**《计算导论与程序设计课程设计》**

**实验报告**

**实验名称   《自动驾驶公交车调度系统》概要设计**

**班   级       2021211321**

**组   号    46**

**姓   名       郭栩源  胡栋康姜杨皓**

# 1．用户交互界面设计

## 1.1 输入输出交互方式

**文件：**

配置文件的名字为dict.dic。文件每行一个参数，格式为：参数 = 值（以#号开头的行是注释）。

**图形界面：**

输入：手动输入，每次输入先按“指令类型”按钮，再按“站点号”按钮。指令共5种。分别为end、clock、counterclockwise、clockwise 和target。

**输出：**

输出区，每个clock刷新一次，在输入指令后刷新一次，输出的格式如下：

TIME:秒数

position:0

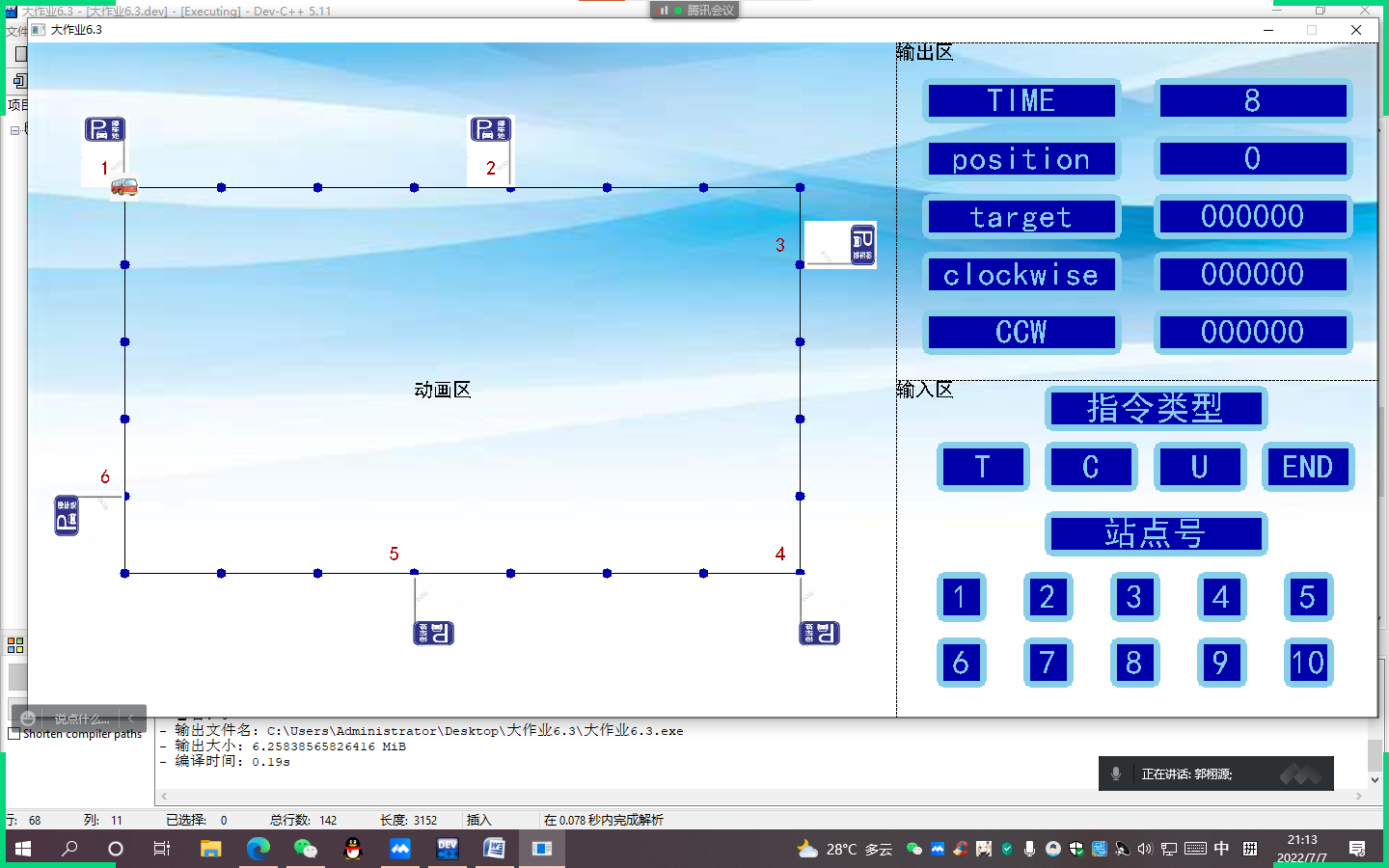
target: 0000000000

clockwise: 0000000000

counterclockwise: 0000000000

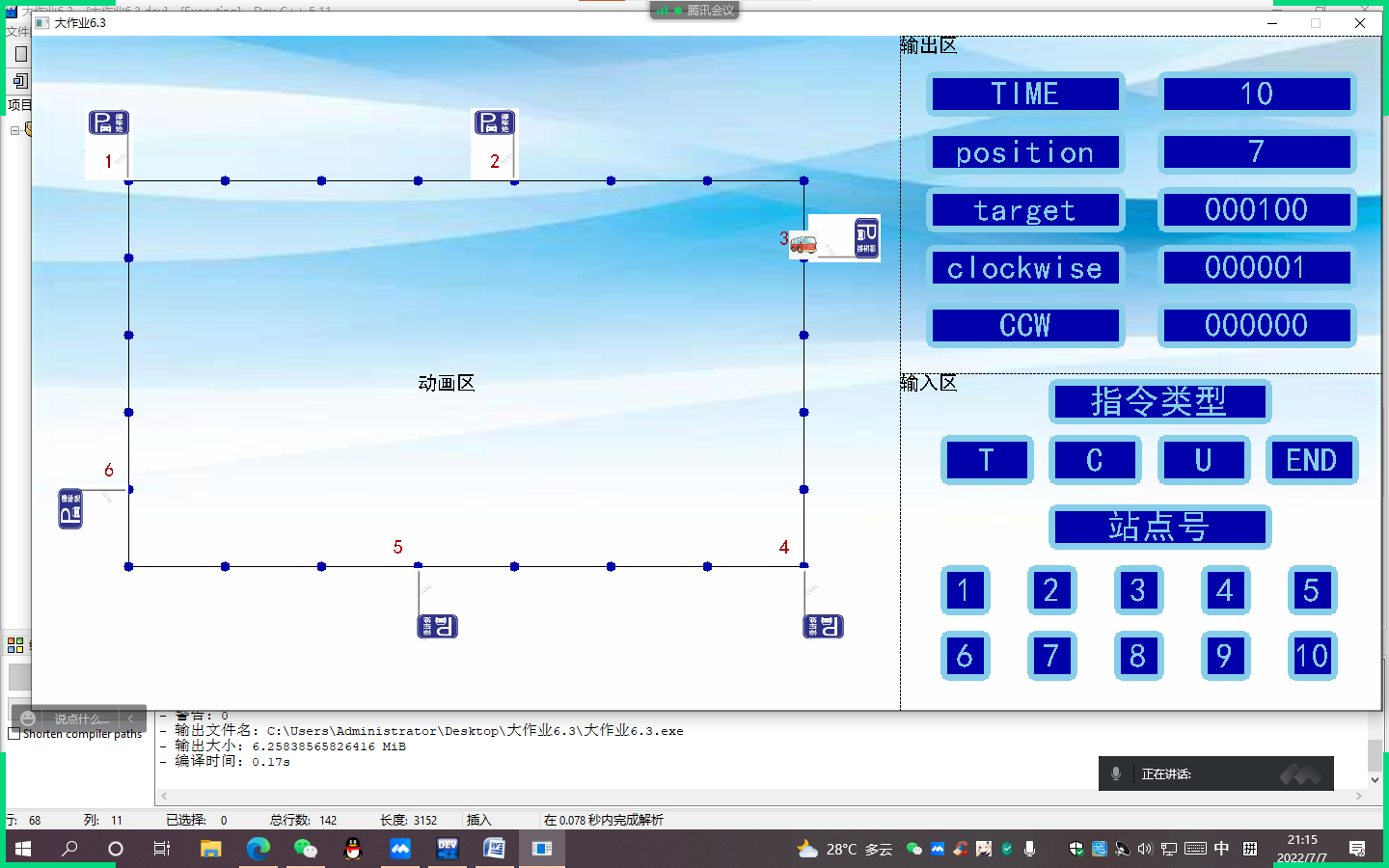
动画区，不断刷新，显示公交车当前所处位置。

