**2021数据结构课程设计**

**校园导览系统**

小组成员：

崔继中：2019211307班-2019211342

白钰渲：2019211310班-2019211459

王振浩：2019211307班-2019211332

课题开始日期：2021年3月1日

课程指导教师：杨震

目录

[一． 概述 5](#_Toc74431673)

[1. 编写目的 5](#_Toc74431674)

[2. 项目背景 5](#_Toc74431675)

[3. 成员分工 5](#_Toc74431676)

[4. 参考资料 5](#_Toc74431677)

[二． 总体设计 6](#_Toc74431678)

[1. 需求概述 6](#_Toc74431679)

[2. 系统环境描述 6](#_Toc74431680)

[开发和运行环境 6](#_Toc74431681)

[运行环境 7](#_Toc74431682)

[3. 系统总体结构设计 7](#_Toc74431683)

[软件功能介绍 7](#_Toc74431684)

[软件总体结构说明 8](#_Toc74431685)

[关键技术与算法 8](#_Toc74431686)

[4. 关键数据结构 14](#_Toc74431687)

[地图存储结构 14](#_Toc74431688)

[寻找最短路径算法实现 16](#_Toc74431689)

[主界面 17](#_Toc74431690)

[三． 系统功能设计 21](#_Toc74431691)

[1. 程序流程 21](#_Toc74431692)

[功能描述 21](#_Toc74431693)

[接口设计 21](#_Toc74431694)

[程序流程图 22](#_Toc74431695)

[2. 函数调用 23](#_Toc74431696)

[四． 函数详细设计 23](#_Toc74431697)

[Mainwindow 23](#_Toc74431698)

[1. 主界面构造函数 23](#_Toc74431699)

[2. 初始化菜单栏 24](#_Toc74431700)

[3. 设定按钮及快捷键 24](#_Toc74431701)

[4. 绘制路线 25](#_Toc74431702)

[5. 清除界面 26](#_Toc74431703)

[6. 设置中间必经点的寻路 26](#_Toc74431704)

[7. 鼠标事件 27](#_Toc74431705)

[findShortestPath 27](#_Toc74431706)

[1. 计算起点到各点的最短距离 27](#_Toc74431707)

[2. findShortestPath类的构造，初始化部分参数 29](#_Toc74431708)

[3. 获取路径 29](#_Toc74431709)

[4. 设置各边拥挤度 30](#_Toc74431710)

[5. 创建地图对应的图 30](#_Toc74431711)

[naviAction 31](#_Toc74431712)

[1. 导航寻路 31](#_Toc74431713)

[2. 控制寻路动画 31](#_Toc74431714)

[dialog 32](#_Toc74431715)

[1. 输出日志 32](#_Toc74431716)

[2. 根据程序运行时间映射得到系统内时间 32](#_Toc74431717)

[3. 格式化输出系统内时间 33](#_Toc74431718)

[insidewidget 33](#_Toc74431719)

[setpos 33](#_Toc74431720)

[五． 测试情况 34](#_Toc74431721)

[六． 心得总结 40](#_Toc74431722)

[1. 遇到的问题 40](#_Toc74431723)

[2. 心得与收获 42](#_Toc74431724)

# 概述

## 编写目的

本设计报告的编写目的在于介绍本小组所设计的校园导览系统的设计背景、具体功能，并说明系统的使用方式及各项功能的具体实现方法。

## 项目背景

**所开发软件的名称**：校园导览系统。

**此软件系统的用户**：需要导览系统来辅助参观校园或规划行程的老师、学生等。

## 成员分工

王振浩：模拟过程的设计和实现，日志的输出，前端页面的设计；

崔继中：导航寻路算法的实现，策略的实现，前端页面的设计；

白钰渲：设计导航系统功能和层次，代码模块划分，撰写报告，测试程序。

## 参考资料

1. 课程设计介绍
2. 项目的计划任务书
3. 需求说明
4. 课程设计概要设计书

# 总体设计

## 需求概述

1. 校园中有众多的建筑物和道路，且学校有多个校区（至少2个）。
2. 某个用户于某一时刻向系统提出导航要求，系统根据该同学的要求为其设计一条线路并输出。
3. 系统支持当前用户查询其在模拟导航过程中不同时刻所处的地点和周围的教学楼、宿舍楼、餐饮、后勤服务、操场等地点的功能、位置信息等。
4. 系统支持用户切换功能，可满足同个设备上多个用户的需要。

## 系统环境描述

系统包括的范围：校园导览系统。

### 开发和运行环境

**硬件环境**

|  |  |
| --- | --- |
| 处理器 | Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz |
| 机带 RAM | 8.00 GB |
| 系统类型 | 64 位操作系统, 基于 x64 的处理器 |

**软件环境**

| **名称** | **版本** | **语种** |
| --- | --- | --- |
| Windows 10 | OS版本19042.928 | 简体中文 |
| Visual Studio 2019 | 16.6.1 | 简体中文 |
| Qt Creator | 5.15.2 | 简体中文 |

### 运行环境

安装Windows 10和C++(及Qt等)相关运行库的电脑。

## 系统总体结构设计

### 软件功能介绍

校园导览系统不同于市面上大多数电子地图软件，旨在为教师、学生等群体提供校园内导览、规划路线和出行时间等服务，具有针对性强、定位精准的特点。

本系统支持多种导航策略，可根据用户的实际使用场景提供不同类型的服务，具有寻找到达校内某点的最短路径，获取花费最短时间到达某地的方法，根据各地点的实际功能导航等功能，以满足不同人群的需求。

