



华中科技大学

题目：定向越野模拟

C 语言课程设计

需求分析与设计报告

专业班级：自动化类 2109 班

小组成员：叶庭茂 U202115162

梁忻健 U202115158

指导老师：周纯杰、何顶新、汪国有、左峥嵘

周凯波、彭刚、高常鑫、陈忠

目录

第一部分 前言	1
一、 编写背景	1
二、 软件功能	1
三、 项目意义	2
四、 参考资料	2
五、 编程软件	2
第二部分 任务概述	3
一、 目标功能	3
二、 编写规范	5
第三部分 运行环境和配置	6
一、 硬件接口	6
二、 软件接口	6
三、 控制	6
第四部分 需求分析	7
第五部分 算法设计	15
一、 数据结构设计	15
第六部分 界面设计	24
一、 鼠标设计	24
二、 界面逻辑	25
三、 界面设计	26
第七部分 源代码	42
第八部分 总结	769

第一部分 前言

一、 编写背景

定向越野(Orienteering)指的是借助地图 (Map) 和指北针 (Compass) 穿越未知地带到访地图上的若干检查点的运动, 一般情况下以最短时间按照指定顺序到访所有检查点的运动员胜出。定向越野是一项使用高精度地图的导航运动, 无论你是经验丰富的旅行家, 还是健步如飞的运动员, 亦或只是一个小组或者家庭在公园里的小活动, 每次定向越野都能够提升你的导航能力。从操场和迷宫到细节丰富的公园校园, 再到荒无人烟的原始森林, 定向越野可以逐步提升你对地图的阅读和使用的能力。同时, 定向越野运动是有效的对身体素质的锻炼。

当下, 定向越野运动正逐渐火热, 各地都有举办相应的定向越野运动比赛。而在我们华中科技大学, 在《大学体育: 户外运动》这一门课程中, 更是对定向越野有相应的讲解与实践。因此, 作者编写了这一套定向越野模拟系统, 旨在帮助未接触或定向越野新手迅速了解定向越野运动。本软件能够提供丰富多样的随机地图, 同时能够预设人物年龄, 天气等参数, 通过节点路径算法模拟其他参赛者, 同时, 本软件还能提供至多 5 位的其他参赛者, 能够实时显示赛况。软件能够对定向越野比赛进行全方位模拟, 使每一次模拟都能够对提升用户对地图的阅读和使用的能力。帮助用户在足不出户的前提下, 能够多次模拟训练对地图的分析、使用能力。

二、 软件功能

本系统在模拟定向越野陆地赛的真实比赛情况, 通过对赛区地图的随机生成和多位参赛者路径的随机规划, 对定向越野陆地区比赛进行多方面的仿真模拟。

本系统支持对多种地区的赛区模拟, 比如城区、郊区和山区, 不同区域具有的地貌特征各不相同; 也支持对多种天气的模拟, 比如晴天、雨天和阴天, 不同天气将会影响参赛者的运动能力与部分比赛区域的通行状态; 还支持对不同年龄段参赛者的模拟, 比如少年、青年、中年和老年, 不同的年龄段参赛者的运动能力与体力都将不同。

同时, 为了丰富与用户的交互性和让用户能够更加真切地体验定向越野模拟比赛, 本程序还支持用户自己选择一条运动路径, 同随机生成的电脑参赛者一起进行定向越野比赛, 从而帮助用户更好地认识和了解定向越野比赛。

三、 项目意义

定向越野模拟系统通过对实际参赛地点的位置、天气等环境因素和参赛成员的年龄、人数的分析，同时参考了一些相关的游戏和真实的运动数据，我们编写出这一份软件需求分析和功能设计报告。

本报告对于整个“定向越野模拟系统”进行了全面的用户需求和功能分析。包括可行性分析，需求分析，系统功能设计，代码实现，软件优势和劣势，调试功能等等。

同时，本报告明确了本软件系统架构设计，软件结构与数据结构设计，各模块之间的接口和调用，系统界面设计，系统功能设计（函数罗列），具体算法设计以及整个软件的源代码。此外，该项报告也明确了两位开发者的设计与分工，增强了后期测试人员对于软件的调试和验收的可读性与可修改性。

本报告的预期受众为需要了解定向越野比赛的人员，热爱运动的人员以及软件开发与测试人员，工程管理人员。

四、 参考资料

1. 王士元. C 高级实用程序设计. 北京: 清华大学出版社. 1996 年
2. 周纯杰, 何顶新等. 程序设计与应用 (用 C/C++编程). 北京: 机械工业出版社. 2008 年
3. 张海藩, 牟永敏等. 软件工程导论 (第 6 版). 北京: 清华大学出版社. 2013 年

五、 编程软件

Borland C++ 3.1

Dev-C++

Visual Studio 2019

第二部分 任务概述

一、 目标功能

该软件可模拟定向越野比赛，同时允许用户自行选择比赛路径。

1. 登录注册功能

用户进入菜单页面后，可以点击左下角的登录按钮进入登录界面。

如果注册过，可以直接输入用户名和密码，并点击“登录”按钮进行登录，如果用户名不存在或用户名与密码不匹配，将显示错误信息。

用户进入登录页面后，可以点击右上角的“注册”按钮进入注册界面。

输入用户名和密码并确认密码进行注册，若用户已被注册则显示错误信息，若注册成功则自动跳转到登录页面，并在登录页面自动输入本次注册的用户名和密码，用户只需点击“登录”按钮即可登录。

2. 帮助功能

用户进入菜单页面后，可以点击“帮助”按钮进入帮助界面。

帮助界面首先提供了定向越野的比赛介绍，旨在帮助用户了解定向越野比赛的相关知识；

其次提供了定向越野比赛规则的介绍，旨在帮助用户在后续的自选比赛路径中了解路径选择的规则；

还提供了定向越野比赛中各个比赛地区的图例帮助，旨在帮助用户在后续的自选比赛路径中了解路径选择的规则 and 不同地图中每个图例的可通过性；

更提供了定向越野比赛中各个参赛者的人物动画演示，旨在帮助用户在后续的比赛模拟中看懂比赛者的各种动作；

此外还提供有定向越野比赛各年龄段参赛者的确切速度参数，旨在帮助用户了解在各种天气下，各个年龄段的参赛者在比赛中大致的速度和大致体力。

3. 地图随机生成功能

对于各种区域的比赛，本系统将会生成符合实际情况的随机地图，这要求各种真实场景中的元素合理排布，同时地图上从起始点到必行点位再到终点必须有一条可行的路径，确保寻路算法的正常运行。

4. 整幅地图的绘制功能

在随机地图生成之后，需要把随机地图绘制到屏幕上。本系统会根据随机生成的地图信息，进行多次扫描，并绘制出一幅协调、美观的地图。确保用户在选择路径时更为直观。

5. 路径计算功能（寻路功能）

对于各种区域的比赛，本系统将会根据对于随机地图的分析，针对各个必须访问的位置（扰动点位或者必行点位），进行寻路算法的计算，最终生成一组方向数组，用以控制参赛者比赛时行动的路线。

6. 路径随机扰动功能

对于多人参赛的比赛，本系统将会根据对于随机地图的可行性分析，基于广度优先算法的初步分析，从各个可行位置随机抽取一些扰动点位，进行多条随机路径的生成，这些路径作为电脑参赛者的模拟路径。

7. 用户自选路径功能

用户在进入模拟地图之后，将会被要求选择一条符合比赛规则的路线，在用户选择完路线之后，用户所对应的参赛者也会被赋予参赛序号，同电脑参赛者共同参与比赛。

8. 实时动画模拟功能

在用户选择路径之后，能够对包括用户在内的最多 6 名选手进行实时的动画模拟，其中，本系统能够以实时动画的形式，表现选手的位置、运动速度、运动状态、运动方向等。能够给用户以直观的定向越野模拟信息。

9. 参赛者状态显示功能

正式模拟开始之时，本次比赛的相关信息将会显示于模拟地图的上方，左侧将会对参赛的比赛完成情况进行实时的显示。

10. 参赛者成绩查询功能

正式模拟完成之后，将会自动进入比赛查询界面。

用户可以对自选路径的比赛成绩进行查询，亦可以对比赛整体成绩进行查询，还可以对参与比赛的各个选手的成绩进行查询。

二、 编写规范

1. 相对独立的程序块之间、变量说明之后加空行，较长的语句分成多行书写
2. 大括号独占一行并对齐，缩进清晰明了
3. 边写代码边注释，注释的量应达到让程序清晰易懂
4. 注释行在被注释内容的附近，确保注释清晰可见，如：

```
//calculate standard way  
Findpath0(.....);
```

5. 标识符的命名要清晰、明了，有明确含义，如：

```
int clock_hour,clock_minute; //当前比赛时间
```

6. 函数名用大写字母开头的单词和下划线组合而成，其含义清晰可见，如：

```
void Direction_Calculate
```

7. 变量和参数用小写字母开头的单词组合而成，如：

```
int addedpoint_x[4];
```

8. 少使用全局变量
9. 不使用贴图

第三部分 运行环境和配置

一、 硬件接口

处理器：Intel Pentium 166 MX 或以上。

硬盘：空间 500MB 以上。

屏幕适配器：VGA 接口。

系统运行内存：要求 32MB 以上。

二、 软件接口

开发软件工具：Borland C++ 3.1

操作系统：DOS

三、 控制

该系统通过鼠标与键盘直接进行控制。

用户将鼠标移至按键或输入框进行点击，通过键盘的按键输入字符或确认消息框。

第四部分 需求分析

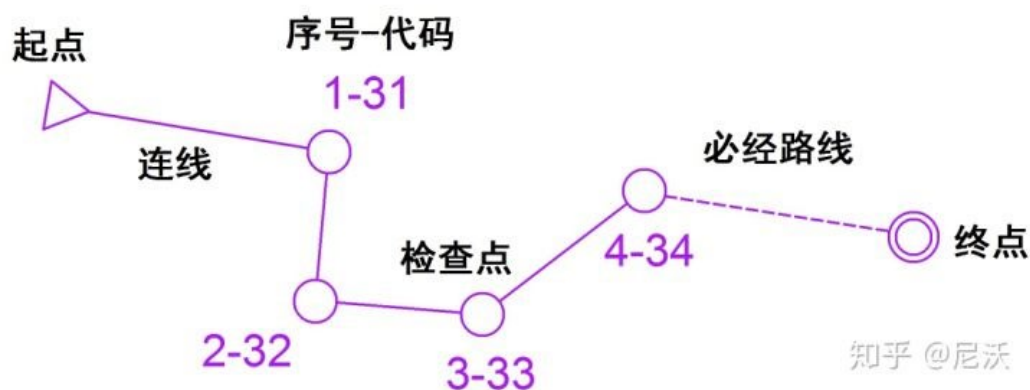
一、概述

使用本软件的目标用户，是希望了解和认识定向越野的人员。对定向越野进行了解分析后，我们认为定向越野模拟的功能的体现应该建立在电脑参赛者的模拟和用户参赛者的模拟的基础上，因为如果没有电脑参赛者的模拟，那么定向越野的模拟将会缺失随机性；而如果没有用户参赛者的模拟，那么定向越野的模拟将会失去用户的参与性。我们希望通过定向越野的模拟来体现中定向越野的基本情况。更进一步的，我们还旨在为用户提供一个参与到定向越野模拟的机会，让用户既能了解定向越野比赛的规则，又能参与到定向越野的比赛中来。

二、参考内容

1. 定向越野比赛点位标识

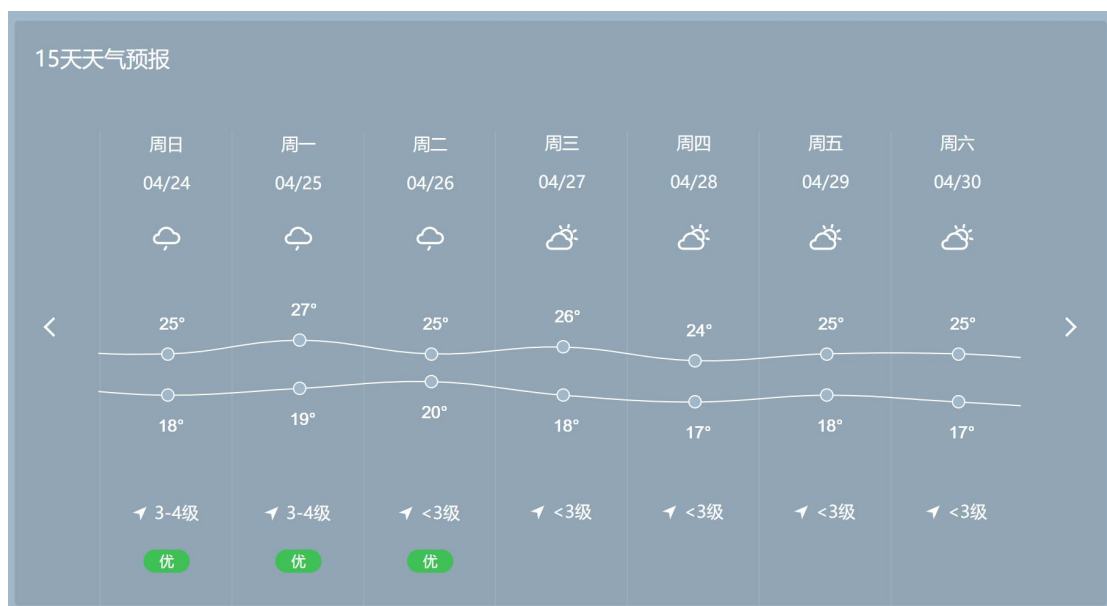
依照比赛点位标识，我们设计了比赛的点位运动规则。



定向越野比赛要求参赛者依次在地图上的打卡点进行打卡，直至到达终点。我们的软件参考了此规则，设计了地图与模拟规则。

2. 天气预报

依照天气预报，我们设计了比赛的天气设置选项。



定向越野比赛的天气包括晴天、阴天和雨天等。我们的软件参考了此信息，设计了模拟定向越野比赛的天气设置选项。

3. 定向越野比赛成绩纸

依照某次真实比赛的成绩纸，我们参考做出了我们的比赛战况搜索界面。

U20年“寻找美丽中华”全国旅游城市
定向系列赛中山南区站和广东省第十
五届定向锦标赛-中距离赛
竞赛日：2020/11/14
姓 名：刘肇天
CH卡号：60019945 选手编号：M14092
成 绩：00:37:06.5
清 除：14:02:14
起 点：14:08:51.3