## 1.2 操作元素和操作效果



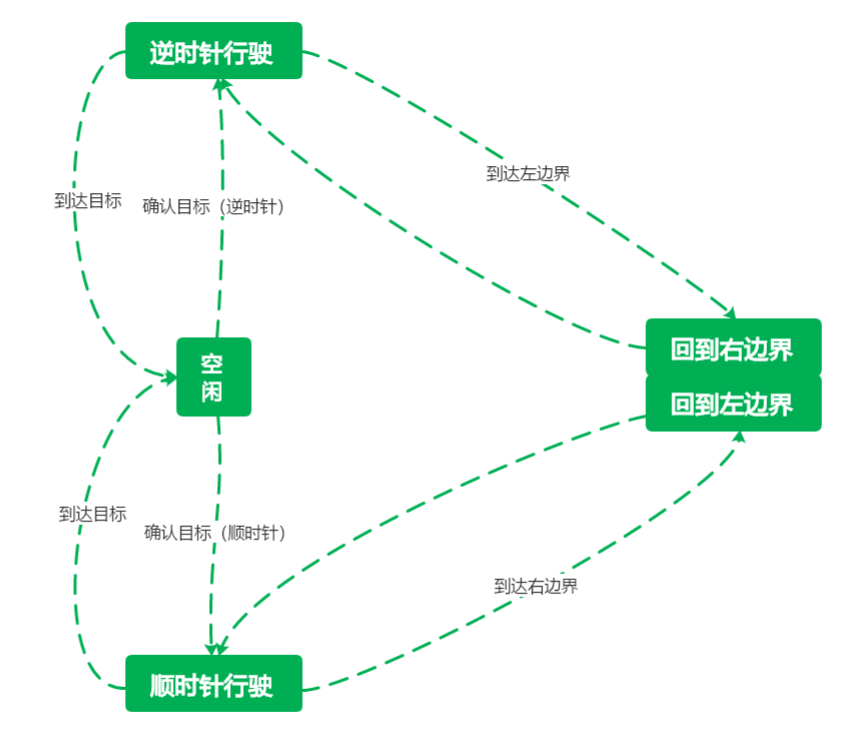
手动输入，每次输入先按“指令类型”按钮，再按“站点号”按钮。指令共5种。分别为end、clock、counterclockwise、clockwise 和target。

输入后，输出区动画区随之刷新，如下图：



# 2 有限状态自动机状态转换图

**2.1 状态图**



**2.2 状态说明**

空闲：已完成该站点请求，未接受到目标；

行驶：顺/逆时针行驶；

回左/右起点：在左（右）边界继续向左（右）行驶会回到右（左）起点；

# 3 高层数据结构设计

(包括：重要的数据常量定义、数据变量定义，即各模块要共享的数据类型和参数设计，相当于头文件内容，加文字描述)

**3.1全局常量/变量定义**

**3.1.1常量**

**当前使用策略**

#define FCFS 1

#define SSTF 2

#define SCAN 3

**近似无穷大**

#define INF 0x7f7f7f7f

**公交车状态**

#define CLCW 0

#define CCLCW 1

#define FREE 2

**3.1.2变量**

**策略**

int strategy;

**输入**

struct Input

{

char cmd[20];

int num;

};

struct Input input;

**时间**

int time;

**公交车**

struct Bus

{

int state;

int pos; //所处位置

int goal; //目标

int goal\_state; //记录当前目标对应的请求类型

double x,y; //像素坐标

};

**站台分布与站台状态[站点状态队列：用于输出站点状态（顺上/逆上/下的01串）]**

（此二变量用于生成/初始化站台信息）

struct Station

{

int pos; //初始化函数initStation通过参数totalStation和distance来初始化各站点坐标

int state[3]；//在调度模块中不断更新，state[0]为该站台顺上请求状态，0为无，1为有； //state[1]逆上,state[2]下。

} stations[10];

int totalStation = 5; //站台总数

int distance = 2; //站台间距

**站点像素坐标**

struct POSITION {

double x;

double y;

