C++程序设计课程设计

实习报告模板

班　　级：**10041811**

姓　　名：周子杰

学　　号：**1005183121**

指导教师：夏军宝

提交日期：**2020.1.5**

目　录

1　系统概述

1.1　学生管理系统程序概述（控制台程序）

参照之前的实习任务，我设计了一款学生管理系统。在功能上，为用户提供了

```
1.  cout << " 0) 退出管理系统\n";
2.  cout << " 1) 添加学生信息\n";
3.  cout << " 2) 移除学生信息\n";
4.  cout << " 3) 查看学生信息\n";
5.  cout << " 4) 修改学生信息\n";
6.  cout << " 5) 查看各科学分\n";
7.  cout << " 6) 全部学生信息\n";
8.  cout << " 7) 其余隐藏功能\n";
```

六个功能，其中，第七个功能在维护中（滑稽）。

随着实习任务的不断推进，我的学生管理系统也在不断地完善，从一开始的数组版本（采用结构体）到第二天的动态扩容版本（采用类），几乎就是重新做了一个新的程序。当然，从第一个版本的图书管理系统，结合实际情况，我将其变更为了目前的学生管理系统。这种每一天增量式的更新真的是一种全新的体验，看着自己的程序一点点被完善，有一种老父亲的感觉。就完成度而言，我对这个管理系统还是满意的，它基本上涵盖了一个学生管理系统应有的各种主要功能。

1.2　红灰忍者の生存+对战大冒险程序概述（图形界面程序）

这是我制作的第一个游戏，是在之前的实习任务的基础上进行修改和完善得来的。本质上还是一个小球碰撞类的游戏，不过进行了修改，也对界面进行了一些美化，加入了新的操作方式，以及一些小的细节部分，如开始游戏、记分板、一键重启等等。这是我进入大学以来制作的第一个游戏，也是第一次用去接触这些新的内容--可视化。尽管没那么完美，但对于自己一点一点地去尝试，一点一

点地去完善，最终形成的这个结果，我还是很有成就感的。

2 需求分析

2.1 学生管理系统程序需求分析（控制台程序）

我做制作的学生管理系统共有六个功能

```
1.  cout << " 0) 退出管理系统\n";
2.  cout << " 1) 添加学生信息\n";
3.  cout << " 2) 移除学生信息\n";
4.  cout << " 3) 查看学生信息\n";
5.  cout << " 4) 修改学生信息\n";
6.  cout << " 5) 查看各科学分\n";
7.  cout << " 6) 全部学生信息\n";
```

这些都是一个学生管理系统该有的功能。

同时，对于学生我进行了按专业的分类，也就是在 Student 类下添加了两个子类 StudentsOfComputerScience 类和 StudentOfUndergroundWaterScience 类。亦即两个专业 1）-计算机科学与技术，2）-地下水科学与工程。(其中，地下水科学与工程是我以前的专业)。

对于不同的专业有着不同的学科（都是百分制），对应不同的学分。

基础课程	高等数学	大学英语	大学物理
学分	6	6	4

计算机科学与技术专业课程	C++	汇编语言
学分	3	3

地下水科学与工程专业课程	地球概论	岩石矿物
学分	6	4

每个学生对应拥有的信息是：1）学号 2）姓名 3）专业 4）各科成绩

对应不同的专业有不同的学科，对应不同的学分，由此算出绩点。

同时，支持按学生学号和姓名两种方式查找来移除或查看学生信息。

关于添加学生信息，首先选择学生的专业，由此分支出不同的学科，输入各专业学生的完整信息。

在 6 号功能中为用户提供了选择排序方式的功能，可以依据 1-学号 2-姓名 3-绩点来降序排序。

最后，该系统还拥有自动存储系统和动态扩容的功能，在每次更新学生信息后自动存储。当学生人数超出最大存储人数时，将最大人数翻倍。

2.2 红灰忍者の生存+对战大冒险程序需求分析（图形界面程序）

这是一个双人游戏，玩家将控制小灰小红两个忍者，进行生存大冒险。在城市的夜空中，两个飞行着的忍者，一个通过键盘 wasd 控制上左下右，另一个通过鼠标划线控制方向和速度，城市的夜空中有着许多飞镖，玩家的任务就是躲避这些飞镖超越另一个忍者到达终点。

每位忍者的初始生命是 10，碰到飞镖的忍者会被传送到起始位置，同时生命-1，一旦到达终点或者对方生命小于等于 0 且小于自己的生命，则判定该忍者为优势忍者。随着时间的继续飞镖的速度会越来越快，游戏的难度会成线性的增长，最终到达一个极限速度。另外，离终点越近的飞镖体积越大，飞行速度越快，也就是难度越高。

同时，为了应对可能发生的 bug（虽然目前暂时还没遇到卡死等情况），设计了一键重启，按下 R 键，立即关闭程序，并重新运行程序。

如此，这个游戏应该能够满足绝大多数人对这类游戏的需求。

最后，附上游戏界面：

3 系统设计与关键技术

3.1 学生管理系统程序总体结构设计（控制台程序）

具体的功能实现可以参考 2-系统设计的需求分析。

这里就具体讲一下设计思路及框架。

首先，我设计了三个类，分别是 StudentManage 类、Student 类、StudentsOfComputerScience 类和 StudentOfUndergroundWaterScience 类。其中，StudentsOfComputerScience 类和 StudentOfUndergroundWaterScience 类是 Student 类的两个子类。

其次，关于功能的函数我将其统一放在了 function.h 里面，包括

```
char menu();
void doAddStudent(StudentManage& a);
void doRemoveStudent(StudentManage& a);
void doViewStudent(const StudentManage& a);
void dochangeStudent(StudentManage& a);
void doViewCredit();
void doViewAllStudnets(const StudentManage& a);
double dogetGPA(double s);
```

8 个功能，除了第一个功能是调用菜单界面，别的函数分别对应菜单里对应的功能,并按需调用 StudentManage 类中的不同内容,然后再在 StudentManage 类中再按需求调用 Student 类、StudentsOfComputerScience 类和 StudentOfUndergroundWaterScience 类中的函数，具体实现将在 4-系统实现中具体解释。

同时，在 StudentManage 类，我加入了存储系统。首先，先在各个学生类以及其子类中的 getInfo 函数（虚函数）中，下面以 StudentsOfUndergroundWaterScience 类为例。

```
string StudentOfUndergroundWaterScience::getInfo()const{
    ostringstream ostr;
```

```
    ostr << getMajor() << endl;
    ostr << Student::getInfo();
    ostr << getEarththeoryScore() << " " << getRockformationScore() << endl;

    return ostr.str();
}
```

在 StudentManage 类中完成数据的存储与读取功能。

```
void readData(std::string filename);
void saveData(std::string filename);
```

具体实现为：

```
void StudentManage::saveData(string filename) {
    ofstream out(filename);
    if (out) {
//        out << maxSize << endl;
//        out << size << endl;
        out<<pStudents.size()<<endl;
//        for (int i = 0; i < size; ++i) {
//            out << pStudents[i]->getInfo();
//        }
        for(auto e:pStudents){
            out<<e->getInfo();
        }
    }
}
void StudentManage::readData(string filename) {
    ifstream in(filename);
    if (in) {
        int fileSize;
        in>>fileSize;
        int type;
        string sn;
        string name;
        double mathsScore, englishScore,physicsScore;
        double earththeoryScore,rockformationScore;
        double assemblylanguageScore,cplusplusScore;
        for (int i = 0; i < fileSize; ++i) {
            in >> type;
            getline(in, sn);
            getline(in, name);
```

```
            in >> mathsScore >> englishScore >> physicsScore;
            if (type == 2) {
                in >> earththeoryScore >> rockformationScore;
                addStudent(new
                    StudentOfUndergroundWaterScience(sn, name,
                        mathsScore,englishScore,physicsScore,earththeoryScore,rockformationScore));
            }
            else if (type == 1) {
                in >> assemblylanguageScore >> cplusplusScore;
                addStudent(new
                    StudentsOfComputerScience(sn,name,
                       mathsScore,englishScore,physicsScore,assemblylanguageScore,cplusplusScore));
            }
        }
    }
    sortStudents();
}
```