### 软件总体结构说明

本系统主要结构分为两部分：前端为用户界面，为用户提供多个选择导航内容的接口；后端主要实现按用户的输入寻找相应路径的功能以及根据用户的输入对导航系统进行控制。

### 关键技术与算法

#### Dijkstra算法及其变体

算法一：输入起点，从起点出发使用迪杰斯特拉算法，计算起点到各点的最短距离并记录路径，从中提取出到输入的终点的最短距离和路径。

具体代码实现如下：

/\* 迪杰斯特拉算法，求出起点到各点的最短距离 \*/  
void MainWindow::findShortestPath::findPath(int startPos)  
{  
   /\* 初始化 \*/  
   for (int i = 0; i < Map.vexnum; i++){  
       distance[i] = INF; //初始距离∞  
  }  
   for (int i = 0; i < Map.vexnum; i++){  
       used[i] = false; //处理过的标志  
  }  
   for (int i = 0; i < Map.vexnum; i++)  
  {  
       prev[i] = -1; //前驱  
  }  
​  
   distance[startPos] = 0; //第一个点到它本身的距离设为0  
​  
   while (true)  
  {  
      int v = -1;  
       for (int u = 0; u < Map.vexnum; u++){  
           if (!used[u] && (v == -1 || distance[u] < distance[v]))  
              v = u;  
      }  
​  
       if (v == -1)  
           break; //各个点全被记录过或者v未被修改  
       used[v] = true;  
​  
       for (int u = 0; u < Map.vexnum; u++){  
           /\* 如果u距离大于u-v的距离，更新并记录 \*/  
           if (distance[u] > distance[v] + Map.arc[v][u].dis){  
               distance[u] = distance[v] + Map.arc[v][u].dis;  
               prev[u] = v;  
          }  
      }  
  }

算法二：通过输入终点，从终点出发使用迪杰斯特拉算法，计算每一个点到终点的最短距离并记录路径，从中提取出从输入的起点开始的最短距离和路径。

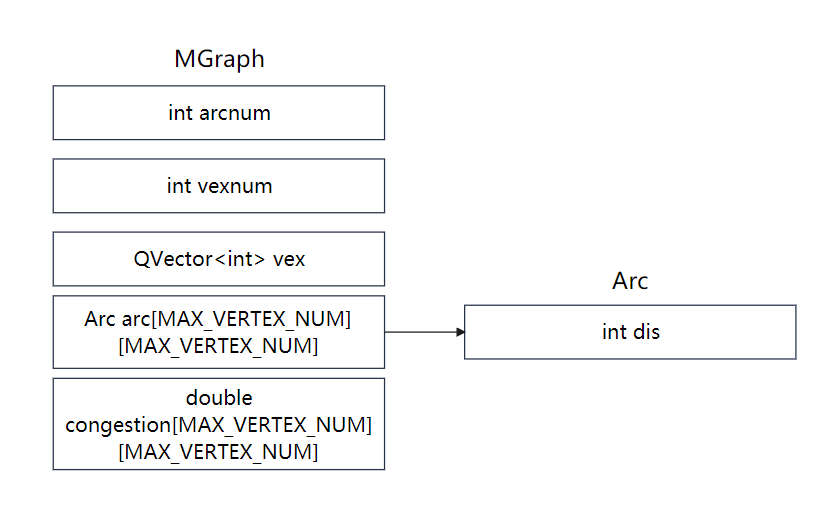
#### Qt路径显示函数

利用Qt软件实现图形化，显示地图，将按照上述算法得到的路径和最短距离在图上体现出来。利用动画效果模拟用户位置变化并将之显示，实现直观的图形化界面。

void MainWindow::paintPath()  
{  
   /\* 根据起点寻路 \*/  
   tour->findPath(startComboBox->currentIndex() - 1);  
   /\* 根据终点选择绘制路线 \*/  
   nextPath = tour->getPath(endComboBox->currentIndex() - 1);  
​  
   /\* 输出寻路的信息 \*/  
​  
   clear(); //清除界面以准备绘制路线  
​  
   for (int i = 1; i < nextPath.size(); i++){  
       qDebug() << nextPath[i] + 1 << "\_\n";  
  }  
​  
   /\* 处理坐标 \*/  
   setStartStation();  
   setEndStation();  
   QPoint tmp1(startX, startY), tmp2(endX, endY);  
​  
   /\* 根据导航信息计算和设置动画 \*/  
   ​  
   QString strtmp, temp;  
   double weight = 0.0;  
   QVector<double> weight\_log;  
​  
   /\* 把坐标存入路径中 \*/  
   for (int i = 1; i < nextPath.size() - 1; i++)  
  {  
       setNextPos(nextPath[i] + 1);  
       ani->setKeyValueAt(weight + 1.0 / (nextPath.size() - 1),QRect(tmpX - 13, tmpY - 50, 30, 40));  
       qDebug() << "tmp:" << tmpX << " " << tmpY << "\_\n";  
​  
       /\* 保存数据以便计时 \*/  
       weight += 1.0 / (nextPath.size() - 1);  
       weight\_log.push\_back(1.0 / (nextPath.size() - 1));  
  }  
  ani->start();  
  double delay = 10000; //总体的时延  
   for (int i = 1; i < nextPath.size() - 1; i++)  
  {  
       Delay\_MSec(weight\_log[i - 1] \* delay);  
       QString str\_temp = startComboBox->itemText(nextPath[i] + 1);  
       temp = QString::number(nextPath[i] + 1);  
      strtmp = "用户已到：" + temp + "->" + str\_temp;  
       printLog(strtmp);  
  }  
​  
   /\* 设置日志信息 \*/  
   Delay\_MSec((1 - weight) \* delay);  
   QString str\_temp =  
       startComboBox->itemText(nextPath[nextPath.size() - 1] + 1);  
   temp = QString::number(nextPath[nextPath.size() - 1] + 1); //获取点的标号  
   strtmp = "用户已到：" + temp + "->" + str\_temp;  
   printLog(strtmp);  
}