检查点	用 时	间隔时间
1(054)	00:02:51	00:02:51
2(063)	00:05:27	00:02:36
3(059)	00:10:33	00:05:06
4(064)	00:11:40	00:01:07
5(032)	00:14:55	00:03:15
6(060)	00:22:36	00:07:41
7(061)	00:24:13	00:01:37
8(034)	00:28:00	00:03:47
9(032)	00:31:46	00:03:46
10(033)	00:32:35	00:00:49
11(031)	00:34:30	00:01:55
12(065)	00:36:00	00:01:30
终 点	14:45:57.8	00:01:06

成绩有效性：有效

** 华瑞健定向 0755-26503866 **
**** www.ChinaHealth.cn ****

成绩纸上包括了本次比赛的名字，选手的总体成绩、各个点位的到达成绩、各个点位之间的间隔时间和终点成绩，目的在于清晰显示出该选手的成绩。

三、 功能模块

1. 功能简述

我们结合上述内容，设计出了以定向越野比赛模拟界面为核心界面的程序。由于程序内存上的限制，我们选择 40*28 大小的模拟地图进行模拟，同时为了选择了 1024*768 的 SVGA 显示模式，以更好的展现定向越野比赛的地貌特征和参赛者竞赛情况。

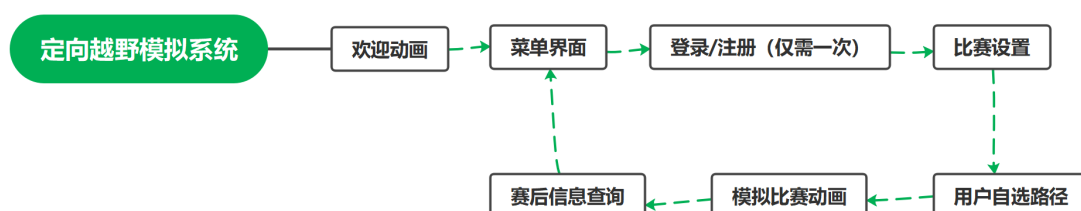
作为核心界面的定向越野模拟界面含有用户自选路径的交互动作，还包括了随机生成的比赛地图以及定向越野比赛模拟的参赛者动画。

总体上，本软件功能以地图模拟生成算法和寻路算法为基础算法，以区块刷新图像显示技术和区块刷新动画技术为核心动画显示技术。用户还能通过自选路径参加比赛，了解关于定向越野比赛的一系列知识。

使用本软件的目标用户，是希望了解和认识定向越野的人员。对定向越野进行了解分析后，我们认为定向越野模拟的功能的体现应该建立在电脑参赛者的模拟和用户参赛者的模拟的基础上，因为如果没有电脑参赛者的模拟，那么定向越野的模拟将会缺失随机性；而如果没有用户参赛者的模拟，那么定向越野的模拟将会失去用户的参与性。我们希望通过定向越野的模拟来体现中定向越野的基本情况。更进一步的，我们还旨在为用户提供一个参与到定向越野模拟的机会，让用户既能了解定向越野比赛的规则，又能参与到定向越野的比赛中来。

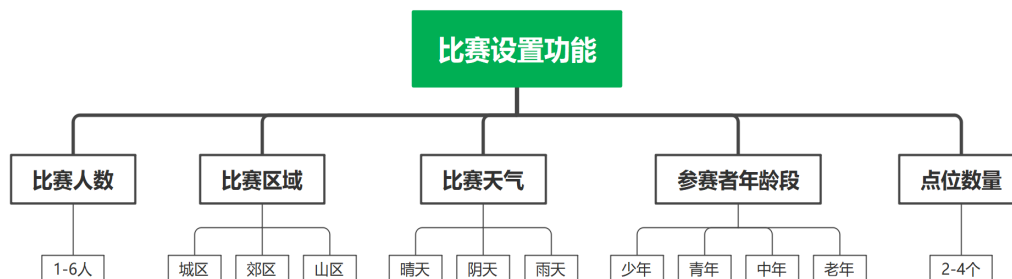
2. 功能流程简述

功能总体流程：

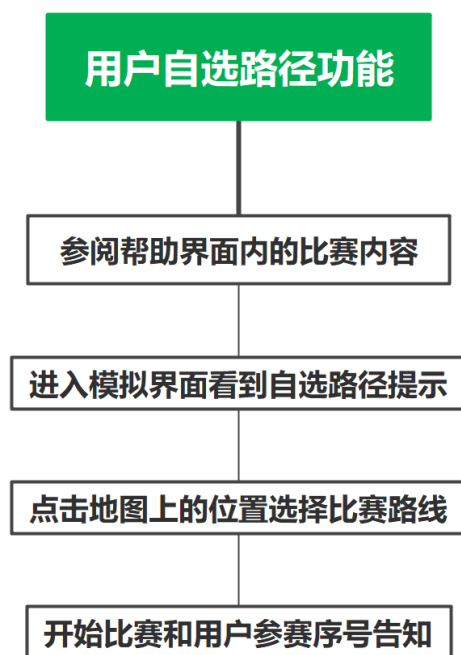


功能具体流程：

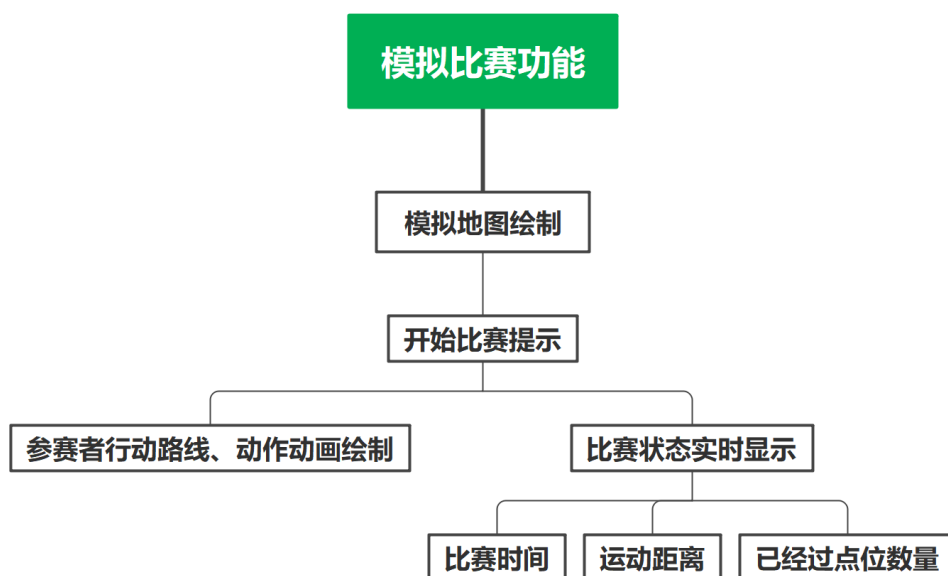
(1) 比赛设置功能



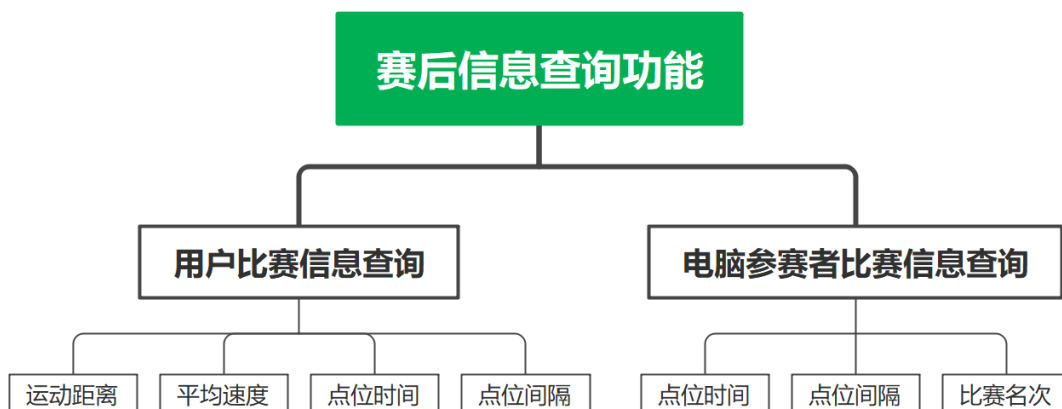
(2) 用户自选路径功能



(3) 模拟比赛功能



(4) 赛后信息查询功能



3. 功能具体介绍

(1) 比赛信息设置

①比赛人数设置

对于用户想模拟的定向越野比赛情况，我们提供了比赛人数的设置选项，人数意味着参赛人员数量的多少。

②比赛区域设置

对于用户想模拟的定向越野比赛情况，我们提供了比赛区域的设置选项，不同的区域意味着参赛者比赛难度的不同，也意味着比赛地图的不同，还意味着参赛者运动速度的不同。

③比赛天气设置

对于用户想模拟的定向越野比赛情况，我们提供了比赛天气的设置选项，不同的天气意味着参赛者比赛难度的不同，也意味着参赛者运动速度的不同。

④参赛者年龄段设置

对于用户想模拟的定向越野比赛情况，我们提供了参赛者年龄段的设置选项，不同的参赛者年龄段意味着参赛者身体素质不同，其中包括着参赛者运动速度的不同和参赛者体力的不同。

⑤点位数量设置

对于用户想模拟的定向越野比赛情况，我们提供了点位数量的设置选项，不同的参赛者年龄段意味着参赛者身体素质不同，其中包括着参赛者运动速度的不同和参赛者体力的不同。

(2) 模拟地图绘制

①模拟地图生成

对于各种区域的比赛，我们将会根据用户选择的比赛区域选项，生成符合实际情况的随机地图，这要求各种真实场景中的元素合理排布，同时地图上从起始点到必行点位再到终点必须有一条可行的路径，确保寻路算法的正常运行。

②模拟地图显示

根据生成的模拟地图数组，我们将会在屏幕上绘制一个尽可能接近实际的模拟地图，大小为 40*28 格，还包括参赛成员必须到达的点位，使用户能够清晰明了地看出各位参赛者的运动路线。

(3) 参赛者竞赛功能

①参赛者运动显示

对于模拟的定向越野比赛，我们将会为每一位参赛者（包括用户参赛者和电脑参赛者）绘制相应的运动路线和运动动作。

根据每一位参赛者的运动能力（包括运动体力和运动速度），系统将会计算其在地图上每一个位置的到达时间和运动动作，并将其在模拟系统中绘制出来。

其中运动动作包括有走路、跑步、爬山，这些动作的对应速度会受到天气、参赛者体力、位置对应的地貌状态和参赛者年龄段多个因素的影响。

②参赛者状态显示

对于模拟的定向越野比赛，我们将会为每一位参赛者（包括用户参赛者和电脑参赛者）实时显示运动状态。

运动状态包括其运动距离、运动时间和已经过点位数量。

③比赛时间显示

对于模拟的定向越野比赛，我们将提供模拟时钟的功能，时间将被显示于右上角。

(4) 赛后信息查询

对于用户参赛者和电脑参赛者在整个比赛之中的运动表现，我们提供有比赛信息的查询界面。

①用户参赛者查询

对于用户参赛者的比赛信息查询，我们提供了详细的比赛数据显示。

基础信息包括用户姓名、用户比赛成绩、用户参赛开始时间、用户比赛完成时间、用户比赛运动距离、用户比赛平均速度。这些信息有助于用户了解本次模拟中自己运动的大致情况。

详细信息包括各个点位的打卡时间和点位之间的运动时间。这些信息有助于用户了解自己比赛的详细完成情况。

②电脑参赛者查询

对于用户参赛者的比赛信息查询，我们提供了较为详细的比赛数据查询功能，支持按照比赛序号和比赛名次进行查询。

搜索出的基础信息包括了参赛者序号、成绩和名次。这些信息有助于用户了解本次模拟中其余参赛者运动的大致情况。

搜索出的详细信息包括各个点位的打卡时间和点位之间的运动时间。这些信息有助于用户了解其余参赛者比赛的详细完成情况。

第五部分 算法设计

一、数据结构设计

结构体：

1.用户基础信息

```
typedef struct people1
{
    int number; //序号

    int ps_consume; //体力剩余值

    int ps_recover; //体力恢复值

    int color; //人物衣服颜色

    double verb_walk; //人物走路速度

    double verb_run; //人物跑步速度

    double verb_climb; //人物爬山速度
}people_basic;
```

2.用户比赛信息

```
typedef struct people2
{
    int distance; //人物运动距离

    int hour_point[4]; //点位到达小时数

    int minute_point[4]; //点位到达分钟数

    int rank; //排名

    int hour_interval[4]; //点位间隔小时数

    int minute_interval[4]; //点位间隔分钟数

    int hour_score; //成绩小时数

    int minute_score; //成绩分钟数
}people_competition;
```

