

《程序设计课程设计》实验报告

实验名称：自动驾驶公交车调度系统概要设计(动画版)

班级：2021211306

组号：16

组员姓名：

蒲俊宋 (2021211116) (组长)

罗奕祺 (2021211115)

毛楚奕 (2021211113)

| 编号 | 日期 | 版本号 | 章节 | 编写者 | 说明 |
|----|----------|------|----|-----|-----------------------|
| 1 | 2022-7-4 | V1.0 | 全部 | 毛楚奕 | 根据《概要设计报告模板(动画版)》撰写报告 |

1. 用户界面设计



图1


如图1所示

画面要素:

- (1)左边为车辆状态栏，可显示当前车辆位置、当前车辆状态（顺时针，逆时针或者停车。图中显示为停车状态）、当前时间
- (2)右边为请求列表栏，可显示所有未完成请求的服务类型、服务类型和生成时间
- (3)中间有更新请求，下一秒和新增请求三个按钮
- (4)下方是显示公交车以及车站位置的地图

操作方法:

(1)点击“新增请求按钮”,弹出图2所示窗口

 新增请求窗口

—

□

×

请求类型

顺时针请求 ▾

请求站点

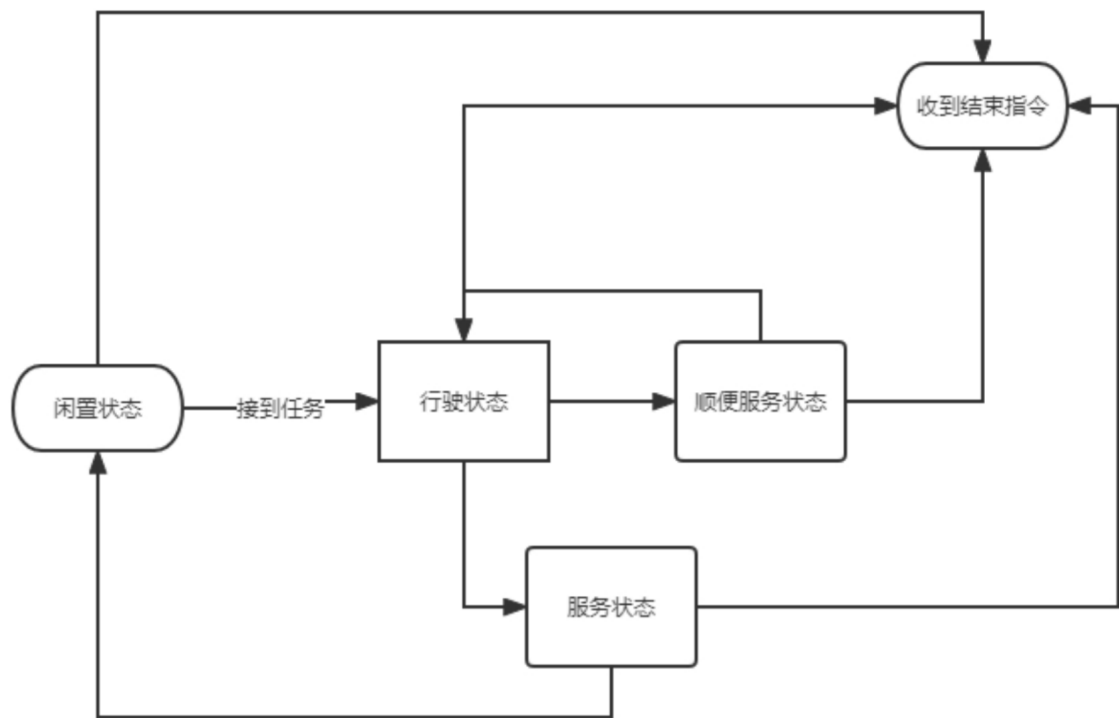
提交请求

清空内容

图2

- (2)输入目标站点，选择请求类型，点击提交请求。完成后右部请求列表栏会出现对应请求。某一时刻如需多次添加请求，重复上述步骤即可。
- (3)回到主窗口，点击"下一秒"按钮，当前时间+1

2. 自动机模型设计



3. 高层数据信息设计

3.1全局常量定义

```
#define MODE_FCFS 1 //调度类型 -- 先来先服务
#define MODE_SSTF 2 //调度类型 -- 最优服务
#define MODE_SCAN 3 //调度类型 -- 顺便服务
#define CLOCK 1 //接到指令 -- 时钟指令
#define NEW_QUERY 2 //接到指令 -- 新任务指令
#define END 3 //接到指令 -- 调度结束
#define CLOCKWISE 1 //任务类型 -- 顺时针站台
#define COUNTERCLOCKWISE 2 //任务类型 -- 逆时针站台
#define TARGET 3 //任务类型 -- 车内
```