## 关键数据结构

### 地图存储结构



//边信息

Arc:

int dis; //权值

//地图信息

MGraph:

int arcnum; //边数

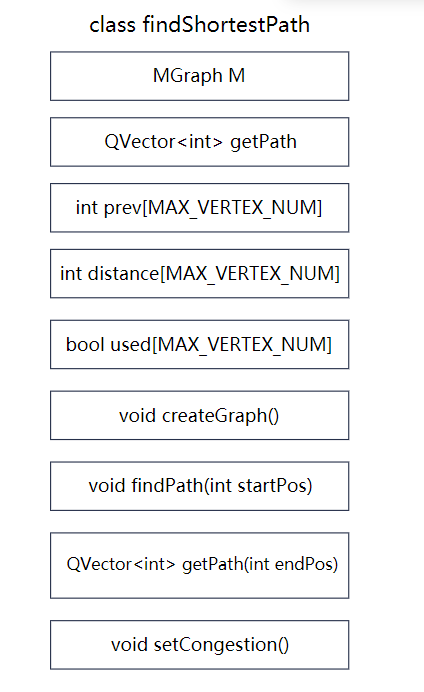
int vexnum; //顶点数

QVector<int> vex; //顶点类型

Arc arc[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]; //存储边的矩阵

double congestion[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]; //保存模拟的路径拥挤度

### 寻找最短路径算法实现

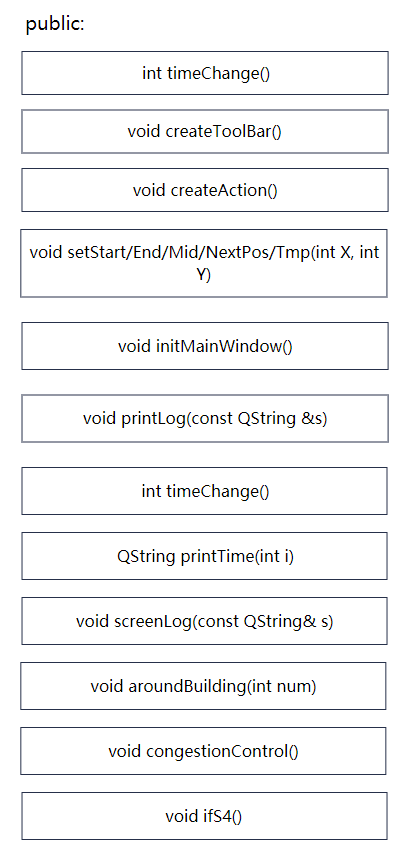


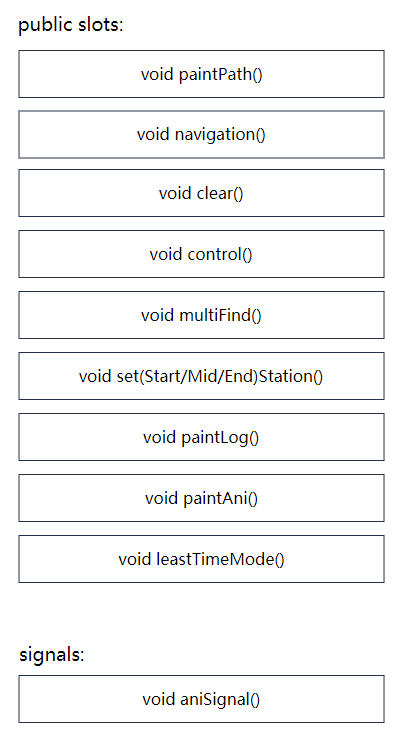
   MGraph Map;  
  QVector<int> getPath(int endPos); //到终点的最短路径  
   void createGraph();  
   int prev[MAX\_VERTEX\_NUM];     //最短路上各前驱结点  
   int distance[MAX\_VERTEX\_NUM]; //表示权值  
   bool used[MAX\_VERTEX\_NUM];    //已被使用过的图  
   void findPath(int startPos);  //求出从起点到各点的最短路径。可改为使用算法二。