};

**策略调度模块：**

struct Queue {

int stationSubscript;

int requestType;

int No;

int requestTime;

};

struct Queue queue[31];

int totalNo;   //请求流水号

//目标站判定

int flagofPassingstation;

Int queueLength;

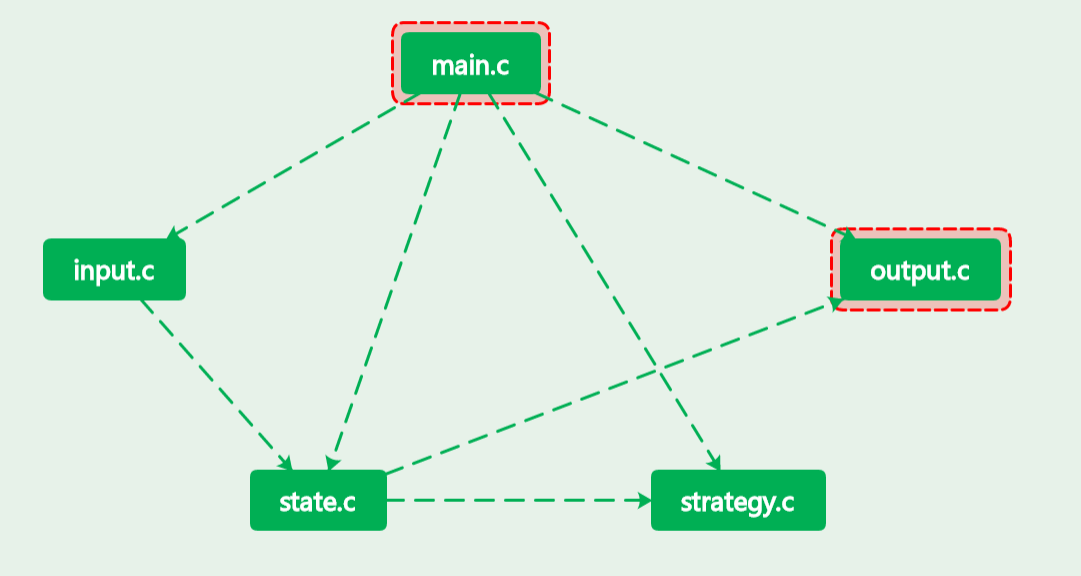
int busdirection;