3.A*算法结构体变量

```
typedef struct Node //node 结构体
{
    int f,g,h; //f(总代价)=g(初始代价)+h(曼哈顿距离)
    int X; //节点横坐标
    int Y; //节点纵坐标
    struct Node* parent; //父节点 Node 结构体首地址
}Node,*Lnode;
```

4.A*算法结构体变量

```
typedef struct Stack //Open Closed 表结构体
{
    Node* npoint; //节点 Node 结构体首地址
    struct Stack* next; //下一节点结构体首地址
}Stack,*Lstack;
```

数据库:

1.用户登录注册信息存储

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00000000h:	61	64	6D	69	6E	00	00	00	00	00	00	61	64	6D	69	6E	; admin.....admin
00000010h:	00	00	00	00	00	00	75	73	65	72	00	00	00	00	00	00	;user.....
00000020h:	00	75	73	65	72	00	00	00	00	00	00	00	63	6F	6C	64	; .user.....cold
00000030h:	00	00	00	00	00	00	00	63	6F	6C	64	00	00	00	00	00	;cold....
00000040h:	00	00	6C	78	6A	00	00	00	00	00	00	00	00	6C	78	6A	; ..lxj.....lxj
00000050h:	00	00	00	00	00	00	00	70	72	6F	66	00	00	00	00	00	;prof....
00000060h:	00	00	00	70	72	6F	00	00	00	00	00	00	00	67	69	00	; ...pro.....gi
00000070h:	6E	61	00	00	00	00	00	00	00	67	69	6E	00	00	00	00	; na.....gin....
00000080h:	00	00	00	00													;

用户名与密码以两个 10+1 长度的 char 存储到 DATA 下的 USER.DAT 中，不足 10 位以'\0'补全。

2. 区块地图数据存储

☐ MAP0.DAT
☐ MAP1.DAT
☐ MAP2.DAT
☐ MAP3.DAT
☐ MAP4.DAT
☐ MAP5.DAT
☐ MAP6.DAT
☐ MAP7.DAT
☐ MAP8.DAT
☐ MAP9.DAT
☐ MAP10.DAT
☐ MAP11.DAT
☐ MAP12.DAT
☐ MAP13.DAT
☐ MAP14.DAT
☐ MAP15.DAT
☐ MAP16.DAT
☐ MAP17.DAT
☐ MAP18.DAT
☐ MAP19.DAT
☐ MAP20.DAT
☐ MAP21.DAT
☐ MAP22.DAT
☐ MAP23.DAT
☐ MAP24.DAT
☐ MAP25.DAT

共 57 个区块地图二进制文件。

二、算法说明

1. 寻路算法

我们的导航系统采用了 A* 算法求取最短路径。

首先初始化第一个节点，然后不断地向四周搜索可行节点，搜索到的可行节点加入 Open 列表，而搜索过的不合适节点加入 Closed 列表。通过计算每一个节点的 F（总代价）值，不断筛选最为合适的节点，直到搜索到终点。

最终由于需要获取方向数组，由链表不断反向搜索即可得到 A* 算法下的相对最短路径。

核心代码如下：

```
//init first node and Open stack
startNode->parent= NULL;
startNode->X = startX;
startNode->Y = startY;
startNode->g = 0;
startNode->h = getH(startX,startY,endX,endY);
startNode->f = startNode->g + startNode->h;
PutintoOpen(startNode);
while(1)
{
    bestNode=getFminNodeFromOpen();
```

```

if(bestNode->X==endX&&bestNode->Y==endY)
{
    bestNode1 = bestNode,nodeSum = 0,nodeIndex = 0
;
    while( bestNode1->parent != NULL )
    {
        bestNode1 = bestNode1->parent;
        nodeSum += 1;
    }
    bestNode1=bestNode,nodeIndex=nodeSum-1;
    while(bestNode1->parent!= NULL&&nodeIndex>=0)
    {
        bestNode1Parent = bestNode1->parent;
        //output path
        if( bestNode1Parent->X - bestNode1->X == 0
&& bestNode1Parent->Y - bestNode1->Y == +1)
        {
            direction[nodeIndex+(*step)] = 1;
            //up
        }
        else if( bestNode1Parent->X - bestNode1->X
== 0 && bestNode1Parent->Y - bestNode1->Y == -1)
        {
            direction[nodeIndex+(*step)] = 2;
            //down
        }
        else if( bestNode1Parent->X - bestNode1->X
== +1 && bestNode1Parent->Y - bestNode1->Y == 0)
        {
            direction[nodeIndex+(*step)] = 3;
            //left
        }
        else if( bestNode1Parent->X - bestNode1->X
== -1 && bestNode1Parent->Y - bestNode1->Y == 0)
        {
            direction[nodeIndex+(*step)] = 4;
            //right
        }
        else if( bestNode1Parent->X - bestNode1->X
== +1 && bestNode1Parent->Y - bestNode1->Y == +1)
        {
            direction[nodeIndex+(*step)] = 5;
            //left-up

```

```

        if(direction[nodeIndex+(*step)+1]==6)
        {
            if(map_process[bestNode1->X][bestNode1Parent->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
                =3;
                direction[nodeIndex+(*step)]=3
                ;
            }
        }
        else if(direction[nodeIndex+(*step)+1]
        ==7)
        {
            if(map_process[bestNode1Parent->X]
            [bestNode1->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
                =1;
                direction[nodeIndex+(*step)]=1
                ;
            }
        }
        //make avenue straight
    }
    else if( bestNode1Parent->X - bestNode1->X
    == +1 && bestNode1Parent->Y - bestNode1->Y == -1 )
    {
        direction[nodeIndex+(*step)] = 6;
        //left-down
        if(direction[nodeIndex+(*step)+1]==8)
        {
            if(map_process[bestNode1Parent->X]
            [bestNode1->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
                =2;
                direction[nodeIndex+(*step)]=2
                ;
            }
        }
        else if(direction[nodeIndex+(*step)+1]
        ==5)

```

```

        {
            if(map_process[bestNode1->X][bestN
ode1Parent->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
=3;
                direction[nodeIndex+(*step)]=3
;
            }
        }
        //make avenue straight
    }
    else if( bestNode1Parent->X - bestNode1->X
== -1 && bestNode1Parent->Y - bestNode1->Y == +1 )
    {
        direction[nodeIndex+(*step)] = 7;
        //right-up
        if(direction[nodeIndex+(*step)+1]==8)
        {
            if(map_process[bestNode1->X][bestN
ode1Parent->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
=4;
                direction[nodeIndex+(*step)]=4
;
            }
        }
        else if(direction[nodeIndex+(*step)+1]
==5)
        {
            if(map_process[bestNode1Parent->X]
[bestNode1->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
=1;
                direction[nodeIndex+(*step)]=1
;
            }
        }
        //make avenue straight
    }
    else if( bestNode1Parent->X - bestNode1->X

```

```

== -1 && bestNode1Parent->Y - bestNode1->Y == -1 )
    {
        direction[nodeIndex+(*step)] = 8;
        //right-down
        if(direction[nodeIndex+(*step)+1]==6)
        {
            if(map_process[bestNode1Parent->X]
[bestNode1->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
=2;
                direction[nodeIndex+(*step)]=2
;
            }
        }
        else if(direction[nodeIndex+(*step)+1]
==7)
        {
            if(map_process[bestNode1->X][bestN
ode1Parent->Y]==0)
            {
                direction[nodeIndex+(*step)+1]
=4;
                direction[nodeIndex+(*step)]=4
;
            }
        }
        //make avenue straight
    }
    nodeIndex--;
    bestNode1=bestNode1->parent;
}
(*step)+=nodeSum;
break;
}
seachSeccessionNode(bestNode,endX,endY);
//generate son-node
}

```

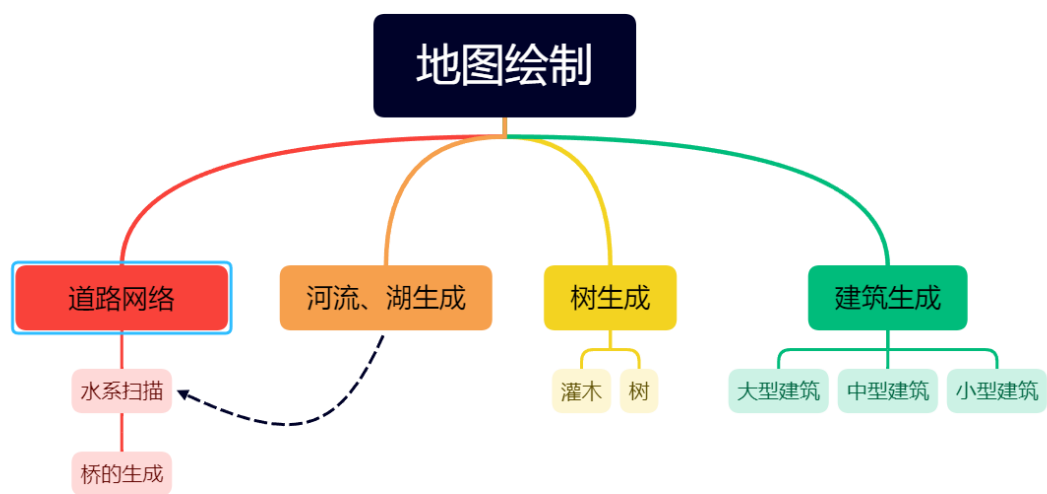
2. 地图扫描绘制算法

对于地图绘制，我们采取了扫描算法。

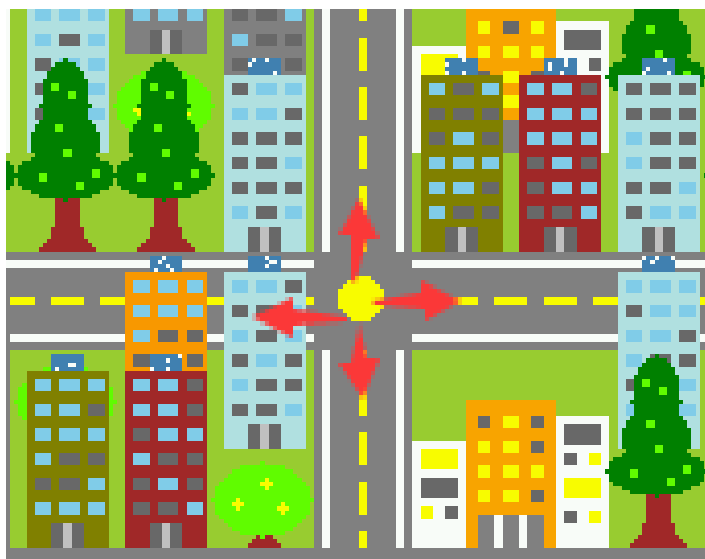
在一个基于规则生成的二维数组中，进行二维数组的遍历，同时根据当前点位信息，向四周方向的地图坐标进行扫描，根据扫描结果绘制所有结果，并记录绘制状态为已绘制，后续遍历不再对改点位进行绘制。（以下仅列举其中一种扫描应用）例如桥长扫描绘制和交叉路口扫描，交叉路口扫描能根据扫描到交叉口绘制不同形式的地图。

最终在扫描绘制的地图，能够得到完整有序的道路网络与建筑排布。

扫描逻辑：



扫描示意：



示例代码:

