# 成都地铁查询系统

**Start from 4-16-2016**

# 要求

## 基本要求

• 假设成都已经建成了10条地铁形成地铁网络，要求该系统应用在地铁口的查询机上。目的是供旅客查询地铁的站点，票价，时间等信息。

• 利用数据库存储这些信息，自动计算出出发地和目的地如何换乘和价格。计算方法有最短路径，时间等选择。

• 在过程中需要考虑到程序的易用性，程序提供地图供用户选择出发点和目的地。

## 我的添加

允许用户绘制地铁线路：

基于已有线路底图

对已经完成的线路进行可视化修改**(针对站点的修改)**

线路的颜色更改

地图特点：

地图可以拖动

地图可以缩放

拥挤程度

通过MetroLookUp程序提供的输入接口来获得关于人流量的信息

预计等待时间

在绘制地图的过程中已经确定。等待时间分为右行耗时和左行耗时。即从右（左） 边出发，由上一个站到该站的估计用时（另外，还可以加上等待时间）

票价

在绘制地图的过程中已经确定。同样分为右行票价和左行票价。从右（左）开始，由上一站到本站需要的票价

# 实现

#### **概要**

我把程序分成了两部分：

MetroLookUp用于查询的查询端

MetroLookUp-LineDrawer地铁线路绘制的绘制端

# 地图文件格式(mld:metro line data)

组成

* 1. 线路颜色数据
  2. 线路绘图粗细数据
  3. 线路每一点的数据

\*a，b属于地图头数据（HEAD），每一点的数据为地图体（BODY）

各部分数据域说明

* 1. **地图头**

地图头由以下结构体定义：

Struct MMD\_HEAD{

Short RGB[3]; //地图线路的颜色

Short bound; //线路绘制粗细

}

* 1. **地图体**

地图体由以下结构体定义：

|  |
| --- |
| struct MMD\_NODE {  CPoint p; //线路中每个点的位置  int id; //如果該点为站台，则这里会有编号，否则为0  char name[30]; //储存站点名  int line = 0; //储存父线路  int ltime = 0; //储存时间  int rtime = 0;  int lprice = 0; //储存票价  int rprice = 0;  bool transfer = false; //标记是否是换乘站点  }; |

# 地图的绘制

地图的绘制存在几种不同的情况：

第一种：地图的重绘，第一次读入

程序首先读入一个线路列表文件（line.ld），在这个文件中存放了所有地铁线路文件的路径。

我们再根据这个地铁线路列表文件来读入每一条线路的数据

当遇到站点的时候，即MMD\_NODE中的id非0时，就根据这个ID去数据库查找相应的站点数据，然后绘图；

第二种：地图在进行平移后的重绘

第三种：地图在缩放后的重绘

## 重难点：

1. 地图的拖动
   1. 这里我用的是当鼠标左键按下与鼠标移动的时候的偏移量来实现平移的
      1. 在移动后，将所有线路写入同一个文件，来实现拖动
2. 地图的缩放
   1. 可能的方法：每次点击放大，我们把线路重新绘制，使其粗细增加m倍，每条线的长度增加k倍
   2. 或者是使用坐标系缩放（这里还需要思考）
3. 当用户单击的时候，地图站点的判断
   1. 可能的方法，判断以被单击点为圆心，半径r内是否有站点存在（抽象为以被单击点为起点，其相关联的连续线路k上，路程耗散x以内是否存在站点）
4. 换乘站点绘制

# 数据库内容

表：

|  |  |
| --- | --- |
| LINE\_DATA | 用来储存线路数据：线路ID及其对应的文件名 |
| LINECOUNT | 只有一行数据，用来储存当前有多少条线路 |
| STATION\_DATA | 用来储存站点数据：站点ID，站点名称，站点双向时间耗散，站点双向票价（从起点到该站的票价与从终点到该站的票价） |
| TRANSFER\_STATION | 用来储存换乘站点数据：站点ID，与该站点属于哪些线路 |