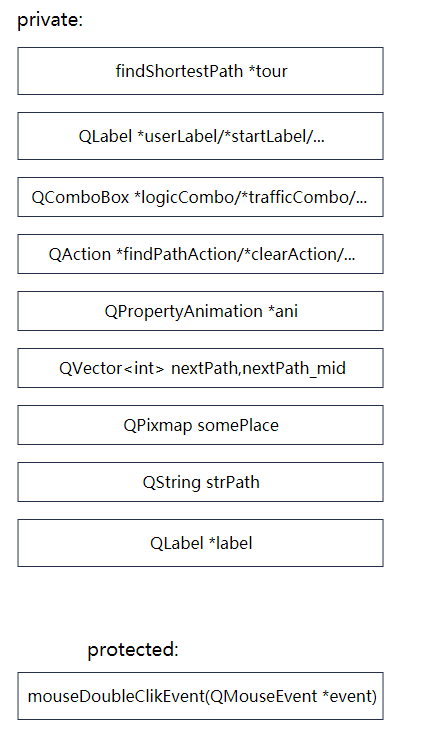
void setCongestion(); //设置各边拥挤度

### 主界面









具体功能描述及实现见**系统功能设计**。

# 系统功能设计

## 程序流程

### 功能描述

用户在界面内选择导览的起点和终点（可以是具体的地点，也可能是按用户需要的功能给出的推荐地点），按需求选择是否有中途必经的某点，利用迪杰斯特拉算法计算起点到每个点的最短距离并记录路径，从记录中提取出到输入的必经点和终点的距离最短（或时间最短）的路径。按下导航按钮后，用户可以看到系统模拟出的导航过程中人的行进情况。

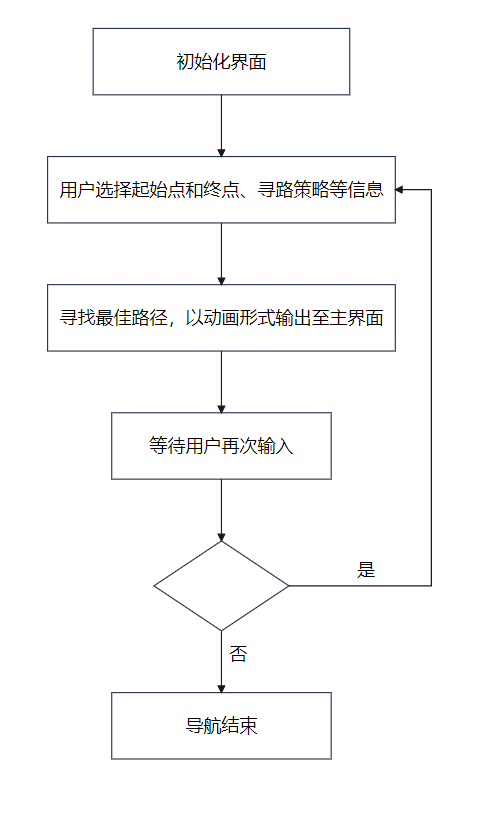
### 接口设计

输入操作：用户可以通过ComboBox控件选择起点、终点、中间必经点，或选择需要的功能；也可以选择交通方式，通过按钮切换最短路径或最短时间策略，模拟导航过程由寻路按钮启动。

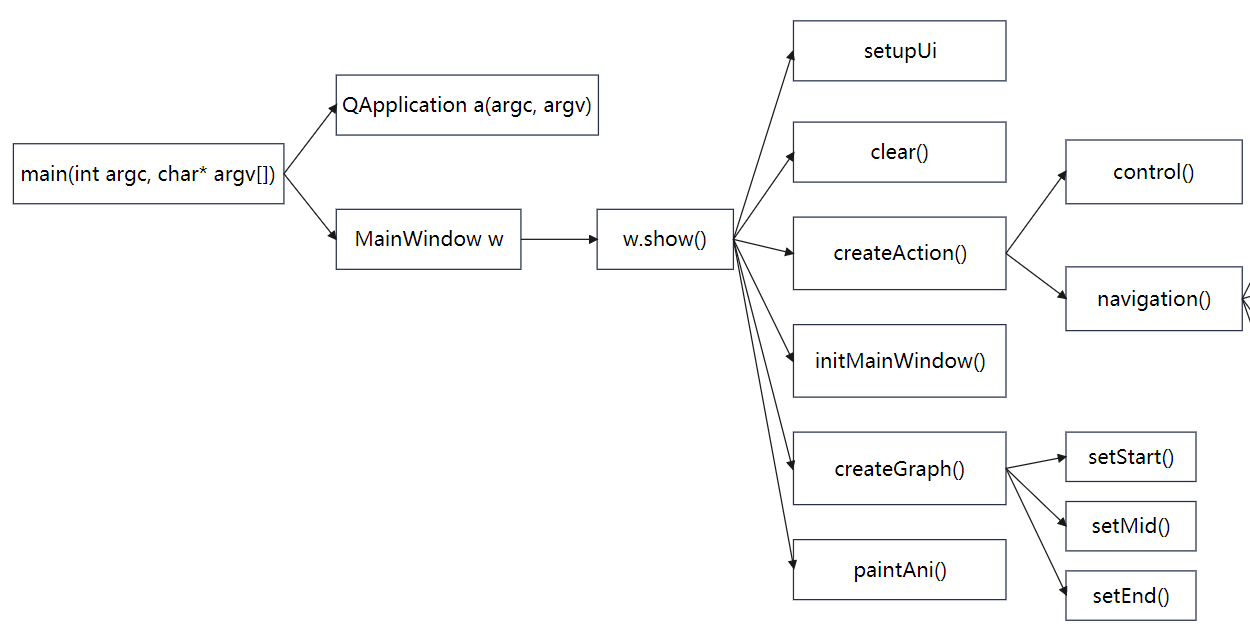
输出效果：界面内的地图上显示模拟导航动画，用户图标按照模拟结果在地图上移动，日志框中显示用户所走过的路劲和路径上各个重要结点的（周围）信息。

