**《程序设计课程设计》实验报告**

**实验名称 《课程设计-外卖派单模拟系统》<版本号:1.0>**

**班 级 2018211318**

**组 号 （按照网站上各组对应的小组序号）**

**姓 名 凌国瀚 张北辰 张开元**

# 1．用户界面设计

## 1.1 开始界面

运行程序后首先出现的界面，如图1-1

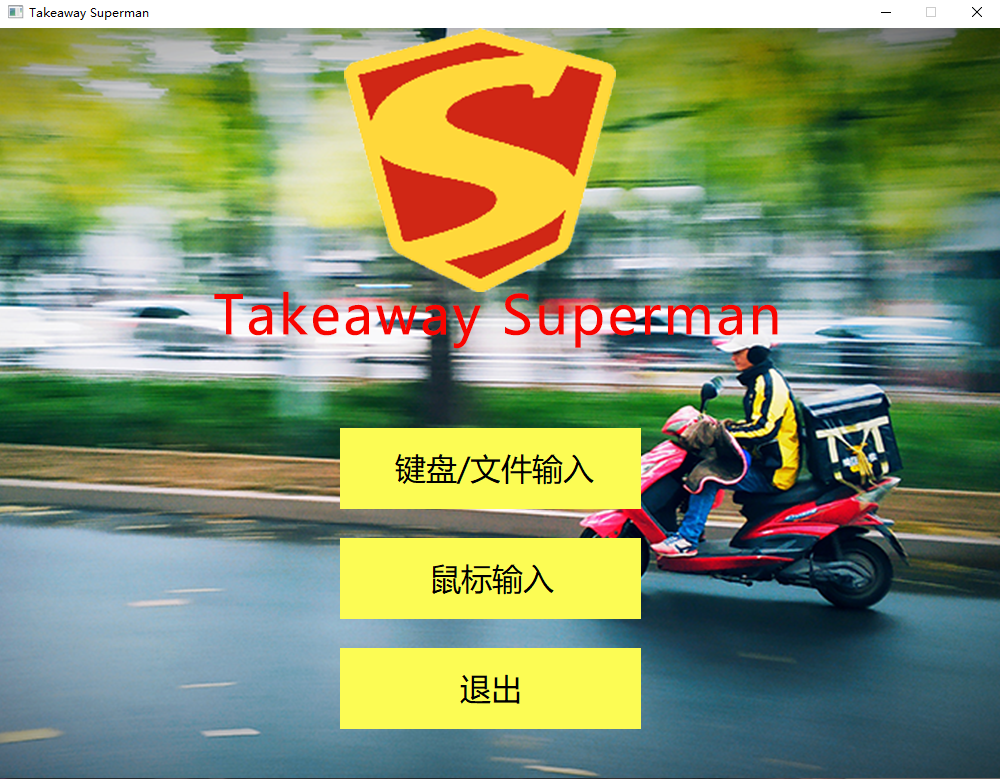


图1-1

该节目为开始界面，在logo与标题下方是三个按钮，按下对应按钮后，相应按钮会变成红色

按下第一个按钮（如图1-2），进入键盘/文件输入模式，检测doc文件夹内是否有标准文件名“sales.txt”,若有则自动进入订单强制文件输入方式。没有则在控制台中选择从键盘输入或是从鼠标输入（如图1-3）。（注：实际操作中，开始界面会被关闭）