3.2全局结构体定义

```
typedef struct Query { //请求结构体定义
    int type; //请求类型定义
    int station; //请求所在站台
    int genTime; //请求生成时间（clock 时间）
} Query ;
```

3.3全局变量定义

```
//变量定义
int distance, strategy, totalStation, totalDistance;

//请求列表和当前请求个数
Query QueryList[1000];
int QueryNumber=0;
```

```

//当前站点请求情况
int isCounterclockwiseQuery[1001]={0};
int isClockwiseQuery[1001]={0};
int isBusQuery[1001]={0};

//巴士任务执行状态
int isInCharge = 0;
int stationGoto = 0; // station, not position

//巴士变量
int busPosition = 0;
int nowDirection = CLOCKWISE;
int nowClock = 0;

//原config_reader.c
//读取配置文件
int g_TotalStation = 5;
int g_distance = 2;
int g_mode = MODE_FCFS;
int g_totalDistance = g_distance*g_TotalStation;
int flag_isinit = 1;

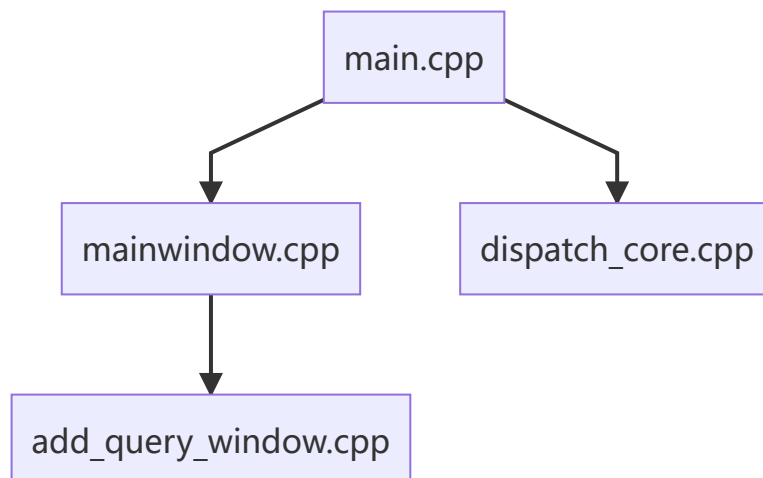
//配合新函数使用
int s_isNearby;
int s_stationGoto;
int s_isinfinish;

// MODE_SCAN第一次任务确定方向标志位
int flagFirstTime = 1;

```

4. 系统模块划分

4.1 系统模块结构图



4.2 模块说明

| 模块文件 | 模块功能 | 模块包含的函数名 | 函数功能 |
|-------------------|------|---------------------|------------------|
| main.cpp | | main | 主函数 |
| | | | |
| dispatch_core.cpp | | g_initDispatchCore | 读取配置文件部分 |
| | | g_getTotalStation | 返回总的车站数量 |
| | | g_getDispatchMethod | 返回调度模式 |
| | | g_getDistance | 返回总的距离 |
| | | g_getNowQueryList | 返回临时的拷贝请求列表 |
| | | g_insertNewQuery | 插入新的请求 |
| | | getNextTask | 返回将执行的任务 |
| | | hasNearBy | 顺便服务判断 |
| | | fulfillQuery | 到目的地后清理请求列表 |
| | | getClockwiseDist | 返回顺时针距离 |
| | | getCounterclockwise | 返回逆时针距离 |
| | | getStationPosition | 返回车站的位置 |
| | | moveForward | 前进函数 |
| | | updateOutputList | 任务删除后更新输出列表 |
| | | chooseDirection | FCFS SSTF启动前确定方向 |
| | | keepDirectionValid | SCAN检查是否需要变向 |
| | | firstTimeScanGetDir | SCAN首次寻找任务 |
| | | outputResult | 输出函数 |
| | | g_nextClock | 完成下一时刻的动作 |
| | | g_getNowState | 获取当前车辆状态 |

| 模块文件 | 模块功能 | 模块包含的函数名 | 函数功能 |
|----------------------|------|--------------------------------|------|
| mainwindow.cpp | | MainWindow | / |
| | | ~MainWindow | / |
| | | on_updateRequestButton_clicked | / |
| | | on_createRequestButton_clicked | / |
| | | handleNewQuery | / |
| | | on_updateClockButton_clicked | / |
| | | eventFilter | / |
| add_query_window.cpp | | add_query_window | / |
| | | ~add_query_window | / |
| | | on_buttonSubmit_clicked | / |
| | | on_buttonClear_clicked | / |