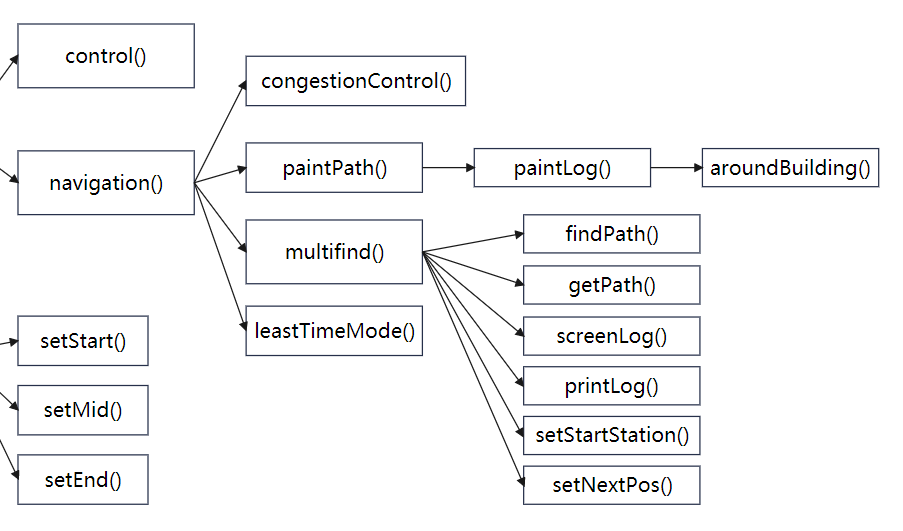
输入操作、启动操作通过界面内的按钮等控件完成。

### 程序流程图



## 函数调用





# 函数详细设计

## Mainwindow

### 主界面构造函数

MainWindow::MainWindow(QWidget \*parent)

  : QMainWindow(parent), ui(new Ui::MainWindow)

{

   初始化主界面;

   创建地图，等待用户输入;

}

### 初始化菜单栏

void MainWindow::initMainWindow()

{

   设置起点和终点;

   设置寻路策略;

   设置交通方式;

}

### 设定按钮及快捷键

在图形化界面的工具栏内设置多个按钮，给用户提供使用各种功能的接口

void MainWindow::createAction()

{

   搜索路径

   清理路径

   中间最短路径

   控制寻路暂停

}

### 绘制路线

模拟导航的关键部分，在此函数中，先完成最短路径的寻找，然后根据储存的最短路，重复地设置动画和输出日志，直到到达终点。此函数实现了对导航、时间的控制。

void MainWindow::paintPath()

{

   tour->findPath(startComboBox->currentIndex() - 1);         //根据起点寻路

   nextPath = tour->getPath(endComboBox->currentIndex() - 1); //根据终点选择绘制路线

   clear(); //清除界面以准备绘制路线

​

   //处理坐标

   setStartStation();

   setEndStation();

   QPoint tmp1(startX, startY), tmp2(endX, endY);

​

   /\* 设置动画 \*/

​

  //输出日志

 //分段输出动画

//先打印起点，再获取终点坐标

//利用ani->start()触发finished()信号

//循环往复，直至寻路结束

}

### 清除界面

清除用户上次导航的设置，重置为默认模式，用户图标恢复原位置。

void MainWindow::clear()

{

   ani = new QPropertyAnimation(ui->frame, "geometry");

   ani->setDuration(100);

   ani->setStartValue(QRect(1, -5, 40, 50));

   ani->setEndValue(QRect(1, -5, 40, 50));

   ani->start();

}

### 设置中间必经点的寻路

void MainWindow::multiFind()

{

   tour->findPath(startComboBox->currentIndex() - 1); //根据起点寻路

   nextPath\_mid = tour->getPath(midComboBox->currentIndex() - 1);

   tour->findPath(midComboBox->currentIndex() - 1);

   nextPath = tour->getPath(endComboBox->currentIndex() - 1); //根据终点选择绘制路线

​

   /\* 其余过程和pathPaint()类似 \*/

}

### 鼠标事件

为了测量各个点的坐标，我们写了这个函数来求鼠标点击位置在窗口内的坐标。

void MainWindow::mousePressEvent(QMouseEvent \*event)

{

   等待鼠标触发事件;

}

## findShortestPath

### 计算起点到各点的最短距离

此处使用dijkstra算法，完成最短路径的寻找。

void MainWindow::findShortestPath::findPath(int startPos)

{

   //初始化各数组

distance[startPos] = 0; //第一个点的距离

   while (true)

  {

       int v = -1;

       for (int u = 0; u < Map.vexnum; u++)

      {

           if (!used[u] && (v == -1 || distance[u] < distance[v]))

               v = u;

      }

​

       if (v == -1)

           break; //全被记录过或者

       used[v] = true;

​

       for (int u = 0; u < Map.vexnum; u++)

      {

           if (distance[u] > distance[v] + Map.arc[v][u].dis)

