedSize(this->width(),this->height());

void MainWindow::on_closebutton_clicked()
{
    this->close(); //关闭窗口
}

void MainWindow::on_minibutton_clicked()
{
    this->showMinimized(); //最小化窗口
}
ui->closebutton->setToolTip("关闭");
ui->minibutton->setToolTip("最小化");
```

- 对窗口进行了7%的透明化处理，使背景不再像纯色那样单调。

```
this->setWindowOpacity(0.93);
```

- 鼠标左键按住窗口任意空白位置并移动即可拖动窗口。

```
void MainWindow::mousePressEvent(QMouseEvent *e){
    if(e->buttons()==Qt::LeftButton)
        startRelativePos = e->pos();
}

void MainWindow::mouseMoveEvent(QMouseEvent *e){
    if(e->buttons()==Qt::LeftButton)
        move(e->globalPos()-startRelativePos);
}

void MainWindow::mouseReleaseEvent(QMouseEvent *e){
    e->accept();
}
```

4 项目成果

5 心得体会