图1-2



图1-3

按下第二个按钮（如图1-4），直接进入鼠标输入模式，运行图形界面鼠标输入

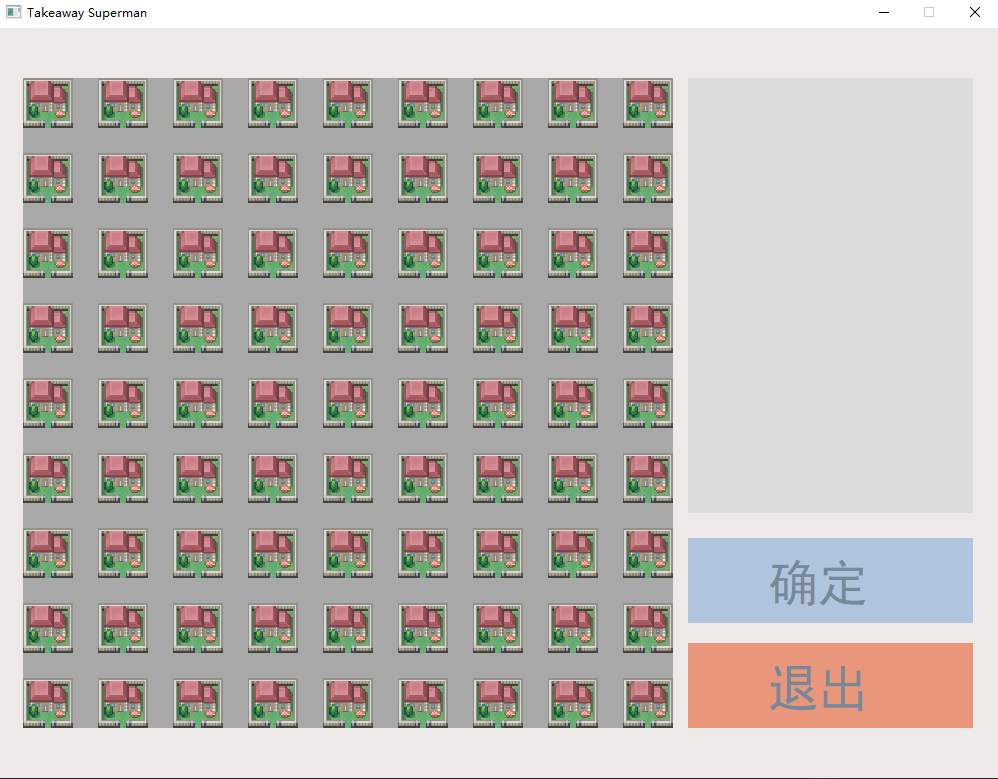


图1-3

按下第三个按钮，退出程序

## 1.2 图形（鼠标）操作界面

// TODO:

# 2. 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)

## 2.1全局常量定义

const int TIME\_UNIT = 2; // 时间单位为TIME\_UNIT秒

const int INIT\_MONEY = 1000; // 起始资金

const int HIRE\_MONEY = 300; // 招募一个人需要的资金

const int DEAD\_TIME = 3; // 超过DEAD\_TIME时间单位后，被视为拒单

const int FINE\_FIRST\_TIME = 30; // 超时时间，第一次超时罚款

const int FINE\_MONEY = 50; // 第一次超时罚款金额

const int FINE\_SECOND\_TIME = 60; // 超时时间，第二次超时倒闭

const int MONEY\_GAIN\_ONE = 10; // 完成一单获得的钱

const int MAP\_X = 17; // 地图X大小

const int MAP\_Y = 17; // 地图Y大小

const int COMPANY\_X = 7; //公司所在X大小

const int COMPANY\_Y = 8; //公司所在Y大小

## 2.2 全局变量定义

/\*

为区分全局常量，全局变量命名如下所示

\*/

int CompanyMoney = INIT\_MONEY; // 当前公司有多少钱

int CompanyRiderCount = 0; // 公司骑手数量

int Time = -1; // 当前时间，即Time个时间单位

int CompanyOrderSum = 0; // 公司接单数

int CompanyOrderFinish = 0; // 公司完成单数

int CompanyOrderOverTime = 0; // 公司超时单数

OrderList \*AllOrderLog; // 全部订单记录(含有头结点)

OrderList \*Buffer; // 缓冲区(含有头结点)