将数据写入.data 文件或者读取出来。

3.2 红灰忍者の生存+对战大冒险程序总体结构设计（图形界面程序）

这个游戏的数据结构其实很简单，它是课上的实习任务的一个整合+完善。所以也继承了实习任务的结构。

总共有三个类，Ball 类、MainWindow 类和 RightWidget 类（我取消了 leftwidget 类-为了界面的美化我将记分板等放在了游戏中线上）。

在 Ball 类中

```
class Ball
{
public:
    Ball();
    Ball(double xpos,double ypos,double r,double s,double a,
             QColor c);
    void setRectangle(const QRect& rect){
        this->rect=rect;
    }
    void draw(QPainter *p);
```

```
    void drawImage(QPainter *p,QImage b);
    void checkCollision(Ball &b);
    void move(); //在指定的方向上移动 1 步
    void checkBoundary();
    void setX(double X){this->x=X;}
    void setY(double Y){this->y=Y;}
    void setAngle(double angle){this->angle=angle;}
    void setSpeed(double speed){this->speed=speed;}
    double getX(){return x;}
    double getY(){return y;}
    double getAngle(){return angle;}
    bool collision(Ball &b);
    bool isItWin();
private:
    double x,y;  //小球的中心为位置
    double angle;  //小球移动角度
    double radius;  //小球半径
    double speed;  //小球移动速度

    QColor color;   //小球填充颜色
    QRect rect;    //用于检测的窗口矩形区域,这里把它作为了数据成员.将来最好做成传参的形式。
};
```

我对小球的一些基本属性进行了封装，并设置了碰撞之后的反弹等等。

在 MainWindow 中：

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = 0);
    ~MainWindow();
    //定时器常常定义在 mainwindows 中。
    RightWidget* getRightWidget(){return right;}//主窗口类中封装获取右侧窗口的接口(让右侧窗口的指针返回来调用）
    void StopTimer();  //停止定时器的接口,保持 leftwidget 的简洁。
    void resumeTimer();  //重启定时器的接口

protected slots:
    void timeToShow();    //QTimer 中很重要的功能为 start,定时器事件对应的槽方法
```

```
15. private:
16.     //LeftWidget *left;//保存左右窗口的指针
17.     RightWidget *right;
18.     QSplitter *splitter;
19.     QTimer *timer;     //定时器对象
20.
21.     void paintEvent(QPaintEvent* e);
22. };
```

对显示进行了一些设置，包括屏幕背景填充等等，并定义了一个计时器来保持定时更新小球位置。

最重要的就是 RightWidget 类：

```
1.  class MainWindow;
2.  class RightWidget : public QWidget
3.  {
4.      Q_OBJECT
5.  public:
6.      explicit RightWidget(QWidget *parent = 0);
7.      void paintEvent(QPaintEvent *);//重载虚函数
8.      void updateBalls();  //定义移动小球的接口
9.      void addBall(const Ball& b);  //添加小球的接口
10.
11.     void mousePressEvent(QMouseEvent *e);
12.     void mouseMoveEvent(QMouseEvent *e);
13.     void mouseReleaseEvent(QMouseEvent *e);
14.     Ball theBlue;
15.     Ball theGreen;
16.     QList<Ball> balls;//多个小球
17.     bool theBlueWin();
18.     bool theGreenWin();
19.     int BlueLife=10;
20.     int GreenLife=10;
21.
22.     void changeTimeLabel2(int count);
23.     void changeBlueLifeLabel2(int life);
24.     void changeGreenLifeLabel2(int life);
25. protected:
26.     void keyPressEvent(QKeyEvent *e);
27.     void keyReleaseEvent(QKeyEvent *e);
28. private:
29.     QPoint p1,p2;
30.     bool drawsignal;
```

```
31.
32.     MainWindow *pmain;  //指向主窗口的指针
33.     QLabel *timeLabel,*blueLifeLabel,*greenLifeLabel,*theWinnerLabel;//都是指
    针.
34.     QLabel *timeLabel2,*blueLifeLabel2,*greenLifeLabel2,*theWinnerLabel2;
35.     QPushButton *stopButton;//控制暂停和重新启动.
36.
37.     QTimer *time;
38. signals:
39.
40. public slots:
41.     //void repeatPress();
42.     void stopBall();
43.     void timeDisplay();
44. };
```

它实现了控制小球的移动，增加了标签，可以判断胜负并持续调用计时器来刷新小球位置等等功能。

3.3 关键技术概述

在学生管理系统中， 从刚才的 2-系统设计的需求分析也有说到，每个学生都按专业进行分类。所以在 Student 类下设了两个子类，分别是 StudentsOfComputerScience 类和 StudentOfUndergroundWaterScience 类。由于拥有不同的学科对应不同的学分，故绩点的计算规则也不一样。这里我选择了运用多态，也就是虚函数。各个类中共通的函数这样就可以享有不同的实现。

```
1.  virtual ~Student() {}
2.  Student() = default;
3.  virtual void output()const;
4.  virtual double getGPA()const;
5.  virtual int getMajor()const=0;
6.  virtual std::string getInfo()const;
7.  virtual void changeData();
```

关于我制作的游戏的关键技术我认为在于计时器的调用，这真的是一个贯穿始终的东西，我一开始记录生命值等等东西无论怎么改都不会改变数值，知道将他们加入到含计时器的 updateBalls 才改变了数值。

其次是两个特殊的小球，theBlue 和 theGreen,要对他们特殊对待，很多东西不能简单的把他们和别的球放在一起用一个循环完事，因为这个问题我也消耗了很多时间。

然后是关于键盘操作可读性问题，尽量使用 Qt::Key_+大写字母的形式而不是 ASCII 码，这样可以增加可读性。

4 系统实现

4.1 学生管理系统程序的实现（控制台程序）

文件的核心类为 StudentManage 类、Student 类以及有它派生的两个子类 StudentsOfComputerScience 类和 StudentOfUndergroundWaterScience 类。

在这里我就以各个功能的实现为线索来解释我的程序。

1）添加学生信息

首先在 function.h 里面添加 doAddStudent 函数，下面是它的实现：

```
1）  void doAddStudent(StudentManage& a) {
2）      string name,sn;
3）      double mathsScore;
4）      double englishScore;
5）      double physicsScore;
6）      double earththeoryScore;
7）      double rockformationScore;
8）      double assemblylanguageScore;
9）      double cplusplusScore;
10）      int type;
11）      cout<<"选择学生专业(1-计算机科学与技术，2-地下水科学与技术)"<<endl;
12）      cout<<"选择： ";
13）      cin>>type;
14）      fflush(stdin);
15）      cout << "输入学生学号:";
16）      getline(cin,sn);
17）      cout << "输入学生姓名:";
18）      getline(cin, name);
19）      cout << "输入高等数学成绩:";
20）      cin >> mathsScore;
21）      cout << "输入大学英语成绩:";
22）      cin >> englishScore;
23）      cout << "输入大学物理成绩:";
24）      cin >> physicsScore;
25）      if (type == 1) {
26）          cout << "输入汇编语言成绩:";
27）          cin >> assemblylanguageScore;
```

```
28）          cout << "输入 C++成绩:";
29）          cin >> cplusplusScore;
30）          a.addStudent(new
31）              StudentsOfComputerScience(sn,name,
32）                mathsScore,englishScore,physicsScore,assemblylanguageScore,cplusp
    lusScore));
33）      }
34）      else if (type == 2) {
35）          cout << "输入地球概论成绩:";
36）          cin >> earththeoryScore;
37）          cout << "输入岩石矿物成绩:";
38）          cin >> rockformationScore;
39）          a.addStudent(new
40）              StudentOfUndergroundWaterScience(sn,name,
41）                 mathsScore,englishScore,physicsScore,earththeoryScore,rockformat
    ionScore));
42）      }
43） }
```

根据用户选择的专业来录入信息，调用了 StudentManage 类的 addStudent 以及不同专业对应的构造函数。