          { //如果u距离大于u-v的距离，更新并记录

               distance[u] = distance[v] + Map.arc[v][u].dis;

               prev[u] = v;

          }

      }

  }

}

### findShortestPath类的构造，初始化部分参数

MainWindow::findShortestPath::findShortestPath()

{

   初始化各结点和边;

}

### 获取路径

从调用dijkstra算法后得到的路径中提取出到用户指定终点的路径。

QVector<int> MainWindow::findShortestPath::getPath(int endPos)

{

   //用一个vector装路径上各结点，数组中是倒着的，所以要reverse

   QVector<int> pathRes;

​

   for (; endPos != -1; endPos = prev[endPos])

  {

       qDebug() << (endPos + 1) << "...\n";

       //qDebug()<<(endPos)<<"...\n";

       pathRes.push\_back(endPos);

  }

   std::reverse(pathRes.begin(), pathRes.end());

   return pathRes;

}

### 设置各边拥挤度

void MainWindow::findShortestPath::setCongestion()

{ //使用最短时间时调用

int i, j; //排除112，113这两个中途点

初始化各边拥挤度

生成1~21的随机数赋给每条边，使用时乘以边长即为带有拥挤度的边长度

}

### 创建地图对应的图

void MainWindow::findShortestPath::createGraph()

{

   初始化各边各结点信息

}

## naviAction

### 导航寻路

在该函数中根据用户选择的终点进行导航策略的控制：不同策略默认给出最短路径。终点中逻辑位置优先级最高，其次根据用户是否选择中途必经点调用不同函数。

void MainWindow::navigation()

{

if 选择了逻辑功能

       paintPath();

   if 当前坐标为起点

       paintPath();

   else

       multiFind();

}

### 控制寻路动画

此处使用Qt库函数，控制模拟导航动画的暂停和启动。

void MainWindow::control()

{

   if (userComboBox->currentIndex() == 0)

  {

       if 动画为暂停状态

           ani->resume();

       else if 动画正在播放

           ani->pause();

}

}

## dialog

### 输出日志

void MainWindow::printLog(const QString &s)

{

  打开日志文件(ReadWrite)

以Append模式向文件中输入日志

关闭文件

}

### 根据程序运行时间映射得到系统内时间

int MainWindow::timeChange()

{

   QTime current\_time = QTime::currentTime();

   int time\_all;

   time\_all = current\_time.hour() \* 60 \* 60 + current\_time.minute() \* 60 + current\_time.second();

   int time\_change;

   time\_change = time\_all % 7200; //相当于两小时

   return time\_change;

}

### 格式化输出系统内时间

QString MainWindow::printTime(int i)

{

   QString time\_return;

   time\_return += QString::number(i / 300) + ":" + QString::number((i % 300) / 5);

   return time\_return;

}

## insidewidget

层次化设置Widget，根据所选按钮切换不同的Widge。

## setpos

设置各点坐标。（包括setStart, setMid, setEnd, setTmp, setStartStation）

# 测试情况

1. 首先我们的系统使用了图形化界面，对用户很友好，且功能多样。在界面中我们显示了自己绘制的模拟沙河-西土城校区地图，并给用户提供了多种可选择的导航选项。



1. 默认使用最短路径策略进行导航，在开始导航时会提示。



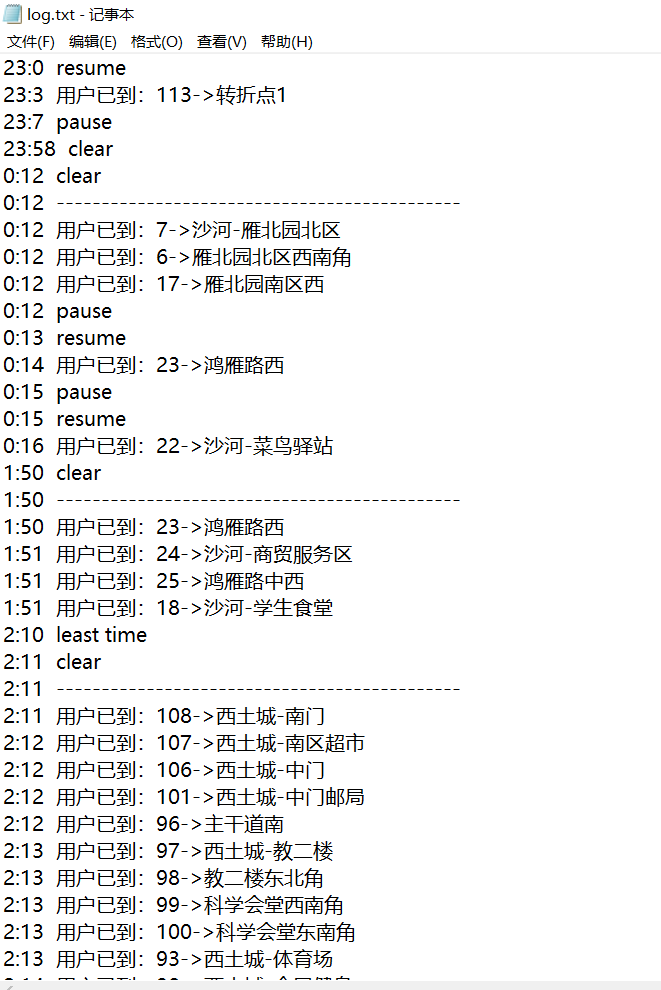
导航过程中，动画人物会在界面中做相应的移动，并把导航的时间、到达位置、途径位置，用户的启动、暂停等操作都在日志框中显示出来。