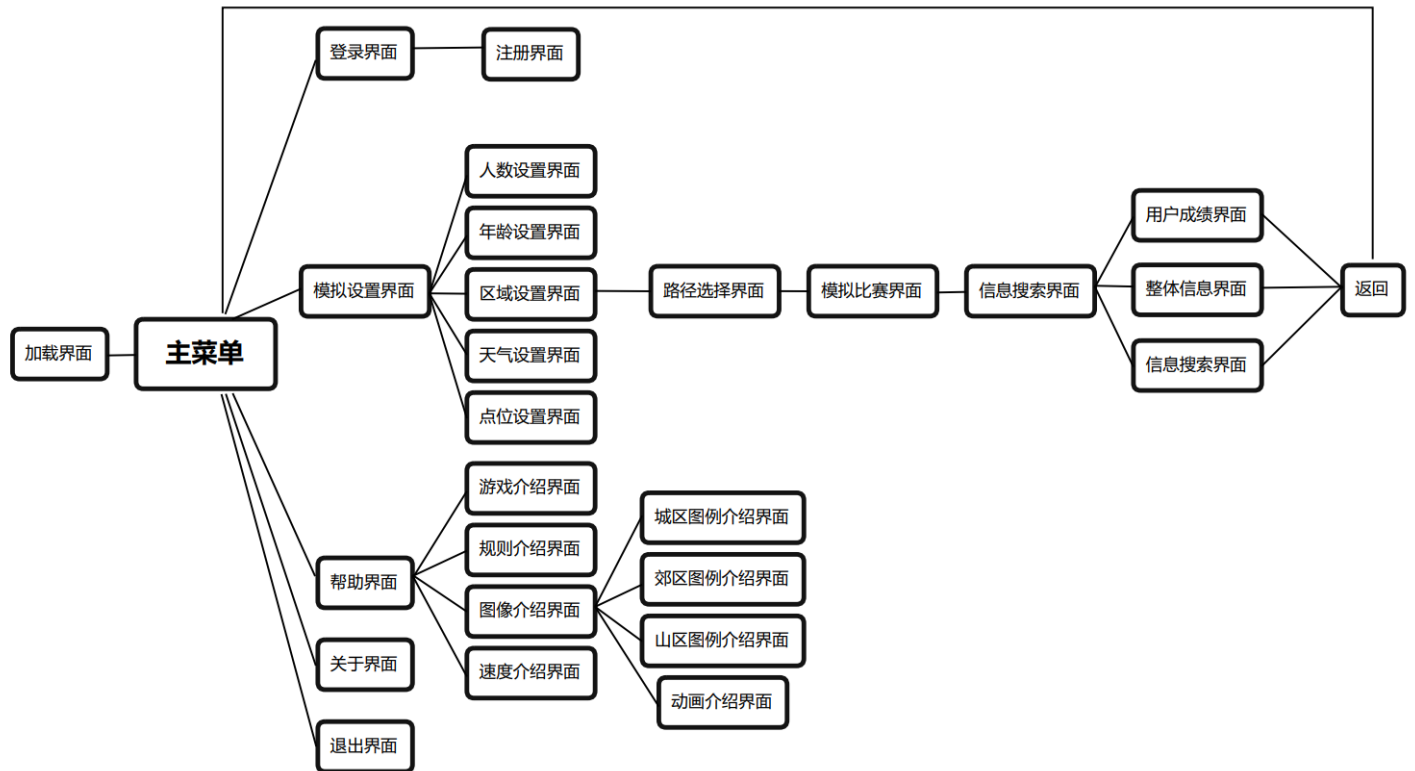
```
int Scan_Sub_Bridge_Crossing(int i,int j)
{
    int road_count=0;
    if(map_display[i+1][j]==3||map_display[i+1][j]==7)
    {
        road_count++;
    }
    if(map_display[i][j+1]==3||map_display[i][j+1]==7)
    {
        road_count++;
    }
    if(map_display[i-1][j]==3||map_display[i-1][j]==7)
    {
        road_count++;
    }
    if(map_display[i][j-1]==3||map_display[i][j-1]==7)
    {
        road_count++;
    }
    if(road_count==3||road_count==4)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

第六部分 界面设计

一、 鼠标设计

我们使用 `MouseX`, `MouseY`, `press`, `shape` 四个参数来获取鼠标的状态。
`MouseX`, `MouseY` 分别代表鼠标的横、纵坐标, `shape` 代表鼠标形态, `press` 代表鼠标左键按压状态。`shape` 默认为箭头鼠标, 1 为十字鼠标, 2 为光标, 3 为手型鼠标; 鼠标左键按下则 `press` 为 1, 右键按下则 `press` 为 2, 抬起则 `press` 为 0。

二、 界面逻辑



三、 界面设计

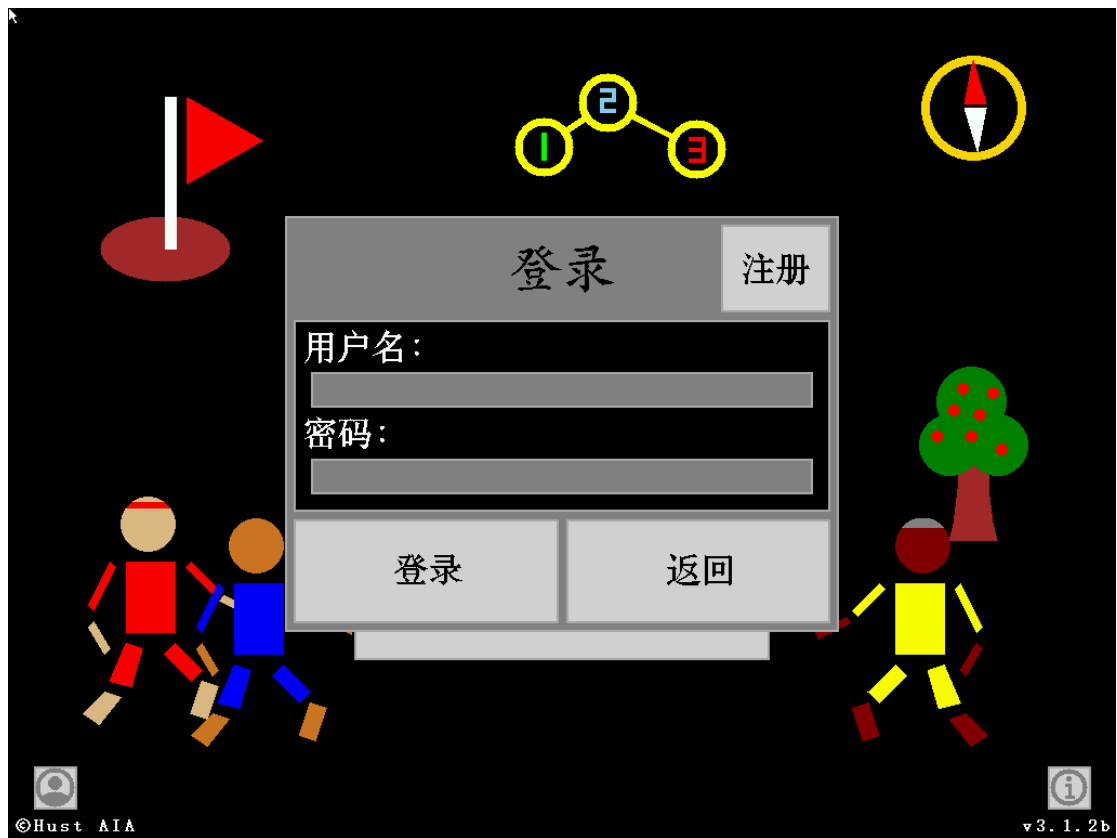
欢迎界面：

欢迎界面使用进度条显示加载进度。



登录界面：

登录界面使用消息框类型界面，同时点击输入框可输入至多 10 位的数据，点击回车或 esc 键完成输入。



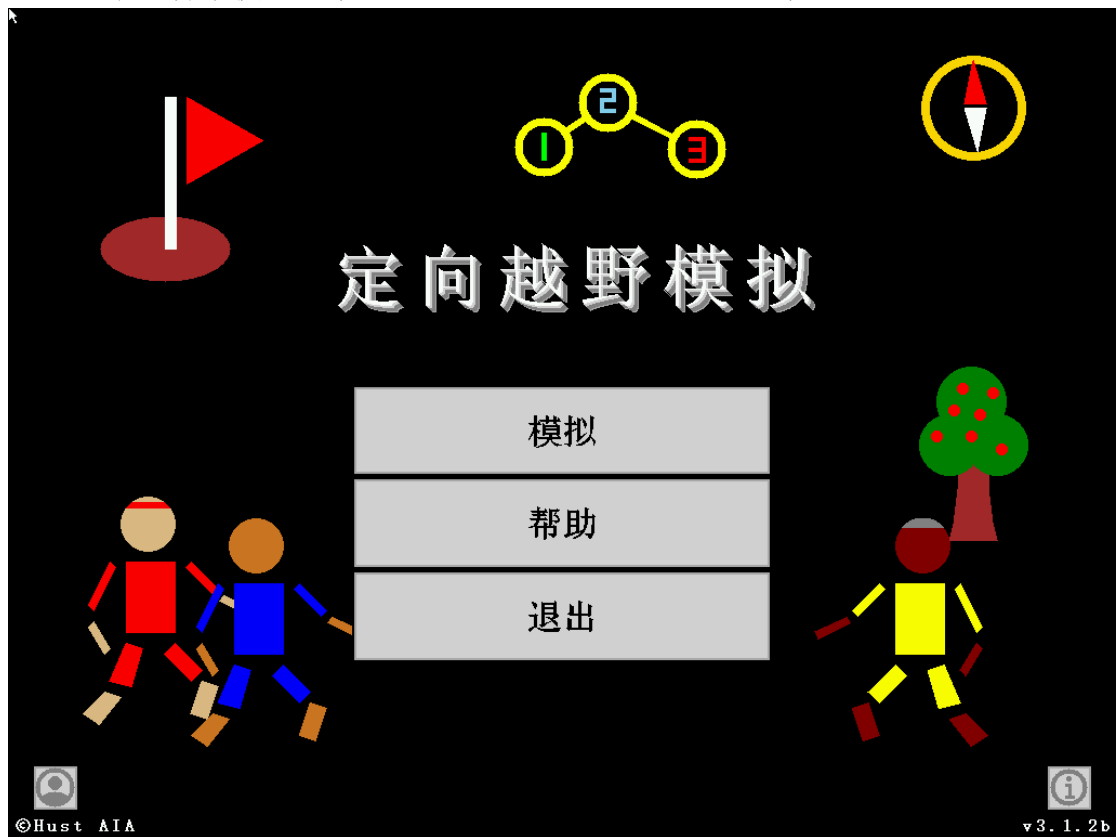
注册界面：

注册界面使用消息框类型界面，同时点击输入框可输入至多 10 位的数据，点击回车或 esc 键完成输入。需要对密码进行一次确认。



主菜单：

主菜单背景使用函数绘制，展现有关定向越野的元素。



模拟菜单界面：

模拟菜单能够在左侧栏设置模拟的数据，如人数、年龄、区域（战区）、天气、点位等数据。以便在接下来的模拟动画中表现。

开始模拟

参赛人员设置

人数

年龄

比赛场地设置

区域

天气

点位

模拟

开始模拟

参赛人员设置

人数

年龄

比赛场地设置

区域

天气

点位

人数

减少

一人

增加

模拟	年龄
开始模拟	少年 走路、跑步、爬山速度不快，体质不差，体力恢复不快，可以支撑适量距离的跑步
参赛人员设置	
人数	青年 走路、跑步、爬山速度很快，体质很好，体力恢复很快，可以支撑较远距离的跑步
年龄	中年 走路、跑步、爬山速度中等，体质中等，体力恢复中等，可以支撑中等距离的跑步
比赛场地设置	老年 走路、跑步、爬山速度不快，体质较差，体力恢复较慢，但仍可以支撑少量距离的跑步
区域	
天气	
点位	

模拟	区域
开始模拟	城区 以公路和建筑为主，还有不少的树木，有时会有河流和湖泊
参赛人员设置	
人数	郊区 以草地和树木为主，还有不少的田地、小径和建筑物，有时会有河流和公路
年龄	山区 以山、树木和草地为主，还有不少的河流和湖泊，有时会有沼泽和小屋
比赛场地设置	
区域	
天气	
点位	

开始模拟

参赛人员设置

人数

年龄

比赛场地设置

区域

天气

点位

天气

晴天
天气较为炎热，走路速度减少五分之一跑步、爬山，速度减半

多云
天气较为舒适，走路、跑步、爬山速度不受影响

雨天
天气较为糟糕，走路、跑步、爬山速度减少二分之一

开始模拟

参赛人员设置

人数

年龄

比赛场地设置

区域

天气

点位

点位

减少

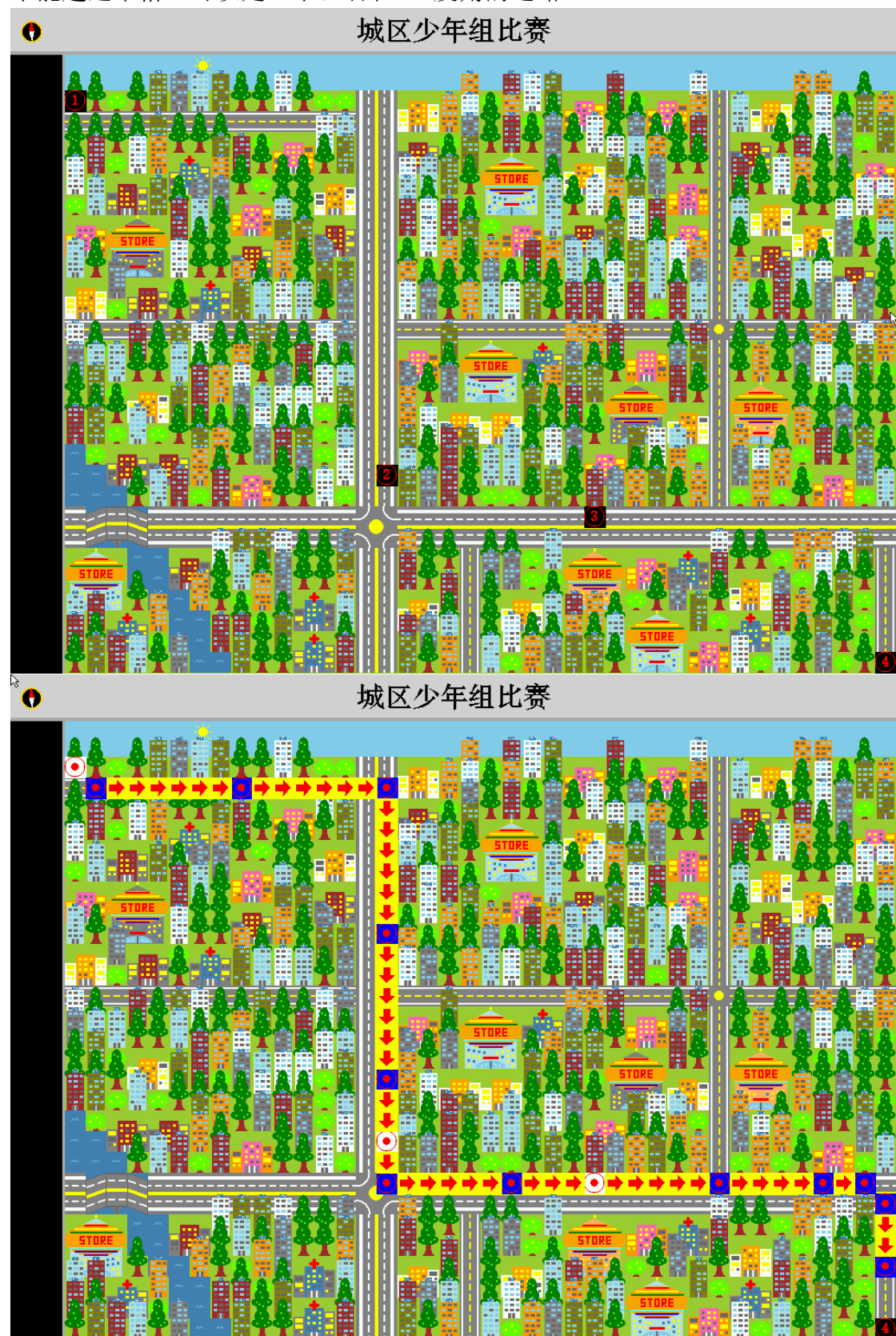
二个

增加

32

