

人工智能与自动化学院

C语言课程设计

红蓝军对战系统

需求分析与系统设计报告

专业班级：自动化1904

小组成员：刘云笛 学号 U201914665

陈旭桐 学号 U201914650

指导老师：周纯杰、何顶新、彭刚、周凯波、桑农、左峥嵘、高常鑫、汪国有、陈忠

上交时间：2020年

目录

1 引言3

1.1 编写目的3

1.2 背景3

1.3 参考资料3

2 可行性分析4

2.1 目标4

2.2 要求4

2.3 用户特点5

3 需求分析5

3.1 总体系统架构 6

3.2 数据流分析7

3.3 接口设计8

3.4 算法模块8

3.5 界面模块9

3.6 功能模块10

4 运行环境的规定11

4.1 设备12

4.2 支持软件12

4.1 接口12

4.4 控制12

5 界面初步设计13

5.1 初始界面13

（1） 菜单界面14

（2） 第二菜单界面15

（3） 固时模拟地图界面16

（4） 智能模拟界面17

（5） 路况分析表格18

6 数据结构设计18

1. 汽车参数18
2. 红绿灯参数18
3. 十字路口红绿灯组参数19
4. 丁字路口红绿灯组参数19
5. 汽车组参数19

7 程序模拟流程图20

（1） 系统流程图20

（2） 汽车模拟流程图21

（3） 汽车行驶流程图22

（4） 汽车组流程图23

（5） 汽车转向流程图24

（6） 固时模拟红绿灯控制流程图25

（7） 智能模拟单个红绿灯流程图26

8 组员分工及时间安排 27

8.1 组员分工表27

8.2 时间安排表27

1 引言

1.1 编写目的

在经过为期近一个月的系统开发，且了解了市面上多款军事策略模拟游戏和军演模拟软件之后，我们编写了本报告，旨在确定软件开发目的和大局方针，以及对我们已经完成的工作进行总结和后续更多开发的总体规划。

本报告中，将会详细提到本系统开发目的和背景、用户的需求分析、系统开发的可行性分析、系统功能设计，还包括本系统的具体信息，运行环境要求、界面设计、数据结构设计、总体程序架构、功能实现、流程结构、以及组员的分工安排和时间安排。本报告将会给出实现功能的函数，大部分源代码也将在本报告中给出。

1.2 背景

在如今的国际局势下，军事人才的培养是必须且亟需的，但是很难要求青年人主动系统地学习军事知识、培养军事素养。

经由调查，军事模拟以及策略战斗类的软件现在广受欢迎。开发一款快节奏、高仿真的系统，在吸引用户使用的同时，也能起到一定程度的军事素养提高作用。相较传统的象棋和军棋，这类软件具有精美的画面以及更加复杂的机制，更加贴近现实或具有天马行空的想象力。与棋类的抽象意识相比，显然此类软件更加友善，能直观有效地提升用户对现代战争的了解，提升自己的运筹能力。除此之外，用户能在使用的过程中，体验博弈的快感，还可以在与朋友对战时增进友谊。这也是此类软件吸引用户的主要方式。

现在中国的经济飞速发展，人们的生活水平在不断提高，科技的进步与城市化为人们带来了更快地生活节奏，对于各种软件的需求也在发生改变。人们更加青睐内容简单、趣味性强、节奏快、可以随时离开的软件。哪款软件能更好的利用用户的碎片化时间，哪款软件就能获得成功。除此之外，软件编写技术快速进步，使编写出一款更加具有趣味性的模拟军事对战软件变得日益简单。图形化界面编程的不断进步，也可以让程序界面可以变得更加精美、吸引人。

我们参考了网络上几款广受好评的软件，结合它们各自的优点以及人们日益变换的需求，设计出了本软件。本软件具有能够吸引人的精美画面，加速战斗进程的机制设定，可以随时保存退出的回合制设计，能够快速上手的简单机制，以及巨大的可能决策空间。希望能够给用户提供一个良好的使用体验。