中途暂停后，动画、日志输出都暂停。

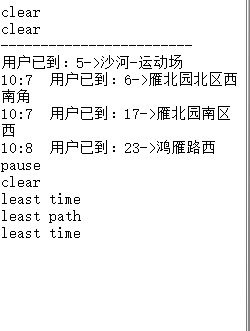
且本导航系统支持导航过程中更改目的地 如下图



在模拟导航过程中，会同时在屏幕和文件中记录日志，对用户模拟导航的内容是保持一致的。



1. 使用最短时间导航策略，日志框会显示用户切换导航策略这一行为。再次点击，会切回最短路径模式



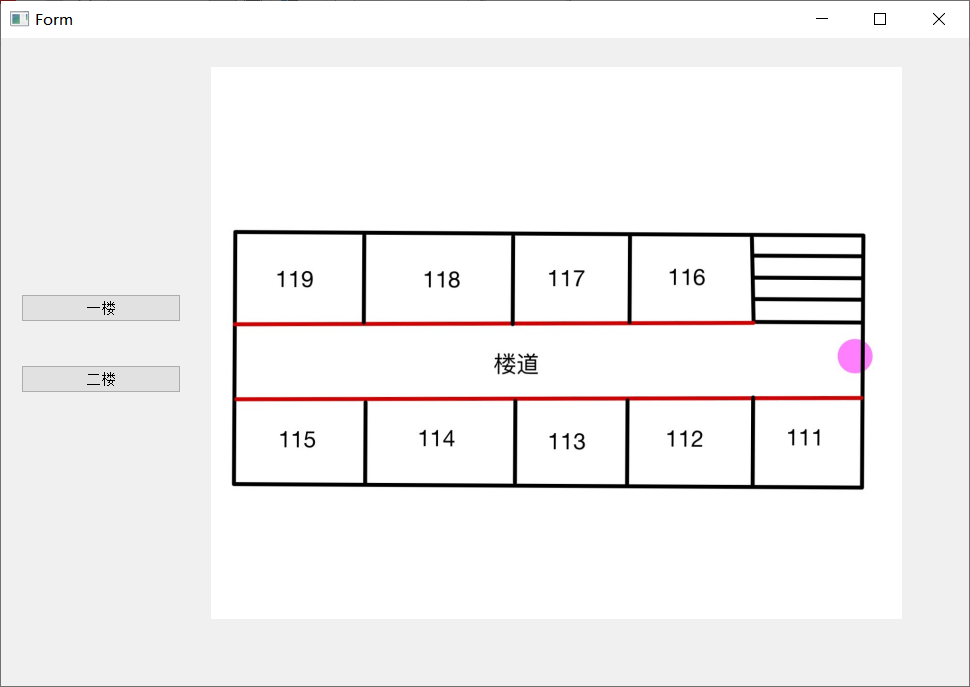
1. 途径某必经点和使用多种交通工具的策略，在我们的程序中，用户可以选择一个导航过程中必须途径的点，导航过程中会先从起点到达该点再去往终点，默认采用最短路径策略完成导航。同时用户可以设置在校园中步行或骑自行车，本系统支持在导航过程中切换交通方式，并以模拟人物移动速度的不同体现两种交通工具的区别。

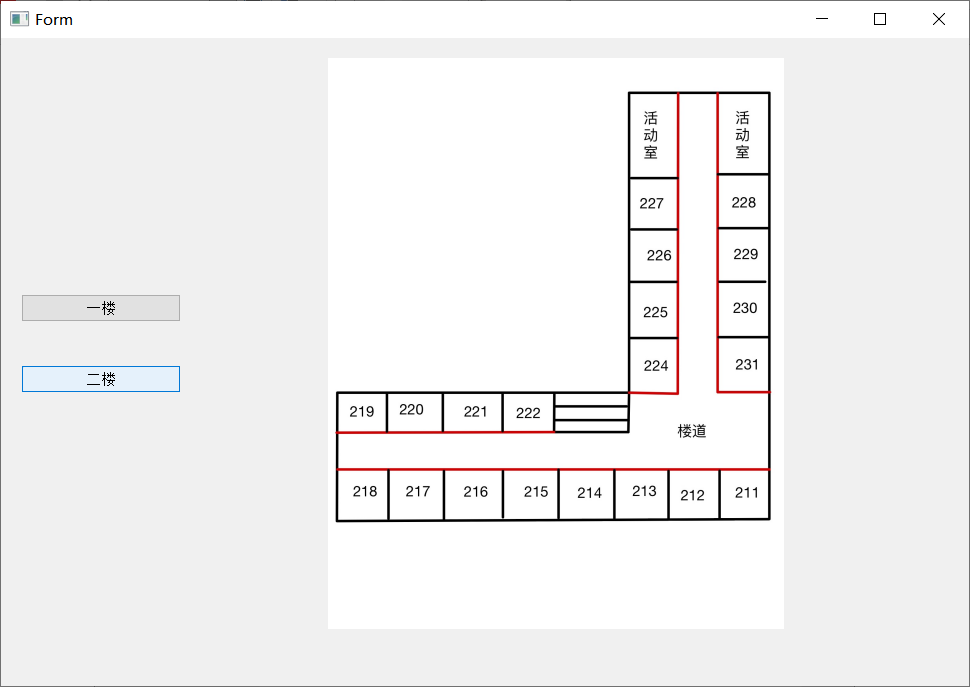




1. 本程序支持沙河雁南园S4楼内导航，如图所示，如果把导航的终点设置为S4，在导航结束后询问用户是否进入S4，进入以后，会在日志输出“进入S4”，并显示出男生宿舍S4内部的部分地图，给用户提供楼内精确到不同楼层、不同房间号的导航，如下图：