路径选择界面：

用户可以在地图上点击选择路径，其中必须走过标黑点位。而且一次选择不能超过十格。可以走上下左右和 45 度角的道路。



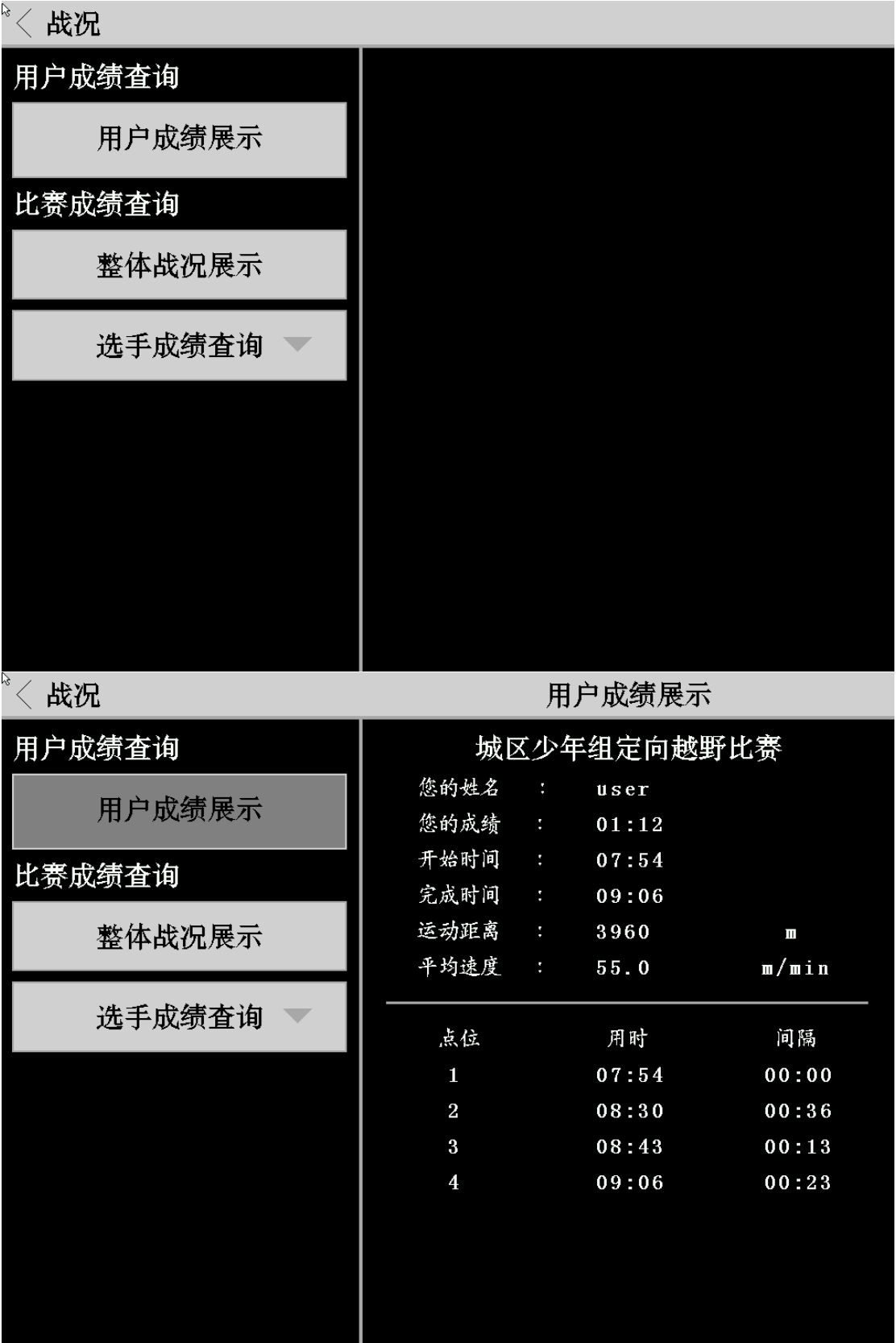
模拟比赛界面：

模拟界面下，实时显示天气系统。在主绘图区，实时显示选手的位置，速度，动作等动画，左侧菜单栏显示选手的完成度、出发时间、完成点位等数据。同时，用户信息标亮显示。右上角显示实时时间。



信息搜索界面：

信息搜索界面可以显示用户的成绩。同时支持显示整体战况，也支持按序号、排名对单个选手进行查询。



战况

用户成绩查询

用户成绩展示

比赛成绩查询

整体战况展示

选手成绩查询

整体战况展示

城区少年组定向越野比赛

起始时间 : 07:47

结束时间 : 09:01

成员成绩

名次	序号	成绩
1	4	01:11
2	5	01:12
3	6	01:12
4	1	01:24
5	3	01:34
6	2	01:45

战况

用户成绩查询

用户成绩展示

比赛成绩查询

整体战况展示

选手成绩查询

选手成绩查询

2

查询

城区少年组定向越野比赛

序号 : 2

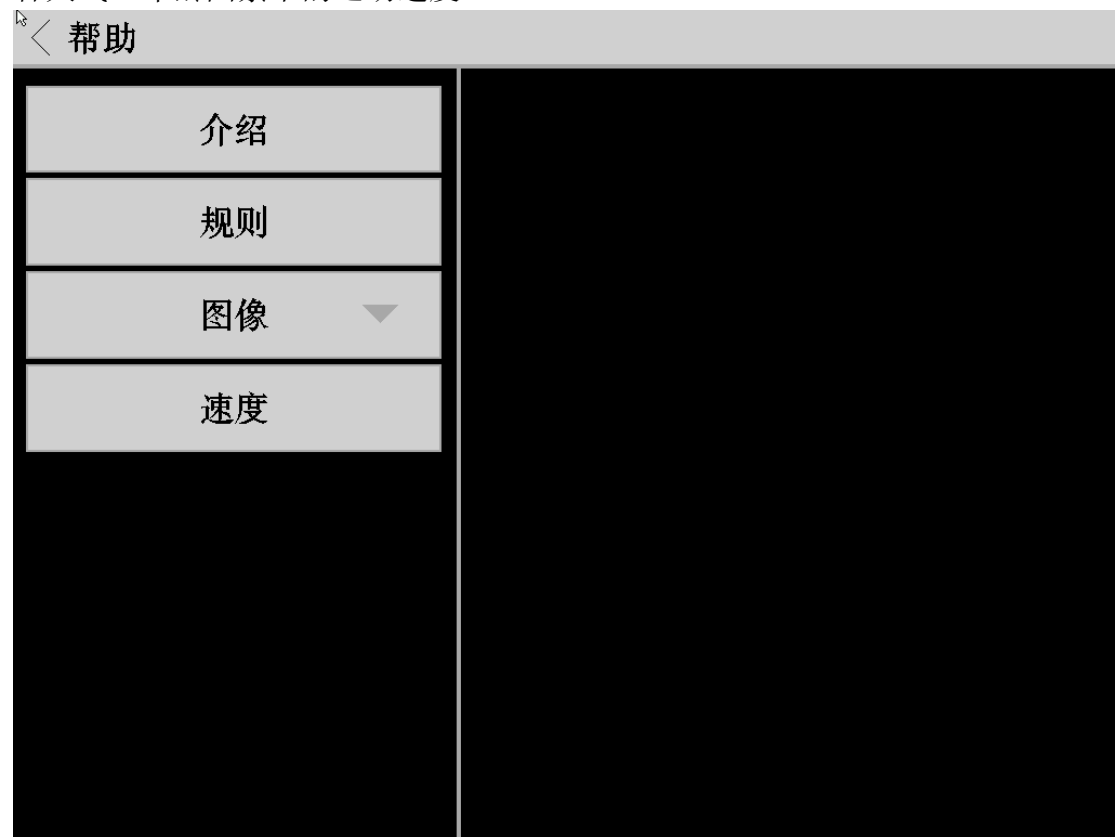
名次 : 6

成绩 : 01:45

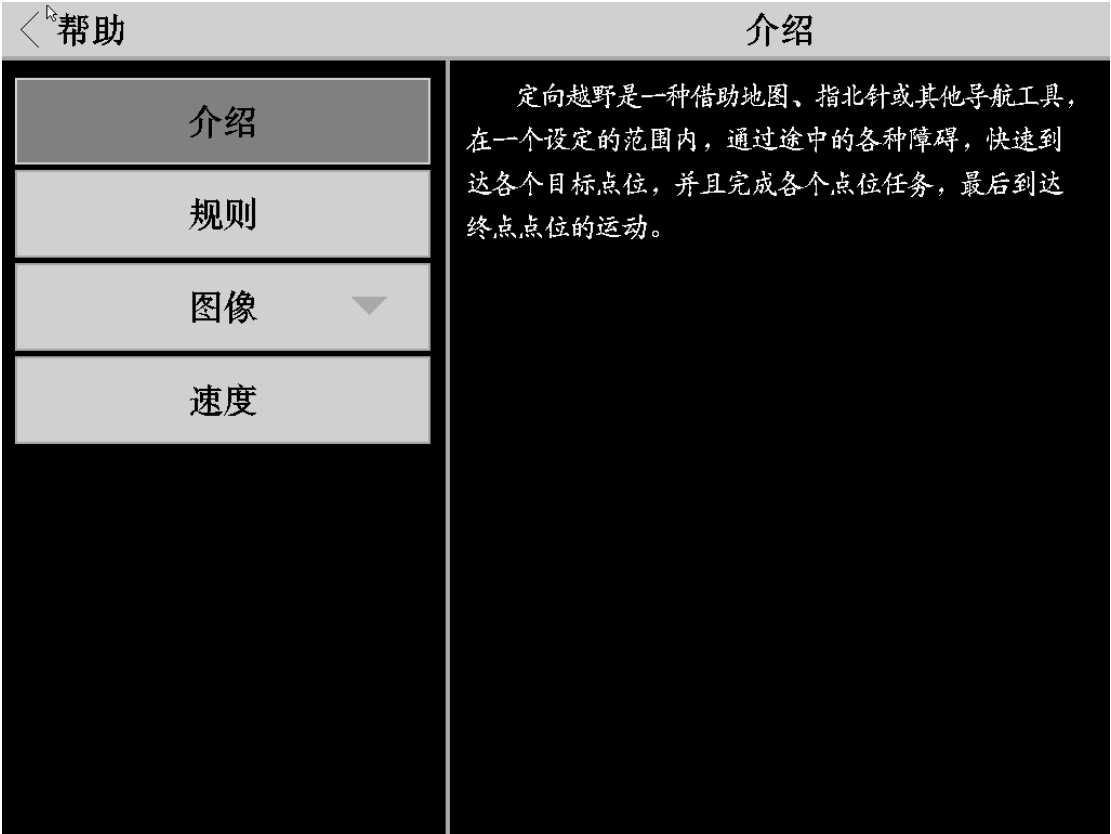
点位	用时	间隔
1	07:53	00:00
2	08:42	00:49
3	08:55	00:13
4	09:38	00:43

帮助界面：

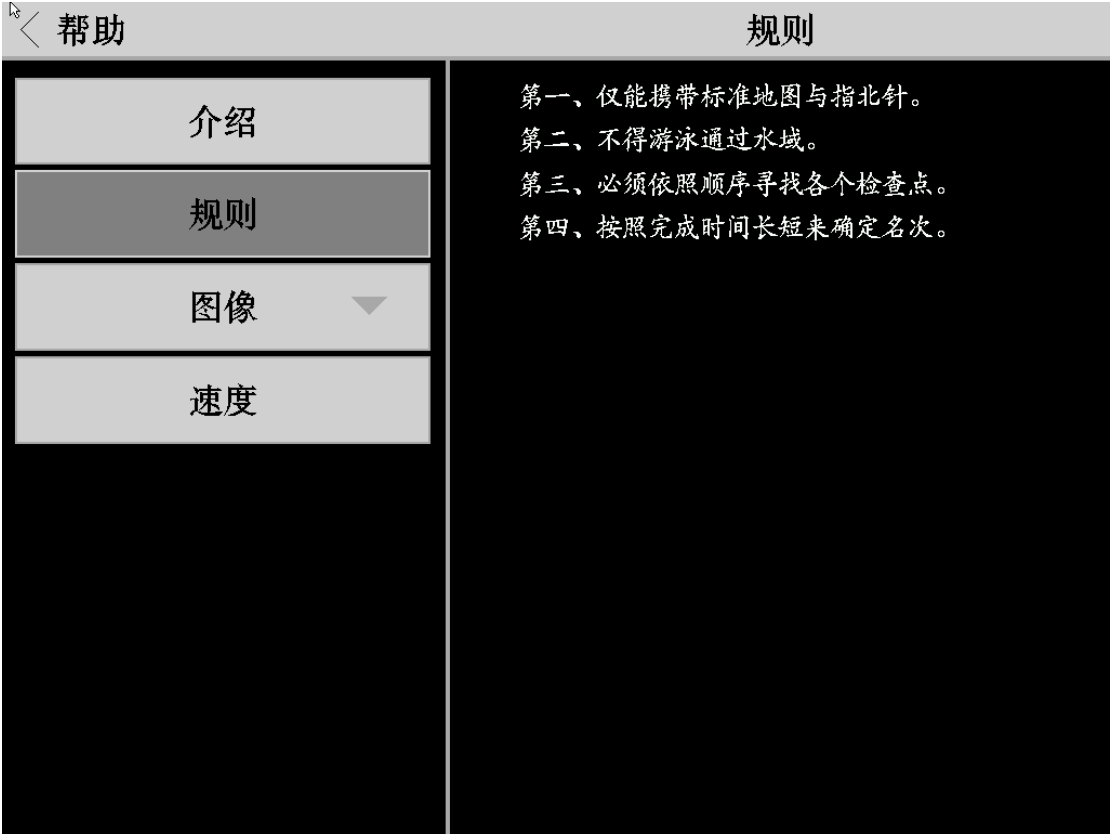
帮助界面包含了游戏介绍与规则，以及各种图例和动画示例，同时注明了各天气、年龄因素下的运动速度。








软件介绍页面：



游戏规则介绍页面：



城区图例页面：

< 帮助		图像	
<div>介绍</div> <div>规则</div> <div>图像</div> <div>速度</div>	图例	名称	可通过性
		建筑物	无法通过
		树木	可以通过
		公路	可以通过
		河流	无法通过
		湖泊	无法通过
		桥	可以通过

郊区图例页面：

< 帮助		图像	
<div>介绍</div> <div>规则</div> <div>图像</div> <div>速度</div>	图例	名称	可通过性
		树木	可以通过
		草地	可以通过
		建筑物	无法通过
		田地	可以通过
		小径	可以通过
		河流	无法通过
		桥	可以通过
		公路	可以通过

山区图例页面：

< 帮助




介绍

规则

图像

速度

图例

图例	名称	可通过性
	山	可以通过
	树木	可以通过
	草地	可以通过
	河流	无法通过
	湖泊	无法通过
	沼泽地	雨天无法通过
	桥	可以通过
	小屋	无法通过

动画示例页面（点击方框播放对应动画）：

< 帮助

介绍

规则

图像

速度

动画

动画	少年	青年	中年	老年
走路				
跑步				
爬山				

速度介绍页面：

帮助

速度

介绍

规则

图像

速度

数值	少年	青年	中年	老年
体力	20-30	45-60	30-50	10-20
恢复	5	15	10	5
走路	1	2	1.5	1
跑步	2	4	3	1.5
爬山	0.5	1.5	1	0.5

数值含义：速度表示一分钟内能走过的方块数，跑步一格方块消耗对应体力，走路一格方块恢复对应体力

天气

晴天 走路速度减少五分之一，跑步、爬山速度减半

多云 走路、跑步、爬山速度不受影响

雨天 走路、跑步、爬山速度减少二分之一

软件信息页面：

关于

欢迎使用定向越野模拟软件！

本软件基于BC3.1搭建，采用SVGA显示模式，旨在模拟定向越野陆地区域比赛。

受限于性能，可能会出现卡顿情况，敬请谅解！

本软件不用作任何商业用途，版权归原作者所有，请勿随意转载。

如果想要了解更多信息，请查阅位于软件根目录下readme.md与version.md。

2022.3.23

开发人员

华中科技大学人工智能与自动化学院
自动化2109班：叶庭茂、梁忻健

指导教师

华中科技大学人工智能与自动化学院
教授、副教授

第七部分 源代码

一、 头文件

1. about.h