1.3 参考资料

王士元. C高级实用程序设计. 北京: 清华大学出版社. 1996

周纯杰，刘正林等. 标准C语言程序及应用. 武汉: 华中科技大学出版社. 2008

六边形网格设计方法

<https://www.redblobgames.com/grids/hexagons/>

CSDN最短路径算法 https://blog.csdn.net/weixin\_39956356/article/details/80620667

2 可行性分析

2.1 目标

本系统旨在给用户提供一个进行模拟军事对战的平台，提供图形化界面，实时操作兵种进行指挥决策，本系统尽量模拟真实战场的对战场景，地形、兵种克制、天气、排兵布阵，各种因素都会影响到用户的决定和战斗的结果，在使用过程中，用户可以亲自体会到作为一名指挥官，在紧张又激烈的战场上如履薄冰地进行每一次决策，调兵遣将，排兵布阵，争夺并合理分配使用资源，以彻底摧毁敌方大本营为战斗胜利目标。

本系统尽量模拟真实战场的对战情景，给用户真实体验的同时，将现代战争复杂难清的战斗优化为可以轻松掌控的模拟战。通过本系统，一方面可以锻炼军事思维能力、培养一定的军事策略水平，另一方面可以给现代战争作为一定的参考，同时，对于军事策略爱好者来说，也可以作为一种不错的休闲娱乐软件。

本产品可实现红蓝军模拟对战的基本功能。产品用户可以在进入的主界面进行登录注册，并可以根据设置的密保问题重置密码，设置的游客登录模式可以让用户快速上手体验。本产品对用户的密码作了良好的加密保护，仅能快速检验密码正确性而不能计算真正密码。

进入主菜单后，用户可以选择进入之前的存档继续对战，或者建立新的存档。本产品具有单人模式（人机对抗），以及两位用户同屏对战的双人模式，可以在不同的模式中体验不同的乐趣。对于首次使用本产品的用户，我们还设置了教学模式，方便新用户快速熟悉各个界面和操作，还可以深入了解各项对战机制。

进入对战界面，可以在此界面进行战斗的所有功能。本产品一方面提供了精美的画质，制作了较为美观的动画效果，又通过优良的算法，尽可能减少了对系统配置的要求以及更新屏幕造成的卡顿，以提升用户使用体验。地图采用六边形网格，为更加复杂多变的移动方式提供可能，采用迪杰斯特拉最短路算法实现每个兵种快速准确地移动。地图上设有多种地形，兵种也具有丰富的多样性，还有争夺资源以获取优势的机制，既能让用户快速上手，又能创造出一个复杂多变的战场。回合制的设定让用户可以体验纯粹的策略比拼，不受时间的限制，让用户可以更加仔细地分析场面形势，降低了使用的门槛。还设置了快速存档的功能，可以实现快速保存退出。