RiderList \*AllRiderLog; // 全部骑手记录(含有头结点)

int Map[17][17]; // 输出用，0为道路，1为普通房屋，2为餐厅，3为宿舍，4为骑手

## 2.3 数据结构定义

/\*

全局结构体

注意：本项目所有链表都具有空头结点！！！

\*/

typedef struct \_Order // 订单

{

int id; // 订单ID

int begin\_time; // 起始时间戳

int warn\_time; // 超时时间戳

int end\_time; // 截止时间戳

int rest\_x, rest\_y; // 餐厅位置

int cust\_x, cust\_y; // 顾客位置

int status; // 订单状态 0未接单，1取餐，2送餐，3完成

} Order;

typedef struct \_OrderList // 订单链表

{

struct \_OrderList \*Pre\_order; // 上一个订单

struct \_OrderList \*Nxt\_order; // 下一个订单

Order \*Cur\_order; // 当前订单

} OrderList;

typedef struct \_Rider // 骑手

{

int id; // 骑手ID

int rider\_x, rider\_y; // 骑手当前位置

OrderList \*Bag; // 骑手背包订单

//Order \*Cur\_order; // 骑手当前订单

} Rider;

typedef struct \_RiderList // 骑手列表(便于遍历)

{

struct \_RiderList \*Pre\_rider; // 上一个骑手

struct \_RiderList \*Nxt\_rider; // 下一个骑手

Rider \*Cur\_rider; // 当前骑手

} RiderList;

# 3. 系统模块划分

## 3.1 系统模块结构图

本程序模块结构图如图3-1

模块功能如下：

1. **模块名称：main.cpp**

模块功能：程序主函数与入口，有创建开始界面，运行文件/控制台输入

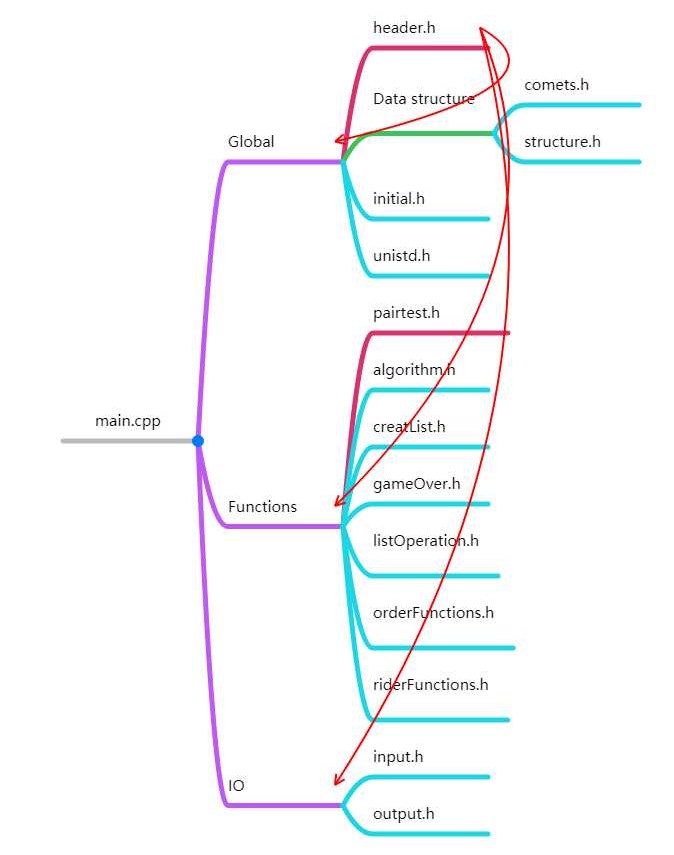


图3-1

## 4.2各模块函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 |  | 何时何条件被何模块调用？  完成什么功能？ |  |  |
| 2 |  |  |  |  |

## 4.3 函数调用图示及说明



图4-2 函数调用关系图示例

解释说明：如图中所示，函数1通过对函数2的调用，完成….

# 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；

**教师评语：**