<201421130610>小组 <高级模拟小火车调度>概要设计

版本号：CTS-Design-1.2.2

编制时间：2015年5月4日

修改时间：2015年5月14日

编制人员：李俊宁、聂艳、董文苑

⒈用户界面设计

1.1 文件方式

A. [输入文件] CTS.dat

地图长 地图宽

R 起点x 起点y

转折点x 转折点y 方向

…（最后一个点为终点）

-1//-1//-1

T 发车时间 运行速度

…（循环R… T…）

E

[注] CTS.dat可以由Edit Mode通过图形界面绘制（绘制规则 见附录1）

B. [输入文件] CTS.log

T=当前时间 S=当前策略

Train=火车序号 位置x 位置y 运行状态

…

…

1.2 控制台方式

A. 输入

空格键 切换不同策略

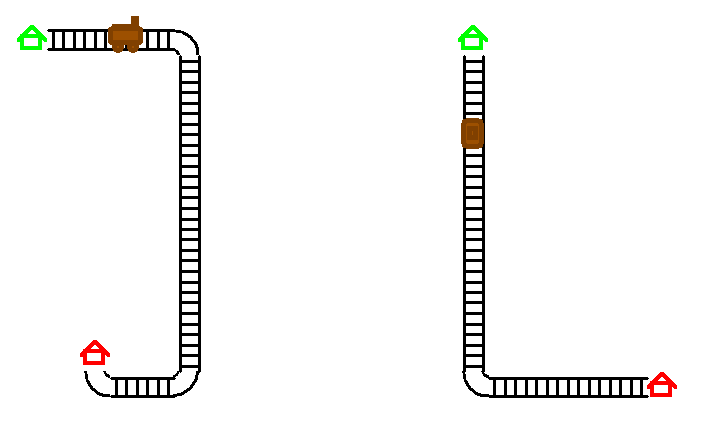
Q/Esc 退出程序

B. 输出

与输出文件CTS.log格式相同

1.3 动画方式（画出图形界面）,可以在作动画版本时更新和补充.

[图片]



[界面操作说明]：

1. 输入操作功能点：

空格键 切换不同策略

Q/Esc 退出程序

1. 输出设计：

定时在窗口绘制火车运行动画，显示所有轨道，车站，火车位置

⒉有限状态自动机状态转换图

* 1. [一级状态图]

时间大于发车时间 使用的轨道未被占用

运行

发车

未发车

将使用的轨道被占用

到达终点站

使用的轨道未被占用

将使用的轨道已被占用

到站

停车

* 1. [二级状态图]