```
1.  void StudentManage::addStudent(Student* p) {
2.  //    if (size == maxSize) {
3.  //        reAllocMemory();
4.  //    }
5.      Student* pStudent = findStudentBySN(p->getStudentNumber());
6.      if (pStudent != nullptr) {
7.          cout << "学号为" << p->getStudentNumber() << "的学生已经存在！添加失败
    \n";
8.          return;
9.      }
10. //    pStudents[size] = p;
11. //    size++;
12.     pStudents.push_back(p);
13.     sortStudents();
14. }
```

在 addStudent 中先根据学号查找这个学生是否存在。具体实现为：

```
1.  const Student* StudentManage::findStudentBySN(string sn) const {
2.  //    for(auto e : pStudents)
3.  //          if(e->getStudentNumber()==sn)
4.  //             return e;
```

```
//    return nullptr;
//}
//void StudentManage::reAllocMemory() {
//    maxSize *= 2;
//    int i;
//    Student** temp = pStudents;
//    pStudents = new Student * [maxSize];
//    for (i = 0; i < size; ++i)
//        pStudents[i] = temp[i];
//    delete[] temp;
    vector<Student*>::const_iterator it=find_if(pStudents.begin(),
        pStudents.end(),[=](Student* p){return p->getStudentNumber()==sn;});

    if(it!=pStudents.end())
        return *it;
    return nullptr;
}
```

然后再添加并排序。

2）移除学生信息

和添加学生一样，首先在 function.h 里定义 doRemoveStudent 函数，并给出实现：

```
void doRemoveStudent(StudentManage& a) {
    string sn;
    cout<<"请选择查找方式 1)学号 2)姓名: ";
    int type; cin>>type;
    switch(type){
    case 1:
        cout << "输入学号:";
        cin >> sn;
        a.removeStudent(sn);
        cout<<"已删除! ";
        break;
    case 2:
        cout<<"输入姓名:";
        cin>>sn;
        a.removeStudentByName(sn);
        cout<<"已删除! ";
        break;
    default:
        cout<<"无效输入\n";
        break;
    }
```

```
22. }
```

按学号或姓名来进行查找并调用 StudentManage 类中的 reMoveStudent：

```
1.  void StudentManage::removeStudentByName(string name) {
2.      Student *pStudent = findStudentByName(name);
3.      if (pStudent == nullptr) {
4.          cout << "姓名为" << name << "的学生不存在!\n";
5.          return;
6.      }
7.      delete pStudent;
8.      pStudents.erase(getIterator(pStudent));
9.  }
```

删除储存在 Vector 中的该学生信息。

3）查看学生信息

还是在 function.h 中定义函数：doViewStudent：

```
1.  void doViewStudent(const StudentManage& a) {
2.      string sn;
3.      cout<<"请选择查找方式 1)学号 2)姓名: ";
4.      int type; cin>>type;
5.      switch(type){
6.      case 1:
7.          cout << "输入学号:";
8.          cin >> sn;
9.          a.viewStudent(sn);
10.         break;
11.     case 2:
12.         cout<<"输入姓名:";
13.         cin>>sn;
14.         a.viewStudentByName(sn);
15.         break;
16.     default:
17.         cout<<"无效输入\n";
18.         break;
19.     }
20. }
```

调用 StudentManage 类中的 viewStudent：

```
1.  void StudentManage::viewStudent(string sn)const {
2.      const Student* pStudent = findStudentBySN(sn);
3.      if (pStudent == nullptr) {
```

```
        cout << "学号为" << sn << "的学生不存在!\n";
        return;
    }
    pStudent->output();
}
```

然后调用 Student 类对应专业的子类中的输出：
首先是基类：

```
void Student::output()const{
    cout<<"学号:"<<studentNumber<<endl
        <<"姓名:"<<name<<endl;
}
```

然后是子类，这里以 StudentOfUndergroundWaterScience 类为例说明：

```
void StudentOfUndergroundWaterScience::output() const{
    Student::output();
    cout<<"专业:地下科学与工程\n"
        <<"高等数学成绩:"<<getMathsScore()<<endl
        <<"大学英语成绩:"<<getEarththeoryScore()<<endl
        <<"大学物理成绩:"<<getPhysicsCsore()<<endl
        <<"地球概论成绩:"<<earththeoryScore<<endl
        <<"岩石矿物成绩:"<<rockformationScore<<endl
        <<"绩点:"<<getGPA()<<endl;
}
```

4）修改学生信息

老规矩，function.h，声明函数 doChangeStudent，并实现：

```
void dochangeStudent(StudentManage& a){
    cout<<"需要修改信息的学生学号：\n";
    string sn;
    cin>>sn;
    a.changeStudent(sn);
}
```

调用 StudentManage 函数中的 changeStudent，实现如下：

```
void StudentManage::changeStudent(string sn){
    Student* pStudent = findStudentBySN(sn);
    if (pStudent == nullptr) {
        cout << "学号为" << sn << "的学生不存在!\n";
        return;
```

```
    }
    pStudent->changeData();
}
```

之后对应专业调用相应的 changeData 函数，这里还是以 StudentOfUndergroundWaterScience 类为例说明：

```
void StudentOfUndergroundWaterScience::changeData(){
    while(true){
        Student::changeData();
        cout<<"6)地球概论成绩"<<endl
            <<"7)岩石矿物成绩"<<endl;
        cout<<"选择:";
        char choice;
        cin>>choice;
        if(choice=='0')
            break;
        switch(choice){
        case '1':{
            string sn;
            cin>>sn;
            Student::setStudentNumber(sn);
            cout<<"修改成功\n";
            break;
        }
        case '2':{
            string n;
            cin>>n;
            Student::setName(n);
            cout<<"修改成功\n";
            break;
        }
        case '3':{
            double s;
            cin>>s;
            Student::setMathsScore(s);
            cout<<"修改成功\n";
            break;
        }
        case '4':{
            double s;
            cin>>s;
            Student::setEnglishScore(s);
            cout<<"修改成功\n";
```

```
            break;
        }
        case '5':{
            double s;
            cin>>s;
            Student::setPhysicsCsore(s);
            cout<<"修改成功\n";
            break;
        }
        case '6':{
            double s;
            cin>>s;
            setEarththeoryScore(s);
            cout<<"修改成功\n";
            break;
        }
        case '7':{
            double s;
            cin>>s;
            setRockformationScore(s);
            cout<<"修改成功\n";
            break;
        }
        default:{
            cout<<"无效输入 重新输入\n";
            break;
        }
        }
    }
}
```

选择要修改的数据进行修改。

5）查看各科学分

这个很简单，简单的输出就行了。

```
void doViewCredit(){
    cout<<"请选择专业：\n"
        <<"1) 计算机科学与技术\n"
        <<"2) 地下水科学与工程\n";
    char choice;
    cin>>choice;
    switch(choice){
        case '1':
            cout<<"课程\t\t"<<"学分\t\n"
```

```
                <<"高等数学\t"<<"6\t\n"
                <<"大学英语\t"<<"6\t\n"
                <<"大学物理\t"<<"4\t\n"
                <<"C++\t\t"<<"3\t\n"
                <<"汇编语言\t"<<"3\t\n";
            break;
        case '2':
            cout<<"课程\t\t"<<"学分\t\n"
                <<"高等数学\t"<<"6\t\n"
                <<"大学英语\t"<<"6\t\n"
                <<"大学物理\t"<<"4\t\n"
                <<"地球概论\t"<<"6\t\n"
                <<"岩石矿物\t"<<"4\t\n";
            break;
         default :
            cout<<"无效输入\n";
            break;
    }
}
```

6）全部学生信息

首先，function.h，函数声明并实现：

```
void doViewAllStudnets(const StudentManage& a) {
    a.viewAllStudents();
}
```

调用 StudentManage 中的 viewAllStudent 函数：