1. 本系统支持按用户的功能需求提供建议，并支持根据时间模拟拥塞控制。





根据本系统模拟的时间，我们设置了针对同一功能，在不同时间给出不同的导航建议——用以模拟不同时间段内，几个食堂、超时等地方人流量的不同。给出建议后，自动把该地点设置为终点，采用默认策略模拟导航。

# 心得总结

## 遇到的问题

* 导航时需要在地图中给用户提供可选择的地点，而实际地图中展示的路径上有明确名字比如“XX楼”的地点数量有限，仅用这些点来进行导航，得到的路径不符合实际；有些地点位于建筑物内，其在建筑内部的路径不便于直接展示。

解决办法：对地图进行一定程度的抽象化，有明确功能的、特别重要的地点，我们在图中标注出来，对于较长的路、路口等重要但无明确名称的点，我们也进行标记但不展示给用户，用于寻路过程的中继和转换结点。测试时部分路径寻路动画会偏离原路径。由于我们用于模拟导航的动画是按坐标运行的，而我们手动标记出的点的坐标有屏幕坐标、window坐标等，并需要考虑图片大小、窗体坐标系统的原点位置等条件，需要给点出的坐标加一个偏移量，以调整动画的位置。

后来发现，由于图层重叠嵌套问题，我们很难获得我们想要的某一图层的某一个点的真实坐标。在经过大量的尝试之后，我们选择重新架构，使用UI界面直接加载地图，后续操作都在这个不会移动的地图上进行，这样我们就可以获得一个不变的坐标，偏移量也就很好解决。

* 实时模拟的动画、日志两个行为匹配不好

我们开始是对一整段动画整体模拟，设置中间节点和不同节点间的时延来进行动画，日志使用同样的时延进行输出，相当于日志是和动画在表面上统一，在深层次脱离。后来发现，如果这样绘制的话，当动画暂停的时候，日志的输出不会暂停。经过思考和尝试后，我们决定使用多段动画分别绘制，当一段动画结束后，立即输出此段的日志，从而达到日志和动画的匹配。但这要涉及到Qt的connect问题，槽函数和信号对我们来说是一个不好实现的过程。经过大量资料的查阅和实验之后，我们最终解决了问题：初始动画—>日志—>动画—>日志……直到动画结束。

## 心得与收获

王振浩：

通过本次课程设计，我对一个项目的整体架构有了更加深入的理解和认识，明白了项目计划的重要性以及根据计划行事的必要性。在开发的过程中，对数据结构的各种知识有了更加切实的理解和体会，在实践中应用了自己学到的知识，加深巩固了理论知识。同时学习到了新的软件——Qt的使用方法，在图形化界面及交互响应方面有了自己的认识。对知识的检索和获取能力大大提升，应对问题的耐心大大提升。开发过程中遇到了很多棘手的问题，通过对知识的搜索和学习，经历过数次波折后，我们都一一解决，而不是中途放弃。

崔继中：

此次数据结构课程设计加强了我的动手实践能力和对理论知识的实际应用能力，培养了我从多个角度考虑问题的思维方式。回顾起此次课程设计，我感慨良多。从拿到题目到完成整个编程，从理论到实践，在两个多月的时间里，我在发现问题、解决问题的过程中，不仅巩固了以前所学过的知识，还提升了自学能力，借助各种资料学到了很多课堂以外的知识。通过这次课程设计使我懂得了理论与实际相结合是很重要的，只有理论知识是远远不够的，只有把所学的理论知识与实践相结合起来，从理论中得出结论，才能真正提高自己的动手能力和独立思考能力。与此同时，我的团队协作能力也通过本次课设得以锻炼。在分析问题时，我们积极讨论，想出多种实现方法并一一分析它们的可行性；遇到问题时，我们为了一个bug可以死磕几个小时……这些难忘的回忆和体验，让我更加深刻地明白了团队精神的重要性。今后，无论是独立还是团队合作完成一个项目时，相信这次课设的宝贵经验都会成为前行路上指引我的一盏明灯。

白钰渲：

在本次数据结构课程设计之前，我和同学合作完成一个项目的经历屈指可数。仅仅靠理论学习不足以让我真正理解不同的数据结构被设计出来的意义，只有通过动手实践，我才深刻体会到其中的奥妙。与此同时，在完成课设的过程中，我的文档编写能力、团队协作能力、分析问题和解决问题的能力也得到了充分的锻炼。从一开始拿到课设要求时的无从下手，到查找相关资料，再到共同完成编写代码和debug工作，程序真正跑起来那一刻，我们每个人都激动万分。经过自己努力换来的成功，是由内而外的高兴。总而言之，这次的课程设计真正激发了我对数据结构的兴趣，也希望以后有更多的机会深入学习这门学科。