```
#ifndef __about_h__
#define __about_h__

extern void About(int *page);
extern void Draw_About();
extern void Button_Light_About(int flag);
extern void Button_Darken_About(int flag);

#endif
```

2. advance.h

```
#ifndef _advance_h
#define _advance_h

//180° arc line
extern void Arc_up(int x,int y,int r,int color);
extern void Arc_down(int x,int y,int r,int color);
//90° arc line
extern void Arc_right_up(int x,int y,int r,int color);
extern void Arc_right_down(int x,int y,int r,int color);
extern void Arc_left_up(int x,int y,int r,int color);
extern void Arc_left_down(int x,int y,int r,int color);
//90° fullfilled sector 竖直水平分4 块
extern void Solid_QuarterSector_right_up(int x,int y,int r,int color);
extern void Solid_QuarterSector_left_up(int x,int y,int r,int color);
extern void Solid_QuarterSector_left_down(int x,int y,int r,int color);
extern void Solid_QuarterSector_right_down(int x,int y,int r,int color);
//90° fullfilled sector 斜45°分四块
extern void Solid_QuarterSector_down(int x,int y,int r, int color);
extern void Solid_QuarterSector_up(int x,int y,int r, int color);
extern void Solid_QuarterSector_left(int x,int y, int r, int color);
extern void Solid_QuarterSector_right(int x,int y,int r,int color);
```

```
#endif
```

3. `animate.h`

```
#ifndef _ANIMATE_H_
#define _ANIMATE_H_

#define EXP 0

extern void Animate_Help_Teen(int x,int y,char forward,char move,
int color);
extern void Map_Part_Save(int partx,int party,char* address);
extern void Map_Data_Save();
extern void Map_Part_Print(int parti, int partj);
extern void Ani_Test();
extern void Map_Refresh(int i,int j);
extern void Map_Head_Save();
extern void Map_Head_Print();
extern void Animate(int **direction,int people,int age,int *clock
_hour,int *clock_minute,people_basic *person_basic,people_competi
tion *person_competition,int point,int point_x[],int point_y[],in
t your_number,int weather);
extern void Animate_Help(int x,int y,char forward,char move,int a
ge,int color);
extern void Color_Random(people_basic *person_basic,int people);
extern int Scan_Advance(int direction,int plus);

#endif
```

4. `asc.h`

```
#ifndef __asc_h__
#define __asc_h__

extern void putzm(int x, int y,int flag,int part,int color,char *
s);
extern void putsz(int x, int y,int flag,int part,int color,int sz
);
extern void putsz_double(int x, int y,int flag,int part,int color
,double sz);

#endif
```

5. back.h

```
#ifndef _BACK_H_
#define _BACK_H_

extern void Back_People_A(int x,int y,int zoom,int color);
extern void Back_People_B(int x,int y,int zoom,int color);
extern void Back_People_C(int x,int y,int zoom,int color);
extern void Back_Flag(int x,int y,int zoom);
extern void Back_Tree(int x,int y,int zoom);
extern void Back_Redball(int x,int y);
extern void Back_Compass(int x0,int y0);
extern void Back_Point(int x,int y);
extern void Back();
extern void Scarf(int x,int y,int radius,int up,int down,int color);

#endif
```

6. clock.h

```
#ifndef _clock_h
#define _clock_h

extern void Clock_Random(int* clock_hour,int* clock_minute,int *clock_hour_start,int *clock_minute_start);
extern void Clock_Display(int x0,int y0,int hour,int minute,int color);
extern void SystemClock_Plus(int* clock_hour,int* clock_minute,int plus_hour,int plus_minute);
extern int Clock_Minus(int hour1,int minute1,int hour2,int minute2);
extern int Clock_Compare(int hour1,int minute1,int hour2,int minute2);
extern void ClockEnd_Get(int people,int point,int *clock_hour_end,int *clock_minute_end,people_basic *person_basic,people_competition *person_competition);

#endif
```

7. color.h

```
#ifndef _colore_h_
#define _colore_h_

#define BLACK 0
```

```

#define DIMGRAY 27469
#define GRAY 33808
#define DARK_GRAY 44373
#define SILVER 50712
#define LIGHT_GRAY 54938
#define GAINSBORO 57083
#define WHITE_SMOKE 63422
#define WHITE 65535
#define SNOW 65503
#define IRON_GRAY 25290
#define SAND_BEIGE 58904
#define ROSY_BROWN 48241
#define LIGHT_CORAL 62480
#define INDIAN_RED 51947
#define BROWN 41285
#define FIRE_BRICK 45316
#define MAROON 32768
#define DARK_RED 34816
#define STRONG_RED 57344
#define RED 63488
#define PERSIMMON 64104
#define MISTY_ROSE 65340
#define SALMON 64526
#define SCARLET 63776
#define TOMATO 64264
#define DARK_SALMON 60591
#define CORAL 64490
#define ORANGE_RED 64032
#define LIGHT_SALMON 64783
#define VERMILION 64096
#define SIENNA 41605
#define TROPICAL_ORANGE 64518
#define CAMEL 41800
#define APRICOT 58572
#define COCONUT_BROWN 18656
#define SEASHELL 65469
#define SADDLE_BROWN 35362
#define CHOCOLATE 54083
#define BURNT_ORANGE 51872
#define SUN_ORANGE 64384
#define PEACH_PUFF 65239
#define SAND_BROWN 62764
#define BRONZE 48006

```



```

#define LINEN 65436
#define HONEY_ORANGE 64908
#define PERU 52263
#define SEPIA 29186
#define OCHER 52132
#define BISQUE 65336
#define TANGERINE 62496
#define DARK_ORANGE 64608
#define ANTIQUE_WHITE 65370
#define TAN 54705
#define BURLY_WOOD 56784
#define BLANCHED_ALMOND 65369
#define NAVAJO_WHITE 65269
#define MARIGOLD 64704
#define PAPAYA_WHIP 65402
#define PALE_OCRE 52625
#define KHAKI 39747
#define MOCCASIN 65334
#define OLD_LACE 65468
#define WHEAT 63222
#define PEACH 65334
#define ORANGE 64800
#define FLORAL_WHITE 65502
#define GOLDENROD 56612
#define DARK_GOLDENROD 48161
#define COFFEE 18880
#define JASMINE 58891
#define AMBER 64992
#define CORNSILK 65499
#define CHROME_YELLOW 58816
#define GOLDEN 65184
#define LEMON_CHIFFON 65497
#define LIGHT_KHAKI 63281
#define PALE_GOLDENROD 61269
#define DARK_KHAKI 48557
#define MIMOSA 59078
#define CREAM 65530
#define IVORY 65534
#define BEIGE 63419
#define LIGHT_YELLOW 65532
#define LIGHT_GOLDENROD_YELLOW 65498
#define CHAMPAGNE_YELLOW 65523
#define MUSTARD 52841

```

```

#define MOON_YELLOW 65513
#define OLIVE 33792
#define CANARY_YELLOW 65504
#define YELLOW 65504
#define MOSS_GREEN 27556
#define LIGHT_LIME 53216
#define OLIVE_DRAB 27748
#define YELLOW_GREEN 40550
#define DARK_OLIVE_GREEN 21317
#define APPLE_GREEN 36640
#define GREEN_YELLOW 45029
#define GRASS_GREEN 40745
#define LAWN_GREEN 32736
#define CHARTREUSE 32736
#define FOLIAGE_GREEN 30151
#define FRESH_LEAVES 40937
#define BRIGHT_GREEN 26592
#define COBALT_GREEN 26603
#define HONEYDEW 63486
#define DARK_SEA_GREEN 36337
#define LIGHT_GREEN 38770
#define PALE_GREEN 40915
#define IVY_GREEN 13798
#define FOREST_GREEN 9284
#define LIME_GREEN 13926
#define DARK_GREEN 800
#define GREEN 1024
#define LIME 2016
#define MALACHITE 9733
#define MINT 5317
#define CELADON_GREEN 30513
#define EMERALD 22095
#define TURQUOISE_GREEN 20272
#define VERIDIAN 5030
#define HORIZON_BLUE 43001
#define SEA_GREEN 11338
#define MEDIUM_SEA_GREEN 15758
#define MINT_CREAM 63487
#define SPRING_GREEN 2032
#define PEACOCK_GREEN 1291
#define MEDIUM_SPRING_GREEN 2003
#define MEDIUM_AQUAMARINE 26229
#define AQUAMARINE 32762

```

```

#define CYAN_BLUE 3569
#define AQUA_BLUE 26620
#define TURQUOISE_BLUE 14137
#define TURQUOISE 14009
#define LIGHT_SEA_GREEN 9621
#define MEDIUM_TURQUOISE 20121
#define LIGHT_CYAN 59391
#define BABY_BLUE 59391
#define PALE_TURQUOISE 44925
#define DARK_SLATE_GRAY 10857
#define TEAL 1040
#define DARK_CYAN 1105
#define CYAN 2047
#define AQUA 44796
#define DARK_TURQUOISE 1658
#define CADET_BLUE 23796
#define PEACOCK_BLUE 1041
#define POWDER_BLUE 46876
#define STRONG_BLUE 782
#define LIGHT_BLUE 44764
#define PALE_BLUE 32217
#define SAXE_BLUE 17622
#define DEEP_SKY_BLUE 1535
#define SKY_BLUE 34429
#define LIGHT_SKY_BLUE 34431
#define MARINE_BLUE 559
#define PRUSSIAN_BLUE 394
#define STEEL_BLUE 17430
#define ALICE_BLUE 63455
#define SLATE_GRAY 29714
#define LIGHT_SLATE_GRAY 29779
#define DODGER_BLUE 7327
#define MINERAL_BLUE 627
#define AZURE 1023
#define WEDGWOOD_BLUE 21559
#define LIGHT_STEEL_BLUE 46651
#define COBALT_BLUE 565
#define PALE_DENIM 23608
#define CORNFLOWER_BLUE 25789
#define SALVIA_BLUE 19484
#define DARK_POWDER_BLUE 403
#define SAPPHIRE 2348
#define INTERNATIONAL_KLEIN_BLUE 372

```