```
void StudentManage::viewAllStudents()const {
    cout << "学生人数:" << pStudents.size() << endl;
    if(pStudents.size()==0)
        return;
    cout<<"指定排序方式（1-学号，2-姓名，3-绩点）：";
    int type;
    cin>>type;
    const_cast<StudentManage*>(this)->sortCommoditiesByType(type);
//    for (int i = 0; i < size; ++i)
//        pStudents[i]->output();
    for(auto e:pStudents)
        e->output();
}
```

可以选择排序方式，实现为：

```
void StudentManage::sortCommoditiesByType(int type){
    if(type==sortType)
        return;
    sortType=type;
    sortStudents();
}
```

调用的 sortStudent 为：

```
void StudentManage::sortStudents(){
    switch (sortType) {
    case 1://学号
        sort(pStudents.begin(),pStudents.end(),
             [=](Student* p1,Student* p2){
            return p1->getStudentNumber()<p2->getStudentNumber();
        });
        break;
    case 2://姓名
        sort(pStudents.begin(),pStudents.end(),
             [=](Student* p1,Student* p2){
            return p1->getName()<p2->getName();
        });
        break;
    case 3://绩点
        sort(pStudents.begin(),pStudents.end(),
             [=](Student* p1,Student* p2){
            return p1->getGPA()<p2->getGPA();
        });
        break;
    }
}
```

另外 GPA 的计算方式是根据专业的不同而不同的，只是一个简单的计算函数（要用虚函数）在，在此就不展示了。

4.2 红灰忍者の生存+对战大冒险程序的实现（图形界面程序）

在 Ball 中构造小球，分别有着横坐标、纵坐标、半径、速度、角度和颜色等等私有成员。

```
Ball::Ball():Ball(40,100,10,10,45,Qt::red)
{   //通过委托构造实现缺省构造函数
}
Ball::Ball(double xpos,double ypos,double r,double s,double a,QColor c)
```

```
5.      :x(xpos),y(ypos),radius(r),speed(s),angle(a),color(c)
6.  {
7.  }
```

首先，实现小球的移动：

```
1.  void Ball::move(){
2.      double dx,dy;
3.      const double PI=3.14159;
4.      dx=speed*std::sin(angle*PI/180);
5.      dy=speed*std::cos(angle*PI/180);
6.      x+=dx;
7.      y+=dy;
8.      checkBoundary();
9.      speed*=1.0002;
10.     if(speed>=10)
11.         speed=10;
12.     if(speed<0.1)
13.         speed=0;
14. }
15. void Ball::checkBoundary(){ //左上角（0,0）向右下递增。
16.     if(y+radius>rect.height()){ //超出底边界
17.         angle=180-angle;
18.         y=rect.height()-radius; //拉回来
19.     }
20.     if(y-radius<0){
21.         angle=180-angle;
22.         y=radius;//拉到相切的位置
23.     }
24.     if(x-radius<0){
25.         angle=-angle;
26.         x=radius;
27.     }
28. }
29. void Ball::checkCollision(Ball &b){
30.     double dx,dy;
31.     dx=x-b.x;
32.     dy=y-b.y;
33.     double dis=std::sqrt(dx*dx+dy*dy);
34.     if(dis<=radius+b.radius){ //碰撞后只是交换速度和角度
35. //        speed*=1.001;
36. //        b.speed*=1.001;
37. //        if(speed<0.1)
38. //            speed=0;
```

```
//        if(b.speed<0.1)
//            b.speed=0;
        if(fabs(speed-b.speed)<0.1 && fabs(angle-b.angle)<20){
            x=b.x+b.radius+radius;
            y=b.y+b.radius+radius;
        }
        else{
            double temp;
            temp=speed;
            speed=b.speed;
            b.speed=temp;
            temp=angle;
            angle=b.angle;
            b.angle=temp;
        }
    }
}
bool Ball::collision(Ball &b){
    double dx,dy;
    dx=x-b.x;
    dy=y-b.y;
    double dis=std::sqrt(dx*dx+dy*dy);
    if(dis<=radius+b.radius)
        return 1;
    else return 0;
}
bool Ball::isItWin(){
    if(x+radius>rect.width())
        return 1;
     else return 0;
}
```

实现小球的移动（不断加速，直到达到最大速度），以及碰撞处理（包括碰到别的小球和边框），其中将最右面的边框记为终点，一旦冲过就判断为一方优势。

接着，是将小球设置图片填充（前期抠图和调整图片大小像素等等在这就略过细节了）：

```
void Ball::draw(QPainter *p)
{
    QPen pen(color,1,Qt::SolidLine);
    QBrush brush(color);
```

```
5.      p->setPen(pen);
6.      p->setBrush(brush);
7.      QRect r(x-radius,y-radius,radius*2,radius*2);//定义坐标,左上角坐标+矩形长
    宽
8.      p->drawEllipse(r);
9.  }
10. void Ball::drawImage(QPainter *p,QImage b){
11.     QRect target(x-radius,y-radius,radius*2,radius*2);
12.     QRect source(0,0,200,200);
13.     p->drawImage(target, b, source);
14. }
```

其次是 MainWindow 的实现：

```
1.  MainWindow::MainWindow(QWidget *parent)
2.      : QMainWindow(parent)
3.  {
4.      setWindowTitle("红灰忍者の生存+对战大冒险（小红请用 wasd 控制 小灰请用鼠标控
    制 r 键重启）");//标题
5.      splitter=new QSplitter(Qt::Horizontal,this);  //水平切分,水平分割，将来可以
    嵌套分割。
6.      right=new RightWidget(this);
7.
8.      splitter->addWidget(right);
9.      splitter->setStyleSheet("QSplitter::handle { background-color: gray }");
     //setStyleSheet,类似网页样式表，不深究。把分割器背景颜色设置为灰色.
10.     splitter->setHandleWidth(10);  //设置分割条的大小和样式
11.
12.     this->setCentralWidget(splitter);
13.
14.     timer=new QTimer(this);
15.     connect(timer,SIGNAL(timeout()),this,SLOT(timeToShow()));//绑定定时器中的
    timeout 事件到槽方法 timeToShow()
16.     //time->setInterval(1000);
17.     timer->stop();
18. }
19.
20. MainWindow::~MainWindow()
21. {
22.
23. }
24.
25. void MainWindow::timeToShow(){//处理定时器
26.     right->updateBalls();//调用 rightwidget 中定义的更新球。各自有各自的职责。
```

```
}
void MainWindow::StopTimer(){
    timer->stop();
}
void MainWindow::resumeTimer(){
    timer->start(10);
}
void MainWindow::paintEvent(QPaintEvent* e){
    QPixmap  pixmap = QPixmap("background.jpg").scaled(this->size());//pixmap 对象
    QPainter painter(this);
    painter.drawPixmap(this->rect(), pixmap);//绘制
}
```

在这个界面最主要的还是计时器，我设置的是一秒 100 贞刷新率。接着是标题等等的设置，计时器的连接等等。

最后，也是最重要的，就是 RightWidget 的实现了：

```
#include "rightwidget.h"
#include <cmath>
#include <QPainter>
#include <QKeyEvent>
#include <QGridLayout>
#include <QProcess>
#include <QApplication>
#include "paintlabel.h"
#include "leftwidget.h"
#include "mainwindow.h"
RightWidget::RightWidget(QWidget *parent) : QWidget(parent)
{
    setMinimumSize(1700,400);//初始化
    this->grabKeyboard();
    balls.clear();
    theBlue=Ball(50,100,20,0,0,Qt::blue);
    theGreen=Ball(50,300,20,0,0,Qt::green);

    addBall (Ball(100,200,20,1,0,Qt::red));
    addBall (Ball(250,200,22,2,180,Qt::red));
    addBall (Ball(400,200,24,3,0,Qt::red));
    addBall (Ball(550,200,26,4,180,Qt::red));
    addBall (Ball(700,200,28,5,0,Qt::red));
    addBall (Ball(850,200,30,6,180,Qt::red));
    addBall (Ball(1000,200,32,7,0,Qt::red));
```