2.2 要求

2.2.1 基本要求

实现红蓝军模拟对战，提升用户的军事素养和运筹能力。

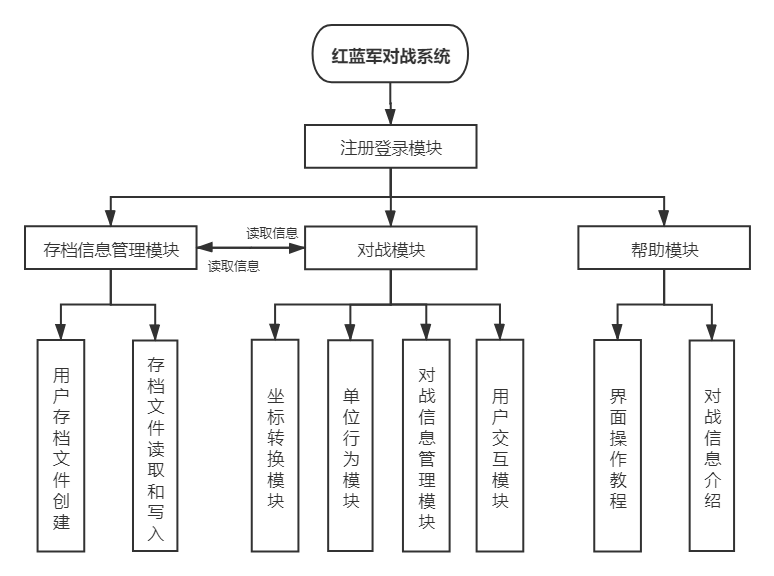
2.2.2 具体要求

1. 使用图形化界面并实现鼠标和键盘操作。
2. 实现各单位的移动、攻击等操作
3. 屏幕上实时显示战场变化

2.3用户特点

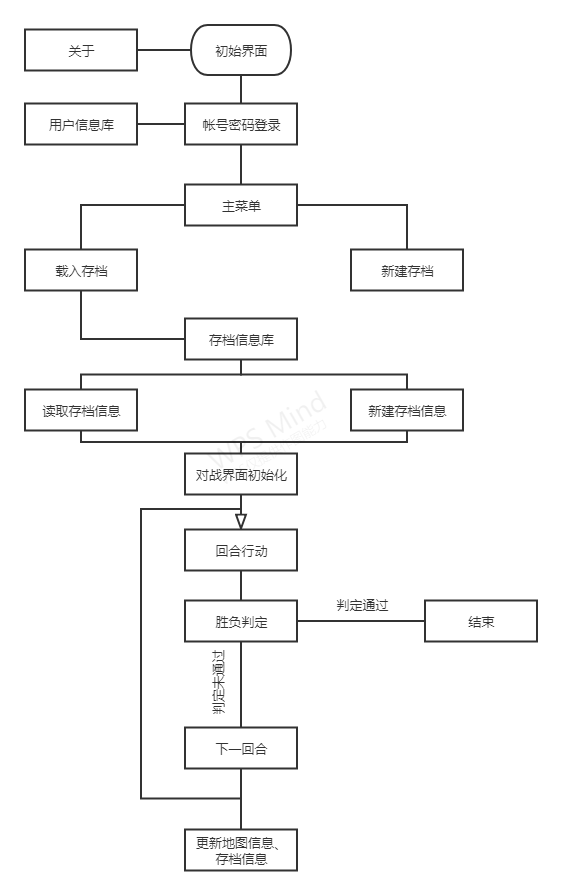
本产品适用于任何对军事、运筹感兴趣的用户，上手简单，只需观看简短的教学即可快速上手，程序中也会有大量的提示引导用户操作。对于维护人员需要一定C语言编程基础，但设有报错功能，可以方便维护人员快速发现问题所在。

3 需求分析

3.1 总体系统架构

3.2 数据流分析

3.3 接口设计



3.4 算法模块

路况信息查看模块

汽车信息查看模块

鼠标点击/键盘输入触发

鼠标点击车辆

对链表中车辆总体信息进行计算并输出结果

从车辆链表中查询车辆信息

汽车行为控制模块

汽车信息查看模块

读取车道对应红绿灯信息

读取车道对应红绿灯信息

顺序查找算法

顺序查找算法

顺序查找算法

图4 算法设计模块图

3.5 界面模块

（1）用户打开该产品首先进入加载页面，若干秒后进入主菜单界面；

（2）首界面可进行鼠标操作，当鼠标移至图标位置，图标下会显示对应的文字：

1）开始模拟，点击可进入模拟界面，进行下一步操作；

2）导航模式，点击可进入模拟导航模式；

3）关于我们，点击可查看产品介绍及其研发人员；

4）退出程序，点击则退出系统；

（3）进入开始模拟界面后，当鼠标移至图标处，图标会等比例放大，起到突出效果：

a)固时模拟；

b)智能模拟；

c)返回菜单；

（4）进入导航模拟后，进入地图选择起始地和目的地，完成导航路线规划以及时间预估。

（5）点击退出程序，则跳转“再见”界面，若干秒后退出产品。

3.6 功能模块

（1） 交通信号灯函数

void lightcontrol( struct \*lampsten, sturct \*lampsding);

通过时间函数控制信号灯的颜色变化(总共四个红绿灯配合变化);

void if5s(CARLIST \*cars, struct \*lampsten, sturct \*lampsding);

判断五秒内是否持续有车通过停车线,并执行对应操作。

void if3s(CARLIST \*cars, struct \*lampsten, sturct \*lampsding);

判断三秒内是否没有车通过停车线,并执行对应操作

void initlamps( struct \*lampsten, sturct \*lampsding);

初始化红绿灯

（2） 车辆控制函数

void control\_car(CARLIST \*cars);

1)judge\_lukou(struct car \*CAR));//在路口进行转弯操作

2)judge\_alarmline(struct car \*CAR); //判断车辆警戒线警报情况并执行对应操作

3)judgelamp(struct car \*CAR, struct lampsten laps10[], sturct \*lampsding ding); //判断路面路灯情况并执行相关操作

4)judgecardistance(CARLIST \*cars);//判断前后车距并执行相关 操作

5)control\_speed(struct car \*CAR);//控制车速

6)ifturn //通过随机函数来决定车辆在路口的转向,直行还是左转.