4.3 函数说明

| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
|----|---|-------------|---|------|
| 1 | int main(int argc, char *argv[]) | / | / | int |
| 2 | void g_initDispatchCore() | 读取配置文件部分 | / | void |
| 3 | bool g_insertNewQuery(Query newQuery) | 插入新的请求 | newQuery为将插入的新的请求 | bool |
| 4 | int getNextTask(Query queryList[], int queryListNumber, int direction, int strategyType, int totalStation, int distance, int nowStation) | 返回将执行的任务 | queryList为请求列表 queryListNumber为列表大小 direction为当前行驶方向 strategyType为策略类型 totalStasion为总站台数量 distance为站间距离 nowStation为当前所在站 | int |
| 5 | int hasNearBy(Query queryList[], int *queryListNumber, int strategyType, int totalStation, int distance, int nowStation, int nowTime, int direction) | 顺便服务判断 | queryList为请求列表 queryListNumber为指向列表大小的指针 strategyType为策略类型 totalStasion为总站台数量 distance为站间距离 nowStation为当前所在站 nowTime为当前时间 direction为当前行驶方向 | int |
| 6 | int fulfillQuery(Query queryList[], int *queryListNumber, int direction, int strategyType, int totalStation, int distance, int nowStation) | 到目的地后清理请求列表 | queryList为请求列表 queryListNumber为指向列表大小的指针 direction为当前行驶方向 strategyType为策略类型 totalStasion为总站台数量 distance为站间距离 nowStation为当前所在站 | int |
| 7 | void moveForward() | 前进函数 | / | void |
| 8 | void updateOutputList() | 任务删除后更新输出列表 | / | void |

| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
|----|--|-------------------------|----|------------------|
| 9 | void chooseDirection() | FCFS SSTF启动 前确定方向 | / | void |
| 10 | void keepDirectionValid() | SCAN检查是否需要变向 | / | void |
| 11 | void firstTimeScanGetDir() | SCAN首次寻找任务 | / | void |
| 12 | void outputResult() | 输出函数 | / | void |
| 13 | int g_nextClock() | 完成下一时刻的动作 | / | int |
| 14 | MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) , ui(new Ui::MainWindow) | / | / | MainWindow |
| 15 | MainWindow::~MainWindow() | / | / | / |
| 16 | void MainWindow::on_updateRequestButton_clicked() | / | / | void |
| 17 | void MainWindow::on_createRequestButton_clicked() | / | / | void |
| 18 | void MainWindow::handleNewQuery(Query ret) | / | / | void |
| 19 | void MainWindow::on_updateClockButton_clicked() | / | / | void |
| 20 | bool MainWindow::eventFilter(QObject *watched, QEvent *event) | / | / | bool |
| 21 | add_query_window::add_query_window(QWidget *parent) | / | / | add_query_window |
| 22 | add_query_window::~add_query_window() | / | / | / |
| 23 | void add_query_window::on_buttonSubmit_clicked() | / | / | void |
| 24 | void add_query_window::on_buttonClear_clicked() | / | / | void |
| 25 | get_() | get函数族，取得相应的计算结果 | / | / |

4.4 函数调用图示及说明

图1 dispatch_core.cpp模块函数调用关系

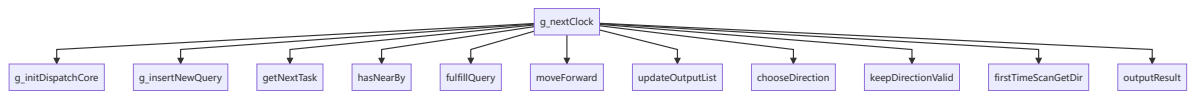
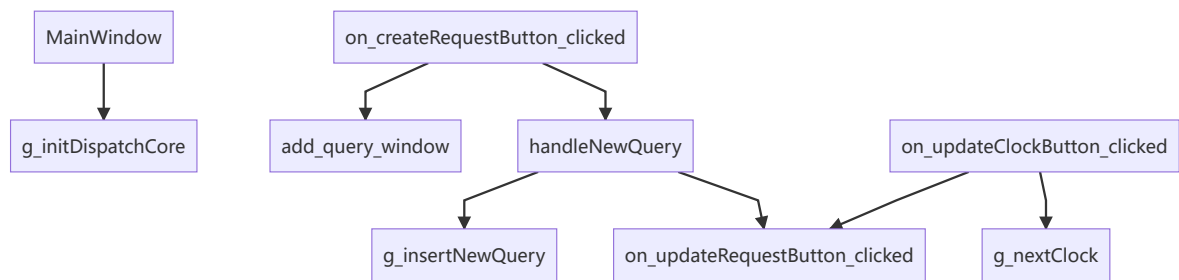


图2 mainwindow.cpp模块函数调用关系



各函数通过对箭头指向函数的调用，完成函数说明中对应的作用

5. 核心算法设计