int busderection = 0;//公交车行驶方向，0无方向 1顺 -1逆

# 4 系统模块划分

## 4.1 系统模块划分(说明共分成哪些程序模块，各模块功能概述)

模块划分思路说明。



**1. 模块名称：输入模块**

模块功能：完成对用户输入的线程的实现,根据输入更新用户请求数据结构。

**2. 模块名称：输出模块**

模块功能：完成格式化的输出。

**3. 模块名称：状态机模块**

模块功能：完成公交车状态，动作和站点状态的改变。

**4. 模块名称：策略模块**

模块功能：根据公交车状态，动作，站点状态和加入的请求来确定公交车当下的目标。

**5. 模块名称：主控模块**

模块功能：完成对各模块的调用。

6. **模块名称：动画模块**

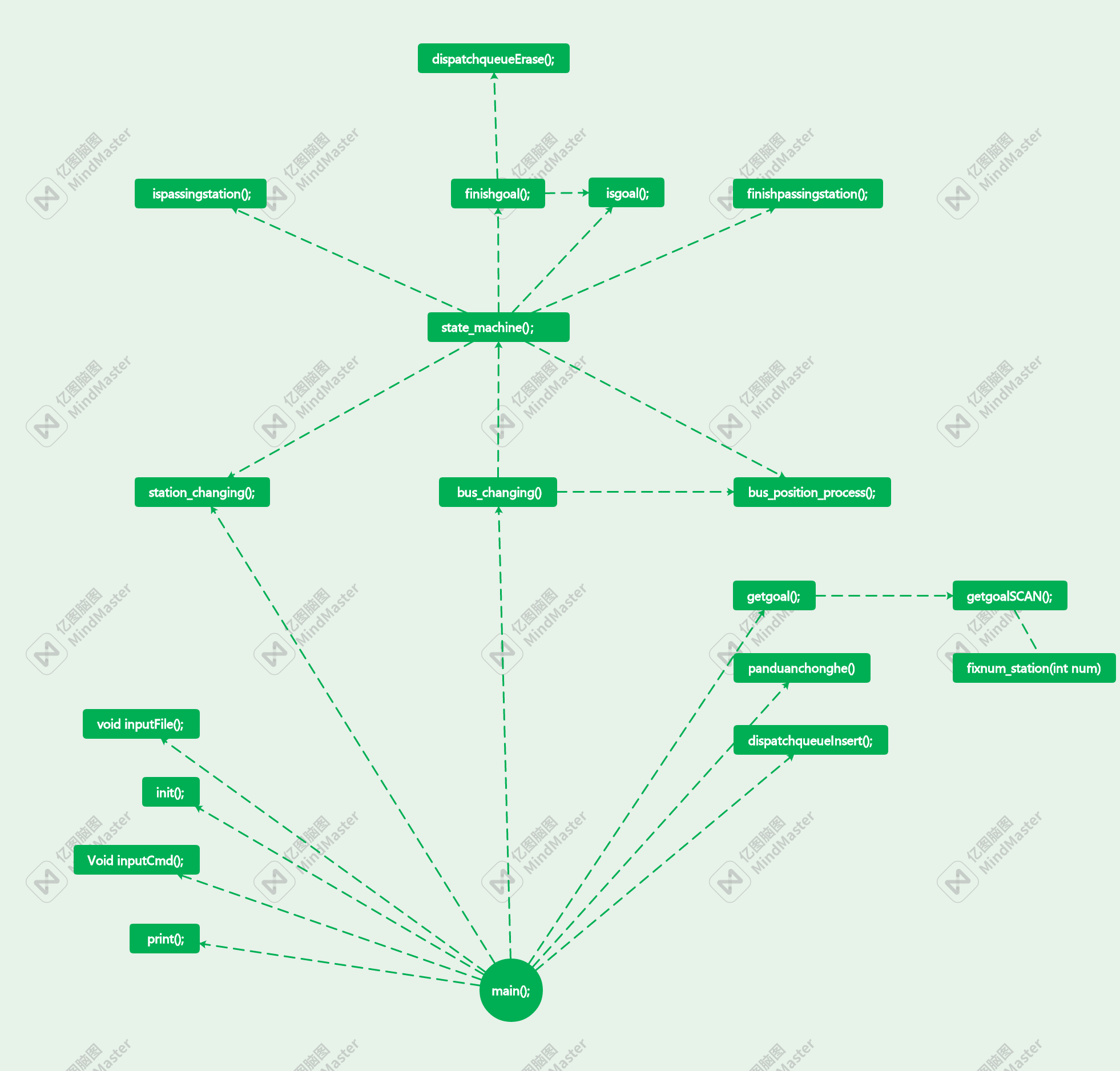
模块功能：实现图形化界面。

## 4.2各模块函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 | int main(); | 调用各函数，程序的中心枢纽 | 无 | 0 |
| 2 | void init(); | inputFile()调用后，被main()调用。  初始化time，bus，station[10]。 | 无 | 无 |
| 3 | void panduanchonghe() | 在策略三停靠时，判断是否有新的请求在本站 | 无 | 无 |
| 4 | void inputFile(); | 程序开始时被main()调用。  从文件中读取strategy，totalStation，distence | 无 | 无 |
| 5 | Void inputCmd(); | main()每次循环调用一次。  从标准输入流中读取input.cmd和input.num,并返回至主函数。 | 无 | 无 |
| 6 | void dispatchqueueInsert(); | 生成调度队列时被main()调用。  在调度队列中插入新请求。 | 无 | 无 |
| 7 | void dispatchqueueErase(); | 目标完成时调用。  删除已被完成的字队列。 | 无 | 无 |
| 8 | void getgoal(); | 获取目标时被main()调用。  从调度队列中获取目标。 | 无 | 无 |
| 9 | int isgoal(); | 判断公交车当前是否到达目标站 | 无 | 无 |
| 10 | void finishgoal(); | 目标完成时调用。  目标完成后完成善后工作。 | 无 | 无 |
| 11 | int ispassingstation(); | 判断站点是否为顺便站 | 无 | 返回是否为顺便站 |
| 12 | void finishpassingstation(); | 完成顺便站请求以及善后工作 | 无 | 无 |
| 13 | void getgoalSCAN(); | 运行策略三寻找目标站 | 无 | 无 |
| 14 | int fixnum\_station(int num) | 防止公交车站点出界 | 站点下标 | 站点下标 |
| 15 | void print(); | 按照格式输出相应内容。 | 无 | 无 |
| 16 | void bus\_changing() | 调用state\_machine();bus\_position\_process();改变公交车状态机 | 无 | 无 |
| 17 | bus\_position\_process(); | 防止公交车越界 | 无 | 无 |
| 18 | void state\_machine()； | 改变公交车的运行情况 | 无 | 无 |
| 19 | void station\_changing(); | 改变公交站台请求状态 | 无 | 无 |