7)ifout //车辆是否驶出画面函数,由于开始控制了画面中车辆的数量,若有车辆驶出画面,须有车辆进入画面作为补充,以维持数量恒定.

（3） 车辆组链表控制函数

1) CARLIST \*initcars(int n);//初始化车辆链表

2) void initcar(struct car \*CAR);//初始化车辆

3) void addcar( CARLIST \*CAR);//增加车辆

4) void carout( CARLIST \*CAR);//减少车辆

（4）人工控制函数

1） void num\_put(int \*flag);//左右键增加减少车辆数

2） void input(int \*b, int y);//输入车辆数控制地图中车辆总数；

3） void simple\_put(struct car \* CAR, int n1, int n2, int n3, int n4, int \*wait, int \*S)；//计算东西南北各方向的车辆流量

4） void analysis(struct car \* CAR, int n1, int n2, int n3, int n4, int \*wait, int \*S,int CASE\_1,int \*s)；//计算优化率和等待时长

（5） 导航函数

1） void innovagation(int \*flag)；//通过人工选择系统生成规划路线；

2） void innovagation2(int \* i)；//计算路线预计所需时间；

4 运行环境规定

4.1 设备

本系统是对“智能交通系统”的基本模拟，因此，对此系统的运行环境将是在计算机上面进 行，即在电脑上模拟实现，系统运行的设备需求如下:

（1) 一台 586 以上的微机及兼容

（2) 内存 16MB 及其以上

（3) 彩色显示器一台

4.2 支持软件

根据“智能交通模拟系统”设计环境要求，即在 DOS 环境下，用 C 语言编译实现，使用 TC 或 BC 开发软件，因此，此模拟系统可在绝大多数现形计算机系统上运行，包括最常见的 Windows XP 操作系统以及 Windows 7 32 位机上运行，但由于兼容性问题，此“智能交通模拟系统”在含 windows 7 64 位操作系统的计算机上运行可能出现意想不到的问题或根本不能运 行，因而，不推荐在包含 64 位操作系统的计算机上测试本系统。

4.3 接口

（1) 用户接口:本系统采用一般性架构，所有界面使用简易风格。

（2) 硬件接口:服务器端建议使用专用服务器。

（3) 软件接口:无特殊要求。

（4） 通信接口:无特殊要求。

4.4 控制

本系统通过鼠标直接进行操控。用户将鼠标移至需要操作的功能区点击，进行相应的操作，操作进行完后，点击相应位置退出系统。控制信号来源于用户鼠标的位置，以及用户使 用鼠标进行的操作。 当用户进入不同模式可根据页面提示进行键盘操作，如输入车辆数量，暂停，返回，增加车辆数等。