线路为环路

到站

未发车

总时间大于发车时间

到达终点车站

由轨道计算位置，速度

将使用的轨道部分被占用

将使用的轨道全部被占用

将使用的轨道未被占用

停车

运行

制动

⒊高层数据结构设计

3.1常量定义

#define WORLDWIDTH 32 //地图宽度

#define WORLDHEIGHT 18 //地图长度

#define FPS 30 //绘图时钟

#define COMPUTETICK 1000 //计算时钟

#define CSTRATEGY 2 //策略数目

#define TS\_NDP 0 //未发车（火车状态）

#define TS\_ARR 1 //到达（火车状态）

#define TS\_SUS 2 //挂起（火车状态）

#define TS\_RUN 3 //运行（火车状态）

#define TS\_BRK 4 //刹车（火车状态）

enum {NORTH = 1, SOUTH = 2, EAST = 4, WEST = 8}; //方向

3.2 全局变量定义

int g\_cxMap, g\_cyMap; //标记绘图区分辨率 在Main.c定义 UI.c使用

SignalList \*g\_lsgHead; //控制信号灯资源 在Element.c定义 Basic.c使用

3.3 基本数据结构

struct ThreadCtrl; //用于线程通信结构

enum Direction; //方向

struct Route; //线路

struct Railway; //铁轨

struct Train; //火车

struct Signal; //信号

struct RouteList; //线路容器链表

struct RailwayList; //铁轨容器链表

struct TrainList; //火车容器链表

struct SignalList; //信号容器链表

⒋系统模块划分

4.1 系统模块划分

CUI

Main

Basic

Element

File

命令行版本

GUI

GUI

Main

Basic

Element

图形界面版本

File

**1. 模块名称CUIMain.c**

模块功能简要描述：控制台模式主函数，接收输入

**2. 模块名称 GUIMain.c**

模块功能简要描述：窗口模式主函数，创建接收输入

**3. 模块名称 Basic.c**

模块功能简要描述：初始化，运算-显示线程控制，控制台输入函数

**4. 模块名称 Element.c**

模块功能简要描述：管理及控制线路，铁路，火车，方向等基本元素

**5. 模块名称 File.c**

模块功能简要描述：地图文件写入及读取，运行日志写入

**6. 模块名称 GUI.c**

模块功能简要描述：图形化界面绘制

4.2各模块接口说明

**1. 模块接口Scales.h**

声明最基本常量，如地图大小，策略数目，绘图-计算时钟

**2. 模块接口 Basic.h**

定义线程通信结构，声明初始化函数，计算-绘图线程，错误提示，控制台输入

**3. 模块接口 Element.h**

定义基本元素，声明控制管理元素的函数

**4. 模块接口 File.h**

声明地图文件写入及读取，运行日志写入

**5. 模块接口 GUI.h**

声明图形化界面绘制函数

4.3各模块函数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 模块文件 | 模块说明 | 模块包含的函数名 | 函数功能概要描述 |
| CUIMain.c | 控制台主函数 | main | 控制台主函数 |
| ConsoleInput | 控制台输入 |
| GUIMain.c | 图形化主函数 | WinMain | 图形化主函数 |
| WndProc | 窗口过程函数（处理消息） |
| Basic.c | 基本函数 | Init | 初始化 |
| Compute | 计算线程函数 |
| Display | 显示线程函数 |
| Err | 错误提示 |
| File.c | 文件输入输出 | ReadConf | 读入地图配置 |
| WriteConf | 写入地图配置 |
| WriteLog | 写入运行日志 |
| Element.c | 基本元素控制 | NewRoute | 新建线路 |
| NewRailway | 新建铁轨 |
| NewTrain | 新建火车 |
| NewSignal | 新建信号 |
| AddRoute | 添加线路到链表 |
| AddRailway | 添加铁轨到链表 |
| AddTrain | 添加火车到链表 |
| AddSignal | 添加信号到链表 |
| FindRoute | 通过id找到线路 |
| FindTrain | 通过id找到火车 |
| FreeRouteList | 释放线路资源 |
| FreeTrainList | 释放火车资源 |
| FreeSignalList | 释放信号资源 |
| GenerateSignal | 初始生成信号 |
| MoveTrain | 移动火车 |
| CalcSignal | 计算信号 |
| GUI.c | 图形化界面 | 未确定 | 稍后版本完成 |

4.4 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能详细描述 | 参数说明 | 返回值说明 |
| 1 | NewRoute | 新建线路 | CTSID id 路线ID  xBeg x坐标  yBeg y坐标 | 返回Route的地址 |
| 2 | NewRailway | 新建铁轨 | x, x坐标  y, y坐标  Direction diTo 方向 | 返回Railway的地址 |
| 3 | NewTrain | 新建火车 | CTSID id 火车ID  begtime 启动时间  speed 速度  \*pRoute 指向线路的  指针 | 返回Train的地址 |
| 4 | NewSignal | 新建信号 | xBeg, 起始x  yBeg, 起始y  xEnd, 终止x  yEnd, 终止y | 返回Signal的地址 |
| 5 | AddRoute | 添加线路到链表 | RouteList \*\*plHead  Route \*pNew | 返回0或1，是否有新的线路生成 |
| 6 | AddRailway | 添加铁轨到链表 | Railwaylist \*\*plHead,  Railway \*pNew | 返回0或1，是否有新的铁轨生成 |
| 7 | AddTrain | 添加火车到链表 | Trainlist \*\*plHead,  Train \*pNew | 返回0或1，是否有新的火车生成 |
| 8 | AddSignal | 添加信号到链表 | Signallist \*\*plHead,  Signal \*pNew | 返回0或1，是否有新的信号生成 |
| 9 | FindRoute | 通过id找到线路 | RouteList \*plHead  CTSID id | 返回目标路线 |
| 10 | FindTrain | 通过id找到火车 | TrainList \*plHead  CTSID id | 返回目标火车 |
| 11 | FreeRouteList | 释放线路资源 | RailwayList \*plHead | 无返回值 |
| 12 | FreeTrainList | 释放火车资源 | TrainList \*plHead | 无返回值 |
| 13 | FreeSignalList | 释放信号资源 | Signallist\*plHead | 无返回值 |
| 14 | GenerateSignal | 初始生成信号 | RouteList \*IrtHead  SignalList \*\*lsgHead | 无返回值 |
| 15 | MoveTrain | 移动火车 | Train \*pCur | 无返回值 |
| 16 | CalcSignal | 计算信号 | Signal \*pSignal | 无返回值 |
| 17 | Init | 初始化 | ThreadCtrl \*tc  const char \*szConf | 返回0或1，表示是否初始化成功 |
| 18 | Compute | 计算线程函数 | ThreadCtrl \*tc | 无返回值 |
| 19 | Display | 显示线程函数 | ThreadCtrl \*tc | 无返回值 |
| 20 | Err | 错误提示 | const char \*str | 无返回值 |
| 21 | ConsoleInput | 控制台输入 | ThreadCtrl \*tc | 无返回值 |
| 22 | ReadConf | 读入地图配置 | Const char \*szConf RouteList\*\*plrtHead  TrainList \*\*pltrHead | 返回0或1，表示是否读入地图配置 |
| 23 | WriteConf | 写入地图配置 | RouteList \*lrtHead  TrainList \*ltrHead | 返回0或1，表示是否写入地图配置 |
| 24 | WriteLog | 写入运行日志 | RouteList \*lrtHead  TrainList \*ltrHead | 返回0或1，表示是否写入运行日志 |