```
26.     addBall (Ball(1150,200,34,7.5,180,Qt::red));
27.     addBall (Ball(1300,200,36,8.0,0,Qt::red));
28.     addBall (Ball(1450,200,38,8.5,180,Qt::red));
29.     addBall (Ball(1600,200,40,9.0,0,Qt::red));
30.
31. //    QPainter painter(this);
32. //    drawImage
33.
34.     time=new QTimer(this);//运行时间
35.     connect(time,SIGNAL(timeout()),this,SLOT(timeDisplay()));
36.     time->start(1000);
37.
38.     pmain=(MainWindow *)parent;//各个标签
39.     timeLabel=new QLabel(this);
40.     timeLabel->setText(tr("运行时长:"));
41.     timeLabel2=new QLabel(this);
42.     timeLabel2->setText("0");
43.     blueLifeLabel=new QLabel(this);
44.     blueLifeLabel->setText(tr("小红生命:"));
45.     blueLifeLabel2=new QLabel(this);
46.     blueLifeLabel2->setText(QString::number(BlueLife));
47.     greenLifeLabel=new QLabel(this);
48.     greenLifeLabel->setText(tr("小灰生命:"));
49.     greenLifeLabel2=new QLabel(this);
50.     greenLifeLabel2->setText(QString::number(GreenLife));
51.     theWinnerLabel=new QLabel(this);
52.     theWinnerLabel->setText(tr("对战状态： "));
53.     theWinnerLabel2=new QLabel(this);
54.     theWinnerLabel2->setText(tr("对战中..."));
55.     stopButton=new QPushButton(tr("开始游戏"),this);
56.
57.     QFont font;//调字体+颜色
58.     font.setPointSize(14);
59.     font.setFamily("方正舒体");
60.     timeLabel->setFont(font);
61.     blueLifeLabel->setFont(font);
62.     greenLifeLabel->setFont(font);
63.     theWinnerLabel->setFont(font);
64.     timeLabel2->setFont(font);
65.     blueLifeLabel2->setFont(font);
66.     greenLifeLabel2->setFont(font);
67.     theWinnerLabel2->setFont(font);
68.     timeLabel->setStyleSheet("color:red;");
69.     blueLifeLabel->setStyleSheet("color:red;");
```

```
70.     greenLifeLabel->setStyleSheet("color:red;");
71.     theWinnerLabel->setStyleSheet("color:red;");
72.     timeLabel2->setStyleSheet("color:red;");
73.     blueLifeLabel2->setStyleSheet("color:red;");
74.     greenLifeLabel2->setStyleSheet("color:red;");
75.     theWinnerLabel2->setStyleSheet("color:red;");
76.
77.     QGridLayout *mainLayout=new QGridLayout(this);//各标签位置
78.     mainLayout->addWidget(timeLabel,0,0);
79.     mainLayout->addWidget(timeLabel2,0,1);
80.     mainLayout->addWidget(blueLifeLabel,0,2);
81.     mainLayout->addWidget(blueLifeLabel2,0,3);
82.     mainLayout->addWidget(greenLifeLabel,0,4);
83.     mainLayout->addWidget(greenLifeLabel2,0,5);
84.     mainLayout->addWidget(theWinnerLabel,0,6);
85.     mainLayout->addWidget(theWinnerLabel2,0,7);
86.     mainLayout->addWidget(stopButton,8,8);
87.     setLayout(mainLayout);
88.
89.     connect(stopButton,SIGNAL(clicked()),this,SLOT(stopBall()));
90.
91. }
92.
93. void RightWidget::paintEvent(QPaintEvent *)//橡皮线等等的绘制
94. {
95.     QPainter p(this);
96.     QImage red("red.png");
97.     QImage green("green.png");
98.     //theBlue.draw(&p);
99.     //theGreen.draw(&p);
100.      theBlue.drawImage(&p,red);
101.      theGreen.drawImage(&p,green);
102.      //QImage image("beibiao.png");
103.      for(auto& b:balls){
104.          //b.draw(&p);
105.          QImage d("feibiao.png");
106.          b.drawImage(&p,d);
107.
108.      }
109.      QPen pen(Qt::black,2,Qt::DashDotLine);
110.      p.setPen(pen);
111.      if(drawsignal)p.drawLine(p1,p2);
112.
113. }
```

```
114.
115. void RightWidget::updateBalls()//刷新位置并作出判断
116. {
117.     theBlue.setRectangle(this->geometry());
118.     theBlue.move();
119.     theGreen.setRectangle(this->geometry());
120.     theGreen.move();
121.     for(auto &b:balls){
122.             b.setRectangle(this->geometry());
123.             b.move();
124.     }
125.     int j;
126.     for(j=0;j<balls.size();++j)
127.         if(theBlue.collision(balls[j])==1){
128.             //theBlue.checkCollision(balls[j]);
129.             theBlue=Ball(50,100,20,0,0,Qt::blue);
130.             BlueLife--;
131.             changeBlueLifeLabel2(BlueLife);
132.             }
133.     for(j=0;j<balls.size();++j)
134.         if(theGreen.collision(balls[j])==1){
135.             //theBlue.checkCollision(balls[j]);
136.             theGreen=Ball(50,300,20,0,0,Qt::green);
137.             GreenLife--;
138.             changeGreenLifeLabel2(GreenLife);
139.             }
140.     theBlue.checkCollision(theGreen);
141.     if(theGreenWin()||(BlueLife<=0&&BlueLife<GreenLife)){
142.         theWinnerLabel2->setText(tr("小灰の优势！"));
143.     }
144.     else if(theBlueWin()||(GreenLife<=0&&BlueLife>GreenLife)){
145.         theWinnerLabel2->setText(tr("小红の优势！"));
146.     }
147.     else theWinnerLabel2->setText(tr("对战中..."));
148.     update(); //更新窗口显示，重绘小球
149. }
150.
151. void RightWidget::addBall(const Ball &b){//加小球
152.     balls.append(b);
153. }
154.
155. void RightWidget::mousePressEvent(QMouseEvent *e){//鼠标操作
156.     p1.setX(int(theGreen.getX()));
157.     p1.setY(int(theGreen.getY()));
```