| 20 | void drawAll(); | 刷新时完成所有图形化界面的绘制，调用其他部分绘制函数 | 无 | 无 |
| 21 | void button(int x, int y, int w, int h, const char\* text\_); | 完成一个按钮的绘制 | 按钮x，y坐标、按钮长宽 | 无 |
| 22 | void showinputarea(); | 完成输入区绘制 | 无 | 无 |
| 23 | void pathway\_graph(); | 针对不同的站台数和站台间距，绘制轨道 | 无 | 无 |
| 24 | void output\_graph(); | 完成输出区绘制 | 无 | 无 |
| 25 | void bus\_move(); | 计算公交车移动方向和距离，完成公交车的跑动 | 无 | 无 |
| 26 | void\* Input(void\*) | 输入线程函数 | 无 | 无 |
| 27 | void\* Output(void\*) | 输出线程函数 | 无 | 无 |

## 4.3 函数调用图示及说明

图示



说明：

1.main（）调用void inputFile();Void init ();完成地图，公交车的初始化工作。

2. void\* Input(void\*)先调用panduanchonghe()dispatchqueueInsert();getgoal();

再使

①getgoal()调用getgoalSCAN();

②getgoalSCAN();调用fixnum\_station(int num)

进而完成策略模块，使得公交车获得目标。

3. void\* Output(void\*)先调用bus\_changing()，station\_changing()；

再使

①bus\_changing()调用bus\_position\_process();state\_machine()；

②state\_machine()调用isgoal();finishgoal()；③ispassingstation();finishpassingstation();station\_changing()

④finishgoal();调用dispatchqueueErase();

进而完成公交车，公交车站状态和动作。

4. void\* Output(void\*)调用void print();完成时间，公交车，公交车站的状态输出。

# 5 高层算法设计

**（用伪代码、NS图或者自然语言描述清楚核心算法的程序设计思路）**

说明重要的控制策略算法思路；

说明输入输出的时间控制逻辑和文件记录生成等关键函数思路；

FCFS策略：

按照请求先后顺序存储请求，存储请求前进行去重处理。获取目标时直接区请求队列首个元素即可。完成目标时删除请求队列对首元素，并获取新请求。

SSTF策略：

按站台下标存储请求，请求与站台下标成映射关系，并以0和1分别代表无请求和有请求。获取目标时搜索整个数组，获取距离最近那个站点，距离相等则取流水号小的请求。完成目标时将相应请求下标下的1变成0即可消除请求，然后获取新的目标。处理顺便站时，不会更改目标站相关信息，只根据公交车当前所在位置和行驶方向判断是否能够完成顺便站请求，完成顺便站时也不会影响目标站相关信息。

策略三 SCAN策略获取目标过程：

在初始时公交车没有行驶方向，顺逆时针交替遍览地图寻找目标站，找到后即设为目标站。

在确定行驶方向后，顺时针行驶时，如果公交车位于两个站台中间，则搜寻当前站与目标站之间寻找请求，找到请求即更新为新的目标站。如果公交车位于站台上，则搜索地图向当前方向的一半寻找请求，找到后即设为新的请求；如果不能找到请求，则逆时针搜索地图的一半寻找请求，如果找到请求，则公交车应改变方向为逆时针，并逆时针行驶；如果仍然无法找到请求，则此时地图上无请求，公交车停止，恢复无行驶方向状态。公交车逆时针行驶时策略与顺时针行驶时基本相同。

**教师评语：**