```

#define CERULEAN_BLUE 10903
#define ROYAL_BLUE 17244
#define DARK_MINERAL_BLUE 8623
#define ULTRAMARINE 415
#define LAPIS_LAZULI 2463
#define GHOST_WHITE 65503
#define LAVENDER 59199
#define PERIWINKLE 52863
#define MIDNIGHT_BLUE 6350
#define NAVY_BLUE 16
#define DARK_BLUE 17
#define MEDIUM_BLUE 25
#define BLUE 31
#define WISTERIA 23196
#define DARK_SLATE_BLUE 18929
#define SLATE_BLUE 27353
#define MEDIUM_SLATE_BLUE 31581
#define MAUVE 25119
#define LILAC 46303
#define MEDIUM_PURPLE 37787
#define AMETHYST 24985
#define GRAYISH_PURPLE 33716
#define HELIOTROPE 20503
#define MINERAL_VIOLET 48409
#define BLUE_VIOLET 35164
#define VIOLET 34847
#define INDIGO 18448
#define DARK_ORCHID 39321
#define DARK_VIOLET 36890
#define PANSY 28692
#define MALLOW 55935
#define OPERA_MAUVE 58399
#define MEDIUM_ORCHID 47802
#define PALE_LILAC 59004
#define THISTLE 56827
#define CLEMATIS 52505
#define PLUM 56603
#define LIGHT_VIOLET 60445
#define PURPLE 32784
#define DARK_MAGENTA 34833
#define MAGENTA 63519
#define FUCHSIA 61460
#define ORCHID 56218

```

```

#define PEARL_PINK 64924
#define OLD_ROSE 47795
#define ROSE_PINK 64313
#define MEDIUM_VIOLET_RED 49328
#define MAGENTA_ROSE 63604
#define ROSE 63503
#define RUBY 51216
#define CAMELLIA 57810
#define DEEP_PINK 63666
#define FLAMINGO 58455
#define CORAL_PINK 64535
#define HOT_PINK 64342
#define BURGUNDY 16388
#define SPINEL_RED 64406
#define CARMINE 57355
#define BABY_PINK 65244
#define CARDINAL_RED 38918
#define LAVENDER_BLUSH 65438
#define PALE_VIOLET_RED 56210
#define CERISE 55692
#define SALMON_PINK 64531
#define CRIMSON 55463
#define PINK 65049
#define LIGHT_PINK 64952
#define SHELL_PINK 64919
#define ALIZARIN_CRIMSON 57638

```

```

#endif

```

8. create.h

```

#ifndef _CREATE_H_
#define _CREATE_H_

/*typedef struct{
    int type[40][28];
    int zone=0;
}RandomType;*/

extern void Map_Create(int x0,int y0,int zone);
extern void Urban_Building_Create(int x0,int y0,int i,int j,int (*draw_done)[28]);
extern int Main_Road_Convert(int point,int forward,int mode);
extern void Urban_Road_Create(int x0,int y0,int i,int j,int (*dra

```

```

w_done)[28]);
extern int Added_Road_Convert(int point,int forward,int mode);
extern int Narrow_Road_Convert(int point,int forward,int mode);
extern void Urban_Tree_Create(int x0,int y0,int i,int j,int (*draw
w_done)[28]);
extern void Suburban_Road_Create(int x0,int y0,int i,int j,int (*
draw_done)[28]);
extern int Suburban_Road_Convert(int point,int forward,int mode);
extern void Path_Create(int x0,int y0,int i,int j,int (*draw_done
)[28]);
extern void Suburban_Tree_Create(int x0,int y0,int i,int j,int (*
draw_done)[28]);
extern int Scan_Small_Crossing(int i,int j);
extern int Scan_Narrow_Road(int i,int j,int forward);
extern int Scan_Edge(int x,int y);
extern void Urban_Bridge_Create(int x0,int y0,int i,int j,int (*d
raw_done)[28]);
extern int Scan_Bridge(int i,int j,int forward);
extern int Scan_Edge_Road(int i,int j,int forward);
extern int Scan_Bridge_Crossing(int i,int j);
extern void Erase_Map();
extern void Suburban_Bridge_Create(int x0,int y0,int i,int j,int
(*draw_done)[28]);
extern int Scan_Sub_Bridge_Crossing(int i,int j);
extern int Field_Convert(int x0,int y0,int x,int y,int mode);
extern void Mountainous_Bridge_Create(int x0,int y0,int i,int j,i
nt (*draw_done)[28]);
extern int Scan_Mountainous_Bridge(int i,int j);
extern void Flag_Create(int x[],int y[],int point);

#endif

```

9. find.h

```

#ifndef __find_h__
#define __find_h__

extern void Direction_Calculate(int people,int zone,int weather,i
nt point,int* point_x,int* point_y,int **direction,people_basic *
person_basic,people_competition *person_competition,int your_numb
er,int flag_gui,int flag_debug);
extern void Direction_Optimize(int zone,int* direction,int x,int
y,int t,int h,int j);
extern void Findpath(int flag,int point,int* point_x,int* point_y

```

```
,int *direction,int *addedpoint_x,int *addedpoint_y,int flag_gui,
int flag_debug);
extern void Findpath1(int point,int* point_x,int* point_y,int *di
rection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag
_debug);
extern void Findpath2(int point,int* point_x,int* point_y,int *di
rection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag
_debug);
extern void Findpath3(int point,int* point_x,int* point_y,int *di
rection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag
_debug);
extern void Findpath4(int point,int* point_x,int* point_y,int *di
rection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag
_debug);
extern void Findpath5(int point,int* point_x,int* point_y,int *di
rection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag
_debug);
extern void Findpath6(int point,int* point_x,int* point_y,int *di
rection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag
_debug);

#endif
```

10. find1.h

```
#ifndef __find1_h__
#define __find1_h__

typedef int      BOOL;

typedef struct   Node      //node 结构体
{
    int f,g,h;           //f(总代价)=g(初始代价)+h(曼哈顿距离)
    int X;               //节点横坐标
    int Y;               //节点纵坐标
    struct Node* parent; //父节点Node 结构体首地址
}Node,*Lnode;

typedef struct Stack      //Open Closed 表结构体
{
    Node* npoint;         //节点Node 结构体首地址
    struct Stack* next;   //下一节点结构体首地址
}Stack,*Lstack;
```

```

extern Lnode getFminNodeFromOpen();
extern Lnode getGminNodeFromOpen();
extern Lnode BelongInOpen(int X, int Y);
extern BOOL BelongInClosed(int X, int Y);
extern void PutintoOpen(Lnode node);
extern void PutintoClosed(Lnode node);
extern int getH(int X,int Y,int destinationX,int destinationY);
extern BOOL isCanMove(int X,int Y);
extern void creatSeccessionNode(Lnode parentNode,int X,int Y,int
destinationX, int destinationY,int gPlus);
extern void seachSeccessionNode(Lnode parentNode, int destination
X, int destinationY);
extern void creatSeccessionNodeG(Lnode parentNode,int X,int Y,int
gPlus);
extern void seachSeccessionNodeG(Lnode parentNode);
extern void cleanStack();
extern void getPath(int startX,int startY,int endX,int endY,int *
direction,int *step,int flag_gui);
extern void SearchAll(int startX,int startY,int flag_gui);
extern void Searchpath(int startX,int startY,int endX,int endY,in
t *direction,int *step,int flag_gui);
extern void Findpath0(int people,int zone,int weather,int point,i
nt* point_x,int* point_y,int *direction,int *addedpoint_x,int *ad
dedpoint_y,int flag_gui,int flag_debug);

extern Lstack Open;
extern Lstack Closed;

#endif

```

11. gins.h

```

#ifndef __gins_h__
#define __gins_h__

struct P_oint{
    int x;
    int y;
};
extern void Bar_Random_Spot(int x0,int y0,int x1,int y1,int numbe
r,int color);
extern void Circle_Random_Spot(int x0,int y0,int radius,int numbe
r,int color);
extern void Diswap(int *a,int *b);

```



```

extern void Hollow_Ellipse(double x,double y,double a,double b,int
t color);
extern void Hollow_Quadrangle(int x0,int y0,int x1,int y1,int x2,
int y2,int x3,int y3,int color);
extern void Long_Diswap(long int *a,long int *b);
extern void Solid_Ellipse(double x,double y,double a,double b,int
color);
extern void Solid_HalfSector_down(int x,int y,int r,int color);
extern void Solid_HalfSector_up(int x,int y,int r,int color);
extern void Solid_Quadrangle(int x0,int y0,int x1,int y1,int x2,i
nt y2,int x3,int y3,int color);
extern void Solid_Triangle(int x0,int y0,int x1,int y1,int x2,int
y2,int color);
extern void Solid_Upper_Ellipse(double x,double y,double a,double
b,int color);
extern void Triangle_Random_Spot(int x0,int y0,int x1,int y1,int
x2,int y2,int number,int color);
extern int Trimax(int a,int b,int c);
extern int Trimin(int a,int b,int c);
extern void YMAX_Array_Sort(struct P_ooint *p,int length);
extern void YMAX_Triswap(int *x0,int *y0,int *x1,int *y1,int *x2,
int *y2);
extern void Ring(int x,int y,int R1,int R2,int color);

#endif

```

12. graph.h

```

#ifndef __gragh_h__
#define __graph_h__

extern void Line_Plus(int x0,int y0,int x1,int y1,int color);
extern void Square(int x0,int y0,int length,int color);
extern void Hollow_Bar(int x0,int y0,int x1,int y1,int color);
extern void Solid_Bar(int x0,int y0,int x1,int y1,int color);
extern void Hollow_Triangle(int x0,int y0,int x1,int y1,int x2,in
t y2,int color);
extern void Hollow_Circle(int xc,int yc,int radius,int color);
extern void Solid_Circle(int xc, int yc, int radius, int color);

#define Hollow_Color        DARK_GRAY
#define Solid_Color         LIGHT_GRAY
extern void Button_Draw(int x0,int y0,int x1,int y1,int ext_color
,int int_color);

```

```

extern void Button_Edge(int x0,int y0,int x1,int y1,int color);

extern void UpperBar_Draw(int thick,int edge,int solid_color,int
edge_color);
extern void Boundary_Draw(int x0,int y0,int thick,int color);
extern void RevertImage_Draw(int x0,int y0,int color);
extern void RevertButton_Draw(int x0,int y0,int color);

extern void SignEdge_Draw(int x0,int y0,int color);
extern void AnimationEdge_Draw(int x0,int y0,int color);

extern void TriangleImage(int x0, int y0, int color);

#endif

```

13. headfile.h

```

/*****
*****
* @author          ytm,lxj
* @function        add all headfiles and bios key values
* @notice          this project is based on SVGA, so needn't <gr
aphics.h>
*****
*****/

#ifndef _headfile_h
#define _headfile_h

//add all system headfiles
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<bios.h>
#include<conio.h>
#include<dos.h>
#include<time.h>
#include<dir.h>
#include<io.h>
#include<alloc.h>
#include<malloc.h>

//add reference headflies

```

```
#include"mouse.h"
#include"color.h"
#include"svga.h"

//add ytm headfiles
#include"hz.h"
#include"welcome.h"
#include"quit.h"
#include"loading.h"
#include"menu.h"
#include"help.h"
#include"setting.h"
#include"graph.h"
#include"setting1.h"
#include"setting3.h"
#include"msgbox.h"
#include"play.h"
#include"search.h"
#include"sign.h"
#include"asc.h"
#include"clock.h"
#include"setting2.h"
#include"map.h"
#include"people.h"
#include"advance.h"
#include"find.h"
#include"find1.h"
#include"about.h"
```

```
//add lxj headfiles
#include "icon.h"
#include "gins.h"
#include "create.h"
#include "move.h"
#include "login.h"
#include "register.h"
#include "animate.h"
#include "info.h"
#include "weather.h"
#include "back.h"
#include "surface.h"
#include "universe.h"
```

```

//add key values
#define F1 0x3b00
#define F2 0x3c00
#define F3 0x3d00
#define F4 0x3e00
#define F5 0x3f00
#define F6 0x4000
#define F7 0x4100
#define F8 0x4200
#define F9 0x4300
#define F10 0x4400
#define ENTER 0x1c0d
#define BACK 0x0e08
#define ESC 0x011b
#define UP 0x4800
#define DOWN 0x5000
#define RIGHT 0x4d00
#define LEFT 0x4b00

//add TURE FALSE for debugging
#define TRUE 1
#define FALSE 0

//mouse.h
extern void cursor(int x, int y);
extern void mousehide(int x, int y);
extern int init();
extern void mouseInit(int *mx, int *my, int *mbutt);
extern int readxy(int *mx, int *my, int *mbutt);
extern void newxy(int *mx, int *my, int *mbutt);
extern int mouse_press(int x1, int y1, int x2, int y2);
extern int mouse_out_press(int x1, int y1, int x2, int y2);
extern void resetMouse(int mx,int my);

extern int MouseX;
extern int MouseY;
extern int press;
extern int shape;

#endif

```

14. help.h

```
#ifndef __help_h__
#define __help_h__

extern void Help(int *page);
extern void Draw_Help();
extern void Button_Light_Help(int flag);
extern void Button_Darken_Help(int flag);
extern void Button_Press_Help(int flag);
extern void Message_Light_Help(int flag);
extern void MessageSign_Light_Help(int flag);
extern void Button_Darken_Verb(int flag);

#endif
```

15. hz.h

```
#ifndef __hz_h__
#define __hz_h__

extern void puthz(int x, int y,int flag,int part,int color,char *
s);

#endif
```

16. icon.h

```
/******
Creator: 梁忻健
Last updating time:2022.3.9 22:48
*****/
#ifndef __icon_h__
#define __icon_h__

#define BODYCOLOR BURLY_WOOD
#define U_P 1
#define D_DOWN 2
#define L_LEFT 3
#define R_RIGHT 4
#define H_ORIZONTAL 5
#define V_VERTICAL 6
#define C_CENTER 7

extern void Draw_Bush_Greenball(int x,int y);
extern void Draw_Bush_Orangeball(int x,int y);
```