```
158. }
159. void RightWidget::mouseMoveEvent(QMouseEvent *e){
160.     p2.setX(e->x());
161.     p2.setY(e->y());
162.     drawsignal=1;
163. }
164. void RightWidget::mouseReleaseEvent(QMouseEvent *e){
165.     p2.setX(e->x());
166.     p2.setY(e->y());
167.     double disx,disy;
168.     disx=p2.x()-p1.x();
169.     disy=p2.y()-p1.y();
170.     double angle,speed;
171.     double PI=3.1415;
172.     if(disy<0&&disx>0)
173.         angle=-(90-atan2(-disy,disx)*180/PI);
174.     else if(disy<0&&disx<0)
175.         angle=-(270+atan2(-disy,-disx)*180/PI);
176.     else if(disy>0&&disx<0)
177.         angle=-(270-atan2(disy,-disx)*180/PI);
178.     else
179.         angle=-(90+atan2(disy,disx)*180/PI);
180.     double dis;
181.     dis=sqrt(disx*disx+disy*disy);
182.     if(dis<10) dis=0;
183.     speed=0.02*(dis);
184.     if(speed>5) speed=5;
185.     theGreen=Ball(p1.x(),p1.y(),20,speed,180+angle,Qt::green);
186.     drawsignal=0;
187. }
188.
189. void RightWidget::keyPressEvent(QKeyEvent *e){//键盘操作
190.
191.     if(e->key()==Qt::Key_W){     //上 w 180 87
192.         theBlue.setSpeed(4);
193.         theBlue.setAngle(180);
194.     }
195.     if(e->key()==Qt::Key_A){     //左 a -90 65
196.         theBlue.setSpeed(4);
197.         theBlue.setAngle(-90);
198.     }
199.     if(e->key()==Qt::Key_D){     //右 d 90 68
200.         theBlue.setSpeed(4);
201.         theBlue.setAngle(90);
```

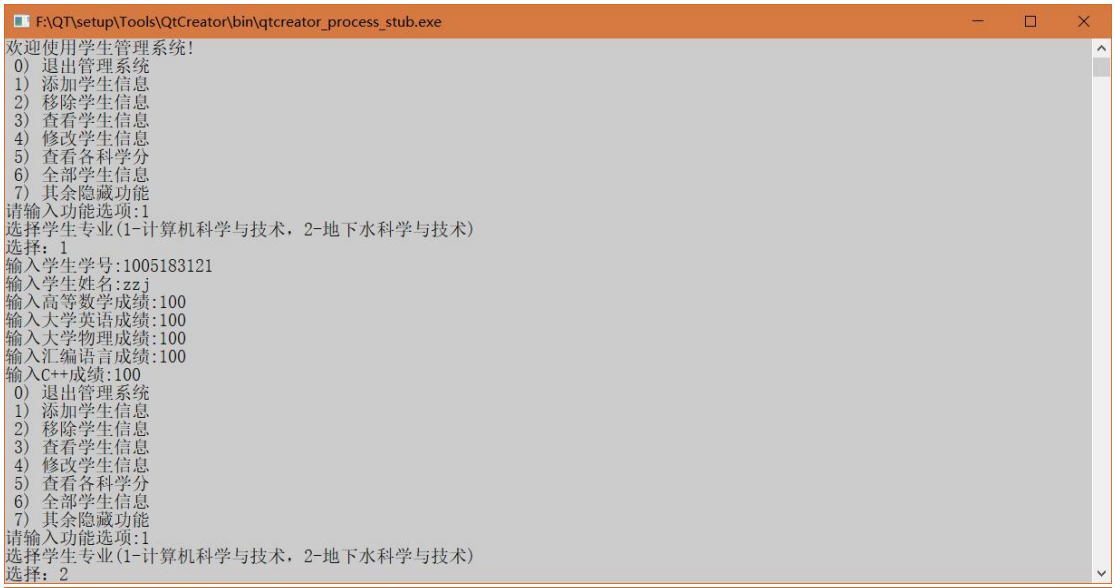
```
202.     }
203.     if(e->key()==Qt::Key_S){     //下 s 0 83
204.         theBlue.setSpeed(4);
205.         theBlue.setAngle(0);
206.     }
207.     if(e->key()==Qt::Key_R){
208.         qApp->closeAllWindows();
209.         QProcess::startDetached(QApplication::applicationFilePath());
210.     }
211. }
212. void RightWidget::keyReleaseEvent(QKeyEvent *e){
213.     if(e->key()==Qt::Key_W){     //上 w 180
214.         theBlue.setSpeed(0);
215.     }