该文件用来测试算法BinRW正确性

# 成都地铁查询系统

**Start from 4-16-2016**

# 要求

## 基本要求

• 假设成都已经建成了10条地铁形成地铁网络，要求该系统应用在地铁口的查询机上。目的是供旅客查询地铁的站点，票价，时间等信息。

• 利用数据库存储这些信息，自动计算出出发地和目的地如何换乘和价格。计算方法有最短路径，时间等选择。

• 在过程中需要考虑到程序的易用性，程序提供地图供用户选择出发点和目的地。

## 我的添加

允许用户绘制地铁线路：

基于已有线路底图

对已经完成的线路进行可视化修改**(针对站点的修改)**

线路的颜色更改

地图特点：

地图可以拖动

地图可以缩放

拥挤程度

通过MetroLookUp程序提供的输入接口来获得关于人流量的信息

预计等待时间

在绘制地图的过程中已经确定。等待时间分为右行耗时和左行耗时。即从右（左） 边出发，由上一个站到该站的估计用时（另外，还可以加上等待时间）

票价

在绘制地图的过程中已经确定。同样分为右行票价和左行票价。从右（左）开始，由上一站到本站需要的票价

# 实现

#### **概要**

我把程序分成了两部分：

MetroLookUp用于查询的查询端

MetroLookUp-LineDrawer地铁线路绘制的绘制端

# 地图文件格式(mld:metro line data)

组成

* 1. 线路颜色数据
  2. 线路绘图粗细数据
  3. 线路每一点的数据

\*a，b属于地图头数据（HEAD），每一点的数据为地图体（BODY）

各部分数据域说明

* 1. **地图头**

地图头由以下结构体定义：

Struct MMD\_HEAD{

Short RGB[3]; //地图线路的颜色

Short bound; //线路绘制粗细

}

* 1. **地图体**

地图体由以下结构体定义：

|  |
| --- |
| struct MMD\_NODE {  CPoint p; //线路中每个点的位置  int id; //如果該点为站台，则这里会有编号，否则为0  char name[30]; //储存站点名  int line = 0; //储存父线路  int ltime = 0; //储存时间  int rtime = 0;  int lprice = 0; //储存票价  int rprice = 0;  bool transfer = false; //标记是否是换乘站点  }; |

# 地图的绘制

地图的绘制存在几种不同的情况：

第一种：地图的重绘，第一次读入

程序首先读入一个线路列表文件（line.ld），在这个文件中存放了所有地铁线路文件的路径。

我们再根据这个地铁线路列表文件来读入每一条线路的数据

当遇到站点的时候，即MMD\_NODE中的id非0时，就根据这个ID去数据库查找相应的站点数据，然后绘图；

第二种：地图在进行平移后的重绘

第三种：地图在缩放后的重绘

## 重难点：

1. 地图的拖动
   1. 这里我用的是当鼠标左键按下与鼠标移动的时候的偏移量来实现平移的
      1. 在移动后，将所有线路写入同一个文件，来实现拖动
2. 地图的缩放
   1. 可能的方法：每次点击放大，我们把线路重新绘制，使其粗细增加m倍，每条线的长度增加k倍
   2. 或者是使用坐标系缩放（这里还需要思考）
3. 当用户单击的时候，地图站点的判断
   1. 可能的方法，判断以被单击点为圆心，半径r内是否有站点存在（抽象为以被单击点为起点，其相关联的连续线路k上，路程耗散x以内是否存在站点）
4. 换乘站点绘制

# 数据库内容

表：

|  |  |
| --- | --- |
| LINE\_DATA | 用来储存线路数据：线路ID及其对应的文件名 |
| LINECOUNT | 只有一行数据，用来储存当前有多少条线路 |
| STATION\_DATA | 用来储存站点数据：站点ID，站点名称，站点双向时间耗散，站点双向票价（从起点到该站的票价与从终点到该站的票价） |
| TRANSFER\_STATION | 用来储存换乘站点数据：站点ID，与该站点属于哪些线路 |

该文件用来测试算法BinRW正确性