5 界面初步设计

5.1 初始界面

（1） 第一菜单界面：

用户进入该页面可选择他想执行的下一步操作：“开始模拟”、“导航模式”、“关于我们”、“退出程序”。

智能交通模拟系统

+logo

退出模拟

智能模拟

关于我们

开始模拟

图5 第一菜单页面

当鼠标移至相应图标时会显示相应的文字。

（2） 第二菜单界面：

智能交通小车logo

返回菜单

智能模拟

固时模拟

2

图6 第二菜单页面

当鼠标移动至相应位置，图标会等比例放大。

（3） 地图界面：

用户可在模拟开始前先选定一个地图中车辆出现的总数，开始模拟整个地图的实时交通状况，在模拟过程中可通过键盘实时增加或减少车辆数，或者暂停来观察静态画面。

建筑物

建筑物

建筑物

红绿灯

红绿灯

建筑物

建筑物

建筑物

红绿灯

红绿灯

建筑物

建筑物

图7 固时模拟页面

满足基本要求：三个十字路口，一个丁字路口。

（4） 智能模拟模式：

用户可控制东西南北四个方向的车辆数来模拟一个十字路口的交通状况，系统将实时监测每条道路不同车辆经过的数目，也可以计算其交通优化率。

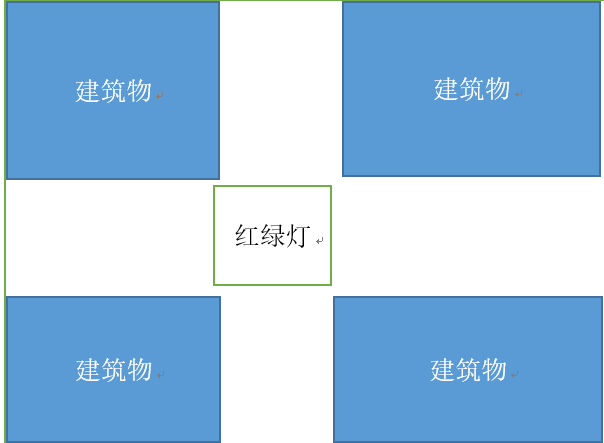


图8 智能模拟页面

(5) 路况分析表格：

|  |  |  |
| --- | --- | --- |
| 车辆总数 |  | 黑车 白车 红车 |
| 东西车流量 | 东方向 | 西方向 |
| 南北车流量 | 南方向 | 北方向 |

表1 路况分析表格

6 数据结构设计

1. 汽车参数

struct car

{

int x; //汽车中心当前x坐标

int y; //汽车中心当前y坐标

unsigned int color; //汽车颜色

int angle; //汽车当前角度

int alarmcar; //警戒距离警报

int alarmline; //警戒线警报

int alarmlamp; //红灯警报

int lane; //汽车所在车道

int street; //汽车所在道路

int speed; //汽车当前速度

int nextlane; //汽车准备前往的道路以及车道

int tors; //汽车左转还是直行

int bx\_by; //车尾的位置

int flag; //用于标志车子是否驶出桌面

};

（2）红绿灯参数

struct lamp

{

int x; //红绿灯中心所在x坐标

int y; //红绿灯中心所在y坐标

unsigned int color; //红绿灯亮灯颜色

int street; //红绿灯所在道路

int counttime; //红绿灯计时器

};

（3） 十字路口红绿灯组参数

struct lampsten

{

struct lamp lampten[4];

int x;

int y;

};

（4） 丁字路口红绿灯组参数

struct lampsding

{

struct lamp lampding[3];

int x;

int y;

};

（5） 汽车组参数

typedef struct carlist{

struct car CCAR;

struct carlist \*next;

struct carlist \*forward;

}CARLIST;

7 程序流程图

（1）系统流程图

图9 系统流程图

（2） 汽车模拟流程图

图10 汽车模拟流程图

（3）

A

Y

AA

是否行驶出区域？

速度+5/s，直至上限

正常行驶

N

离前车距离小于一个车位？

AB

Y

N

离行驶方向路口停车线距离小于一个车位？

Y

AC

N

图11 汽车行驶流程图

新增一个初始化的汽车节点到链表中

把该车节点从链表断开，释放内存

AA

车速-10/s，直至与前车车速相同

AB

保持当前车速行驶

车速变为与前车相同

Y

前车车速减小？

N

N

与前车距离＞1.5车位？

正常行驶

Y

N

车速变为与前车相同

Y

车速<0?

图12 汽车组流程图

读取汽车所在路段与车道

AC

执行左转操作

N

汽车直行？

Y

车速变为与前车相同

N

对应的红绿灯是绿色?

Y

执行右转操作

Y

在右转车道?

N

执行直行操作

图13 汽车转向流程图

B1

图14 固时模拟红绿灯控制流程图（以一个路口红绿灯为例）

B2

Y

Y

计时器归零，红绿灯变黄

N

初始持续时间小于现持续时间？

初始持续时间-3s，计时器归零

初始持续时间小于6s？

N

Y

三秒内无车通过停车线？

N

Y

Y

N

初始持续时间大于20s？

初始持续时间+3s，计时器归零

五秒内持续有车通过停车线？

红绿灯变为绿色，初始持续时间12s

图15 智能模拟单个红绿灯控制流程图

8 组员分工及时间安排

8.1 组员分工

|  |  |  |
| --- | --- | --- |
| 姓名 | 分工 | 备注 |
| 谢浩阳 | 图形动画模块；智能模拟模块；链表操作模块 |  |
| 柳登攀 | 界面模块；固时模拟模块； |  |

表二 组员分工表

8.2 时间安排