216.     if(e->key()==Qt::Key_A){     //左 a -90
217.         theBlue.setSpeed(0);
218.     }
219.     if(e->key()==Qt::Key_D){     //右 d 90
220.         theBlue.setSpeed(0);
221.     }
222.     if(e->key()==Qt::Key_S){     //下 s 0
223.         theBlue.setSpeed(0);
224.     }
225. }
226.
227. bool RightWidget::theBlueWin(){//判断输赢
228.     if(theBlue.isItWin())
229.         return 1;
230.     else return 0;
231. }
232. bool RightWidget::theGreenWin(){
233.     if(theGreen.isItWin())
234.         return 1;
235.     else return 0;
236. }
237.
238. void RightWidget::changeTimeLabel2(int count){//改变标签值
239.     timeLabel2->setText(QString::number(count));
240. }
241. void RightWidget::changeBlueLifeLabel2(int life){
242.     blueLifeLabel2->setText(QString::number(life));
243. }
244. void RightWidget::changeGreenLifeLabel2(int life){
245.     greenLifeLabel2->setText(QString::number(life));
```

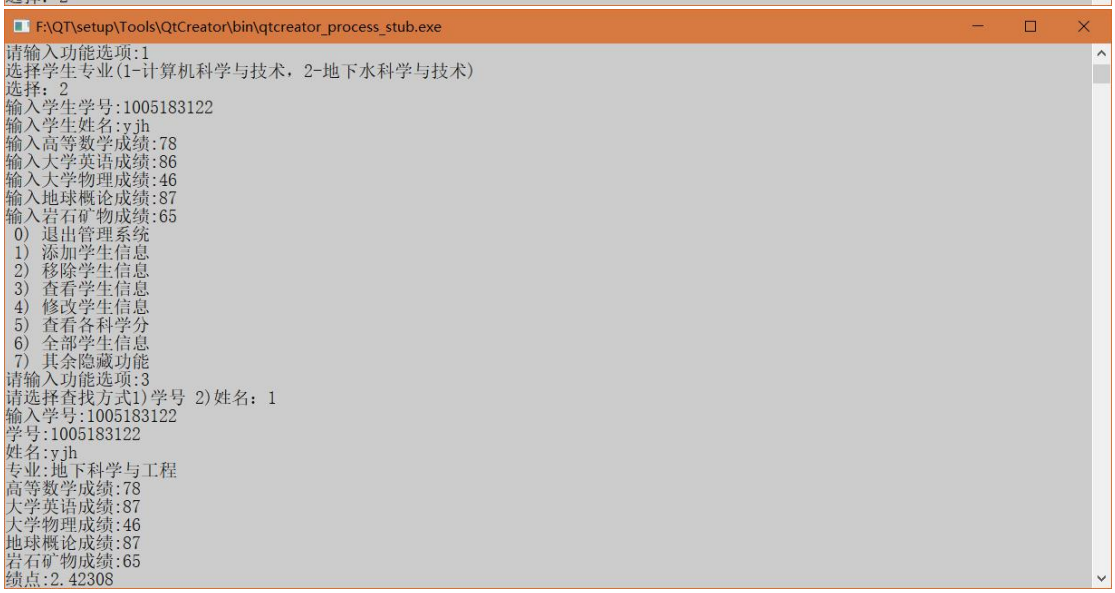
```
246. }
247.
248. void RightWidget::stopBall(){//开始与暂停键-槽方法
249.     if(stopButton->text()==tr("开始游戏")){
250.         pmain->resumeTimer();
251.         stopButton->setText(tr("暂停"));
252.      }
253.     else if(stopButton->text()==tr("暂停")){
254.         pmain->StopTimer();
255.         stopButton->setText(tr("开始"));
256.     }
257.     else{
258.         pmain->resumeTimer();
259.         stopButton->setText(tr("暂停"));
260.     }
261. }
262.
263. void RightWidget::timeDisplay(){//显示运行时间-槽方法
264.     static int count = -1;
265.     count++;
266.     changeTimeLabel2(count);
267. }
```

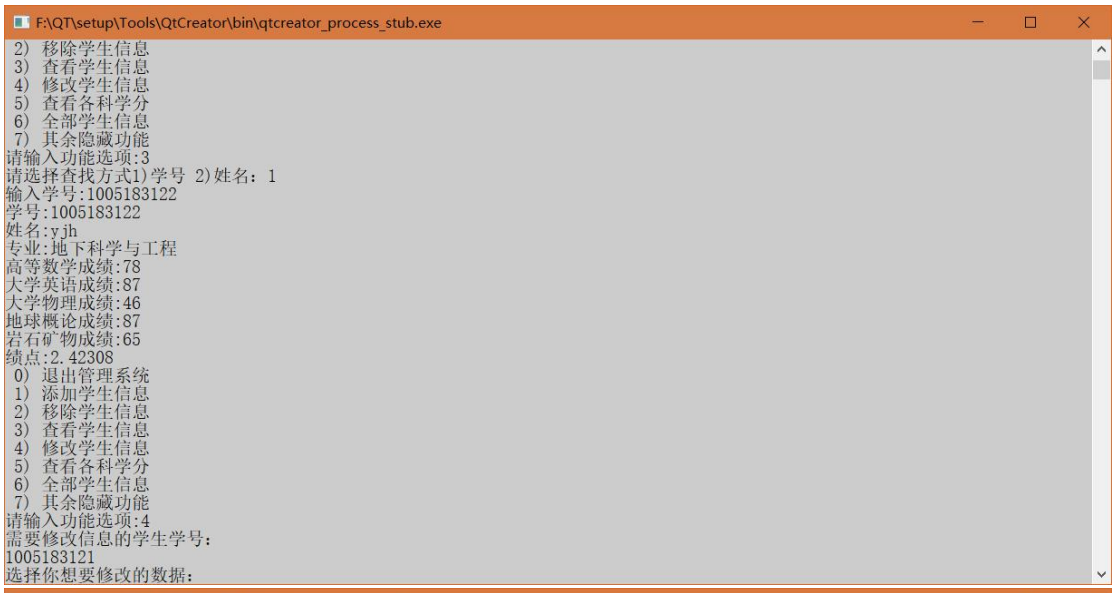
具体的各个代码实现什么操作我都以及打在注释中了，再次就不再赘述了。

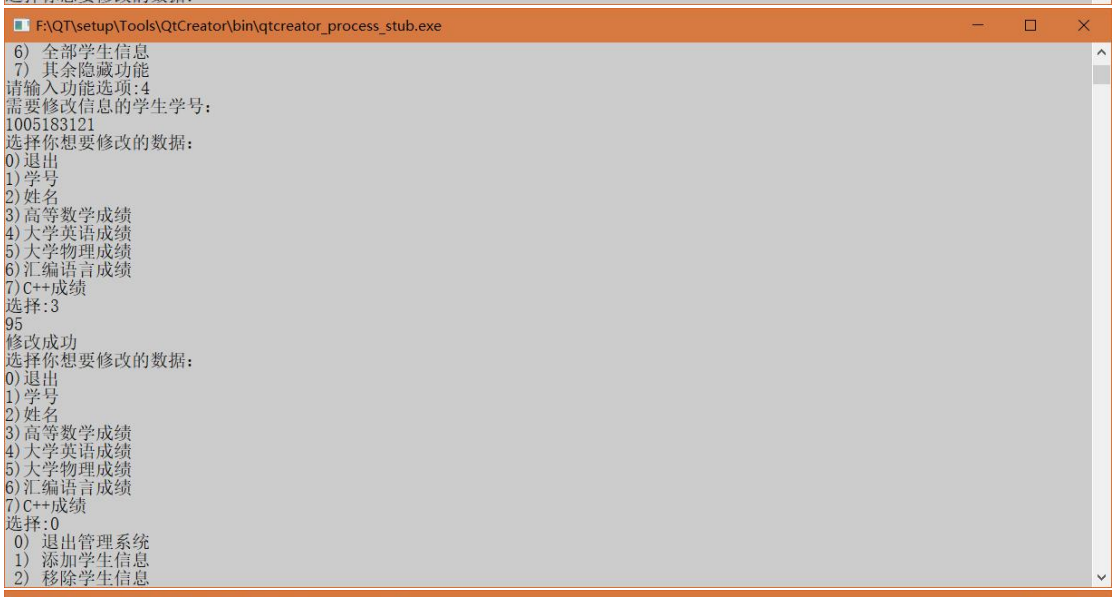
5 系统测试

5.1 学生管理系统程序的测试（控制台程序）

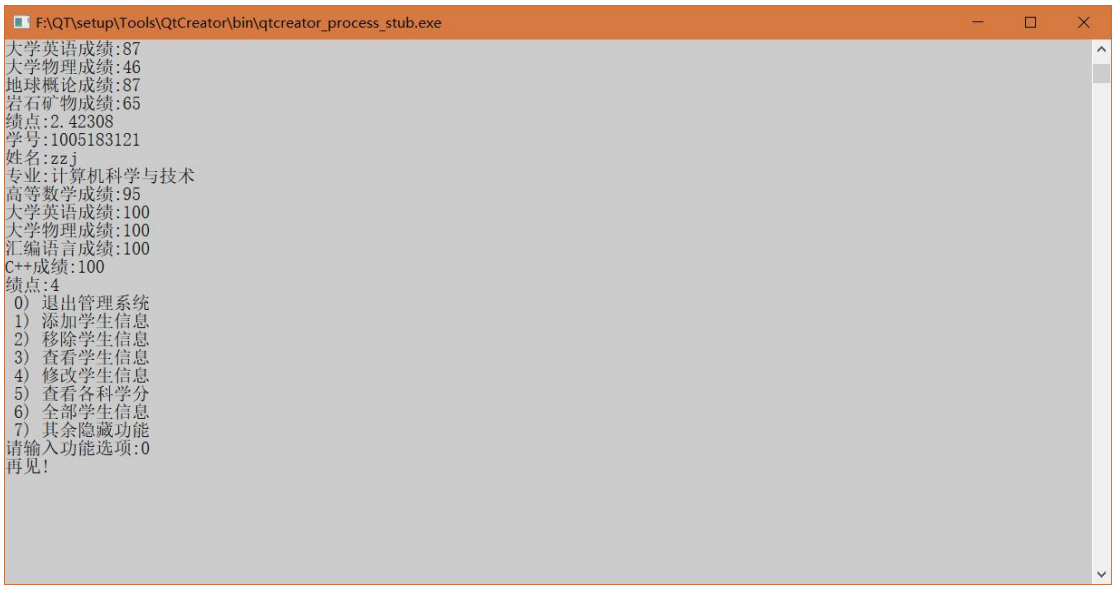





F:\QT\setup\Tools\QtCreator\bin\qtcreator_process_stub.exe
2) 移除学生信息
3) 查看学生信息
4) 修改学生信息
5) 查看各科学分
6) 全部学生信息
7) 其余隐藏功能
请输入功能选项:3
请选择查找方式1)学号 2)姓名：1
输入学号:1005183122
学号:1005183122
姓名:yjh
专业:地下科学与工程