⒌核心算法设计

基本架构：

Main

Exit

Error

Init

CreateThread

Error

Exit

Exit

CreatEvent

Error

SetEvent(Compute)

Input

FreeResource

Exit

Compute

If(Exit)

Waiting(Compute)

Computing

↓

SetEvent(Display)

Display

If(Exit)

Waiting(Display)

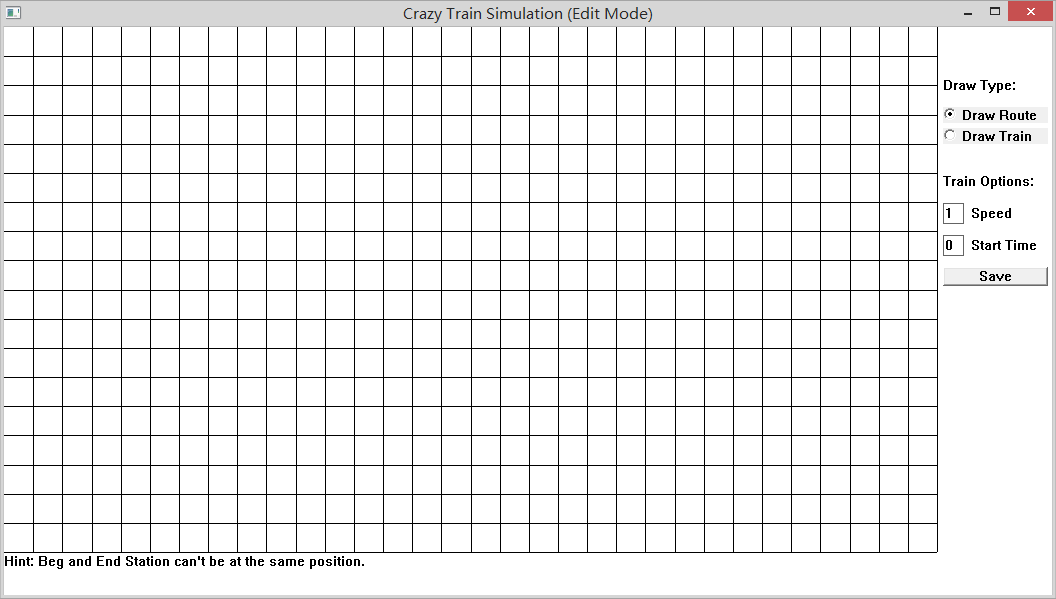
Displaying

↓

SetEvent(Compute)

附录1 关于绘制CTS.dat的图形界面方法

1. 主界面



1. 控制区域



1. Draw Type绘制模式
   * + 1. Draw Route 绘制线路/铁轨 //见3.
       2. Draw Train 绘制火车 //将火车放到起点车站
2. Train Options 火车属性选项 //仅在Draw Train时有效
   * + 1. Speed 火车速度 //1-9
       2. Start Time 发车时间 //0-9
3. Save 保存地图 //保存为CTS.dat
4. 提示区域



1. Hint 提示文字
2. Current Pos 绘制时鼠标当前网格位置
3. 绘制线路/铁轨详细说明
4. 单击网格产生起点车站
5. 移动鼠标产生铁轨
6. 释放鼠标产生终点车站
7. 单击终点车站并移动可以延伸当前铁轨
8. 绘制线路/铁轨详细规则
   1. 铁轨必须有一定长度
   2. 不同线路不能共用起点
   3. 单击终点车站并移动时不能反向延伸铁轨
   4. 起点与相同线路终点重合即变为环路
   5. 起点与不同线路终点不能重合
   6. 其他规则见提示Hint