高等数学成绩:78
大学英语成绩:87
大学物理成绩:46
地球概论成绩:87
岩石矿物成绩:65
绩点:2.42308
0) 退出管理系统
1) 添加学生信息
2) 移除学生信息
3) 查看学生信息
4) 修改学生信息
5) 查看各科学分
6) 全部学生信息
7) 其余隐藏功能
请输入功能选项:4
需要修改信息的学生学号：
1005183121
选择你想要修改的数据：


F:\QT\setup\Tools\QtCreator\bin\qtcreator_process_stub.exe
6) 全部学生信息
7) 其余隐藏功能
请输入功能选项:4
需要修改信息的学生学号：
1005183121
选择你想要修改的数据：
0)退出
1)学号
2)姓名
3)高等数学成绩
4)大学英语成绩
5)大学物理成绩
6)汇编语言成绩
7)C++成绩
选择:3
95
修改成功
选择你想要修改的数据：
0)退出
1)学号
2)姓名
3)高等数学成绩
4)大学英语成绩
5)大学物理成绩
6)汇编语言成绩
7)C++成绩
选择:0
0) 退出管理系统
1) 添加学生信息
2) 移除学生信息

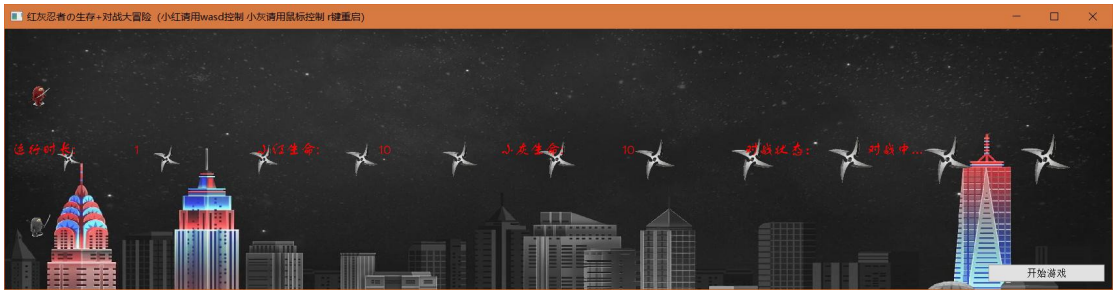

F:\QT\setup\Tools\QtCreator\bin\qtcreator_process_stub.exe
6)汇编语言成绩
7)C++成绩
选择:0
0) 退出管理系统
1) 添加学生信息
2) 移除学生信息
3) 查看学生信息
4) 修改学生信息
5) 查看各科学分
6) 全部学生信息
7) 其余隐藏功能
请输入功能选项:6
学生人数:2
指定排序方式（1-学号，2-姓名，3-绩点）：3
学号:1005183122
姓名:yjh
专业:地下科学与工程
高等数学成绩:78
大学英语成绩:87
大学物理成绩:46
地球概论成绩:87
岩石矿物成绩:65
绩点:2.42308
学号:1005183121
姓名:zzj
专业:计算机科学与技术
高等数学成绩:95
大学英语成绩:100
大学物理成绩:100
汇编语言成绩:100



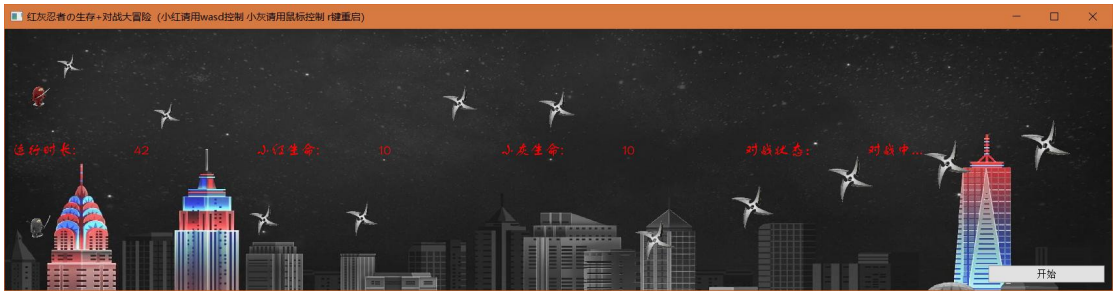
由测试结果可以看出，各个功能运行正常。

5.2 红灰忍者の生存+对战大冒险程序的测试（图形界面程序）

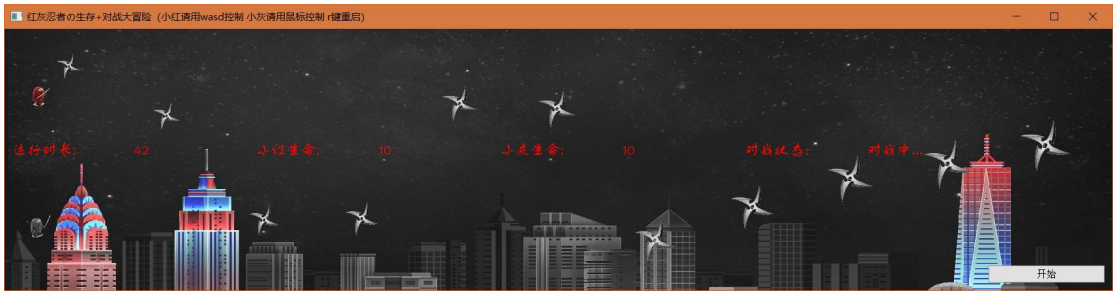
起始界面：



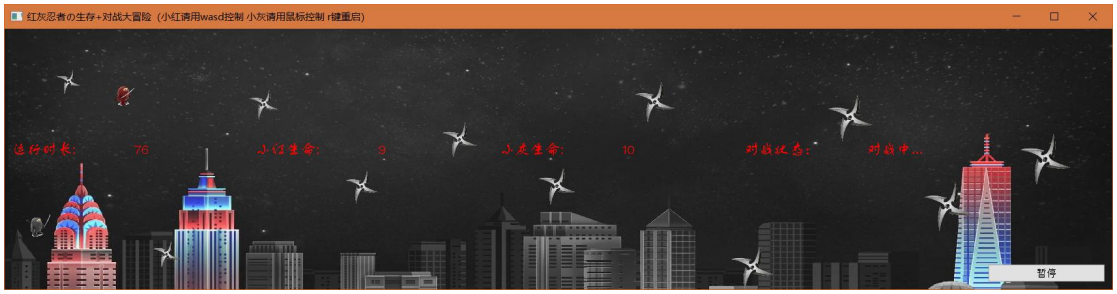
开始游戏：



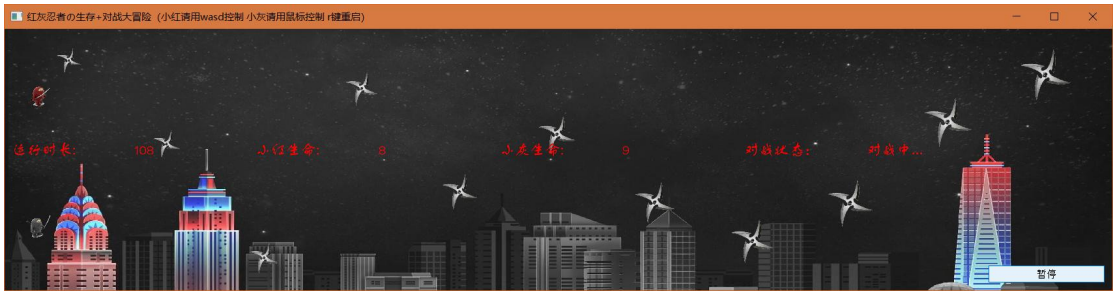
暂停：



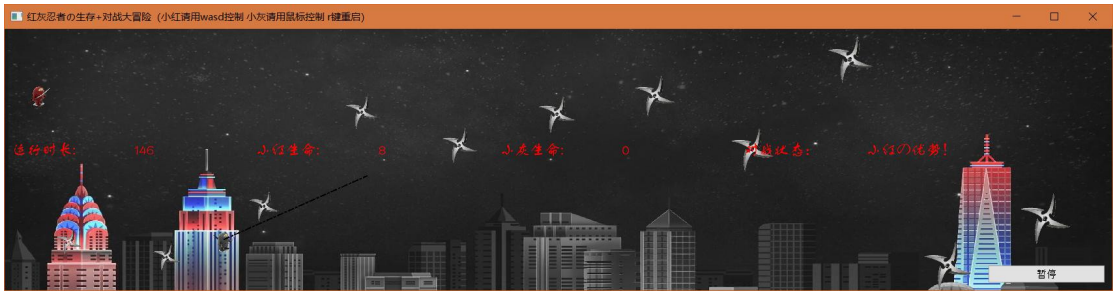
操作小红：



死亡回到起始位置：



操作小灰（同时小灰血量为 0 时注意计分板）：



由测试结果可以看出，程序各功能运行正常。

6 总结与感想

这次的编程我可以说是收益匪浅，毕竟这是我第一次真正意义上自己动手去编一个程序。

在以前，我们学习的 C++都是一些碎片化的知识点，我一直对它缺乏一个整体的认识。但是，学生管理系统带给我的，这种框架等等的东西，老实说，有很大的震撼。都是一些学过的东西，当然也有一些我不太熟悉的东西，比如 STL。当这些东西结合在一起的时候，虽然之前就早已预料，但实际操作时还是会很震惊，能产生如此美丽的结果（不光光是结果本身，还有它的框架等等）。这整整意义上的实现了 C++编程的一部分意义，就是为现实服务，它不再是一个个枯草的知识点的累加，而成为了一个整体。

给我最大震撼的还是可视化这一部分。一开始我真的很难去接受它，虽然夏老师一直再说这个东西很简单。但在一开始，面对这个全新的东西，我很多时候都无从下手，直到听完老师讲解并实际操作之后，才一点点了解并且熟练起来。可视化真的是一个很有意思的东西，我凭借它做出了我的第一个游戏，虽然过程很辛苦，比如我在如何响应键盘按键这一部分几乎消耗了一下午，在标签数值变化部分几乎花了一晚上等等。但当我看着自己的游戏，从一开始的小球碰撞，到有着我自己想法的我自己的小球，再到我可以操纵的小球，最后再到图片填充之后那酷酷的界面，真的有一种老父亲的感觉。

在以前，对我来说，C++真的是一门很虚幻的学科。虽然一直在学习，但我一直对它抱有疑惑。我们学的东西真的能做到那些游戏啊之类的东西吗？它真的有实际的作用吗？可以说，是结课设计让我接触到了真正意义上的编程。这些代码自此有了实际的价值。