```

extern void Draw_Bush_Redball(int x,int y);
extern void Draw_Child(int x0,int y0,int color);
extern void Draw_Cloud(int x0,int y0,int color);
extern void Draw_Compass(int x0,int y0);
extern void Draw_Flower(int x,int y,int color);
extern void Draw_Grass(int x,int y);
extern void Draw_H_Narrow_Highway(int x,int y,int length);
extern void Draw_H_Narrow_Highway_Part(int x,int y);
extern void Draw_Leaf_1(int x0,int y0,int radius,int color);
extern void Draw_Leaf_2(int x0,int y0,int radius,int color);
extern void Draw_Mid(int x0,int y0,int color);
extern void Draw_Mountainous_Bush(int x,int y);
extern void Draw_Mountainous_Grass(int x,int y);
extern void Draw_Mountainous_House(int x,int y);
extern void Draw_Mountainous_Lake(int x,int y);
extern void Draw_Mountainous_Swamp(int x,int y);
extern void Draw_Mountainous_Tree(int x,int y);
extern void Draw_Narrow_Bridge_Connection(int x,int y,int forward
);
extern void Draw_Narrow_Bridge_End(int x,int y,int forward);
extern void Draw_Old(int x0,int y0,int color);
extern void Draw_RainDrop(int x0,int y0,int color);
extern void Draw_Rainy(int x0,int y0,int color);
extern void Draw_Random_Single_Window(int x,int y,int sizex,int s
izey);
extern void Draw_Random_Window(int x,int y,int length,int heigh,i
nt block);
extern void Draw_River(int x0,int y0,int x1,int y1);
extern void Draw_Store_Window(int x,int y,int length,int forward)
;
extern void Draw_Suburban_Appletree(int x,int y);
extern void Draw_Suburban_Bush(int x,int y);
extern void Draw_Suburban_Field(int x,int y);
extern void Draw_Suburban_Grass(int x,int y);
extern void Draw_Suburban_House(int x,int y);
extern void Draw_Suburban_H_Narrow_Bridge(int x,int y,int length)
;
extern void Draw_Suburban_H_Path(int x,int y,int length);
extern void Draw_Suburban_Path_Part(int x,int y,int mode);
extern void Draw_Suburban_Sakura(int x,int y);
extern void Draw_Suburban_V_Narrow_Bridge(int x,int y,int length)
;
extern void Draw_Suburban_V_Path(int x,int y,int length);

```

```

extern void Draw_Sun(int x0,int y0,int color);
extern void Draw_Swamp_Wave(int x,int y);
extern void Draw_Teen(int x0,int y0,int color);
extern void Draw_Urban_Flower(int x,int y);
extern void Draw_Urban_Hospital(int x,int y);
extern void Draw_Urban_House(int x,int y,int color);
extern void Draw_Urban_H_Wide_Bridge(int x,int y,int length);
extern void Draw_Urban_H_Wide_Highway(int x,int y,int length);
extern void Draw_Urban_H_Wide_Highway_Part(int x,int y);
extern void Draw_Urban_Lake(int x,int y);
extern void Draw_Urban_Store(int x,int y,int color);
extern void Draw_Urban_Tree(int x,int y);
extern void Draw_Urban_V_Wide_Bridge(int x,int y,int length);
extern void Draw_Urban_V_Wide_Highway(int x,int y,int length);
extern void Draw_Urban_V_Wide_Highway_Part(int x,int y);
extern void Draw_V_Narrow_Highway(int x,int y,int length);
extern void Draw_V_Narrow_Highway_Part(int x,int y);
extern void Draw_Wave(int x,int y);
extern void Draw_Wide_Bridge_End(int x,int y,int forward);
extern void Draw_Wide_Bridge_Part(int x,int y,int forward);
extern void Draw_Window(int x,int y);
extern void Draw_Urban_House_Icon(int x,int y);
extern void Draw_Random_Window_Icon(int x,int y,int length,int height,int block);
extern void Draw_River_Icon(int x,int y);
extern void Draw_Urban_Bridge_Icon(int x,int y);
extern void Draw_Mountain(int x,int y);
extern void Draw_Urban_Midhouse(int x,int y,int color1,int color2);
extern void Draw_Mountainous_On_Grass(int x,int y);
extern void Draw_Small_Crossing(int x,int y);
extern void Draw_Main_Crossing(int x,int y);
extern void Draw_Small_Crossing(int x,int y);
extern void Draw_Choose(int x,int y,int done);
extern void InfoImage_Draw(int x,int y,int color);
extern void Draw_On_Mountainous_Grass(int x,int y);
extern void Draw_Teen_Icon(int x0,int y0,int color);
extern void Draw_Old_Icon(int x0,int y0,int color);
extern void Draw_Child_Icon(int x0,int y0,int color);
extern void Draw_Mid_Icon(int x0,int y0,int color);
extern void LoginImage_Draw(int x,int y,int color);
extern void Draw_Arrow(int x,int y,int direction,int color);

```

```
extern void Draw_Flag(int x,int y);
#endif
```

17. info.h

```
#ifndef _INFO_H_
#define _INFO_H_

#define CHILD 1
#define TEEN 2
#define MID 3
#define OLD 4

extern void Info_Point(int x,int y,int finish,int point);
extern void Info_Time(int x,int y,int hour,int minute);
extern void Info_Power(int x,int y,int power);
extern void Info_People(int x,int y,int finish,int point,int hour
,int minute,int power,int age,int color);
extern void Info(int finish[],int point,int hour[],int minute[],i
nt power[],int age,int color[],int number);
extern void Time_Clac(int *hour,int *minute,people_competition pe
rson_competition,int clock_hour,int clock_minute);
extern void Distance_Clac(int map[],int **direction,int map_now[]
,int people);
extern void Info_Player(int your_number);

#endif
```

18. loading.h

```
#ifndef _loading_h
#define _loading_h

typedef struct people1
{
    int number;    //序号
    int ps_consume;    //体力剩余值
    int ps_recover;    //体力恢复值
    int color;    //人物衣服颜色
    double verb_walk;    //人物走路速度
    double verb_run;    //人物跑步速度
    double verb_climb;    //人物爬山速度
}people_basic;
typedef struct people2
```



```

{
    int distance;    //人物运动距离
    int hour_point[4];    //点位到达小时数
    int minute_point[4];    //点位到达分钟数
    int rank;        //排名
    int hour_interval[4];    //点位间隔小时数
    int minute_interval[4];    //点位间隔分钟数
    int hour_score;    //成绩小时数
    int minute_score;    //成绩分钟数
}people_competition;

extern void Loading_Welcome(int x0,int y0,int time_ms);
extern void Loading_Play(int x0,int y0,int people,int age,int zone,int weather,int point,int* point_x,int* point_y,int* clock_hour,int* clock_minute,int *clock_hour_start, int *clock_minute_start , people_basic *person_basic, people_competition *person_competition);
extern void Loading_Search(int x0,int y0,int people,int age,int zone,int weather,int point,int *clock_hour_end, int *clock_minute_end, people_basic *person_basic, people_competition *person_competition);

#endif

```

19. login.h

```

#ifndef _LOGIN_H_
#define _LOGIN_H_

typedef struct{
    char Username[11];
    char Password[11];
}USER;

extern void Login(int *page,char *your_name);
extern void Login_Check();
extern int Scan_Login(char* user_buffer,char* pass_buffer);
extern void Create_Admin();
extern char Login_State(char i);

#endif

```

20. map.h

```
#ifndef _map_h
#define _map_h

extern void Map_Random_Zone1(int* x0,int *y0,int flag_gui);
extern void Map_Random_Zone2(int flag_gui);
extern void Map_Random_Zone3(int flag_gui);
extern void Map_Random(int zone,int point,int* point_x,int* point_y,int weather,int flag_gui,int flag_debug);
extern void Point_Random(int zone,int point,int* point_x,int* point_y,int x,int y,int flag_gui,int flag_debug);
extern void Map_Analysis(int zone,int weather);
extern void Map_Display(int x0,int y0,int zone,int weather,int point,int* point_x,int* point_y,int flag_gui,int flag_debug,int clock_hour);

extern int map_process[40][28];
extern int map_display[40][28];

#endif
```

21. menu.h

```
#ifndef _menu_h
#define _menu_h

extern void Menu(int *page);
extern void Draw_Menu();
extern void Button_Light_Menu(int flag);
extern void Button_Darken_Menu(int flag);
extern void CopyrightText_Draw(int x0,int y0,int color,int bk_color);

#endif
```

22. mouse.h

```
#ifndef _mouse_h
#define _mouse_h

extern void getMousebk(int x,int y);
extern void setMousebk(int color);
extern void changeMousebk(int fromColor,int toColor);
extern void backgroundChange(int mx, int my, int x1, int y1, int x2, int y2);
```

```
extern void AddFrame(int mx, int my, int x1, int y1, int x2, int
y2, int thick, int color);
```

```
#endif
```

23. move.h

```
#ifndef _MOVE_H_
#define _MOVE_H_
```

```
#define WALK 1
```

```
#define RUN 2
```

```
#define CLIMB 3
```

```
#define MOVE_UP 1
```

```
#define MOVE_DOWN 2
```

```
#define MOVE_LEFT 3
```

```
#define MOVE_RIGHT 4
```

```
#define MOVE_LEFTUP 5
```

```
#define MOVE_LEFTDOWN 6
```

```
#define MOVE_RIGHTUP 7
```

```
#define MOVE_RIGHTDOWN 8
```

```
extern void Teen_Movement(int x,int y,char forward,char move,char
frame,int color);
```

```
extern void Mid_Movement(int x,int y,char forward,char move,char
frame,int color);
```

```
extern void Old_Movement(int x,int y,char forward,char move,char
frame,int color);
```

```
extern void Child_Movement(int x,int y,char forward,char move,cha
r frame,int color);
```

```
#endif
```

24. msgbox.h

```
#ifndef _msgbox_h
```

```
#define _msgbox_h
```

```
extern void MsgBox2_Draw_Setting();
```

```
extern void MsgBox2Button_Light(int flag);
```

```
extern void MsgBox2Button_Darken(int flag);
```

```
extern void MsgBox0_Draw_Play();
```

```
extern void MsgBox0_Draw1_Play();
```

```
extern void MsgBox2_Draw_Search();
```

```

extern void MsgBox0NoInput_Draw_Search();
extern void MsgBox0NoSearch_Draw_Search();

#endif

```

25. people.h

```

#ifndef __people_h__
#define __people_h__

/*
typedef struct people1
{
    int number;        //序号
    int ps_consume;    //体力剩余值
    int ps_recover;    //体力恢复值
    int color;         //人物衣服颜色
    double verb_walk;  //人物走路速度
    double verb_run;   //人物跑步速度
    double verb_climb; //人物爬山速度
}people_basic;
typedef struct people2
{
    int distance;      //人物运动距离
    int hour_point[4]; //点位到达小时数
    int minute_point[4]; //点位到达分钟数
    int rank;          //排名
    int hour_interval[4]; //点位间隔小时数
    int minute_interval[4]; //点位间隔分钟数
    int hour_score;     //成绩小时数
    int minute_score;   //成绩分钟数
}people_competition;
//store people's competition data

//notice: edit them in loading.h
*/

extern void People_Init(people_basic *person_basic,people_competition *person_competition);
extern void People_Random(int people,int age,int weather,int clock_hour,int clock_minute,people_basic *person_basic,people_competition *person_competition,int flag_debug);
extern void People_Interval(int people,int point,people_basic *person_basic,people_competition *person_competition);

```

```

extern void People_Score(int people,int point,people_basic *person_
n_basic,people_competition *person_competition);
extern void People_Rank(int people,people_basic *person_basic,peo
ple_competition *person_competition);

#endif

```

26. play.h

```

#ifndef __play_h__
#define __play_h__

extern void Play(int *page,int *people,int *age,int *zone,int *we
ather,int *point,int* point_x,int* point_y,int* clock_hour,int *c
lock_minute,int *clock_hour_end,int *clock_minute_end,people_basi
c *person_basic,people_competition *person_competition,int* your_
number);
extern void Draw_Play(int age,int people);
extern void Title_Draw(int age,int zone);
extern void TitleZone_Draw(int zone);
extern void TitleAge_Draw(int age);
extern int Check_Direciton(int last_i,int last_j,int i,int j);
extern int Check_Distance(int last_i,int last_j,int i,int j,int f
lag_gui);
extern int Check_Availablility(int last_i,int last_j,int i,int j)
;
extern int Check_Point(int i,int j,int last_point,int point,int*
point_x,int *point_y);
extern int Min2(int a,int b);
extern int Max2(int a,int b);
extern void BlockColor_Change(int i,int j,int color);
extern void Choose_Your_Way(int zone,int weather,int point,int *p
oint_x,int *point_y,int *direction,people_basic *person_basic, pe
ople_competition *person_competition,int your_number,int flag_gui
,int flag_debug);

#endif

```

27. quit.h

```

#ifndef _quit_h
#define _quit_h

extern void Quit(int *page);
extern void Draw_Quit();

```

```
extern void Button_Light_Quit(int flag);
extern void Button_Darken_Quit(int flag);
```

```
#endif
```

28. register.h

```
#ifndef _REGISTER_H_
#define _REGISTER_H_

extern void Register(int *page);
extern void Register_Check(int *page);
extern void New_User(char* username,char* password);
extern int Scan_reRegister(int usernum,char* username);
extern int Scan_Confirm(char* pass_buffer,char* confirm_buffer);
extern int Scan_Nullinput(char* buffer);

#endif
```

29. search.h

```
#ifndef __search_h__
#define __search_h__

extern void Draw_Search();
extern void MsgBox2Button_Light_Search(int flag);
extern void MsgBox2Button_Darken_Search(int flag);
extern void Button_Light_Search(int flag);
extern void Button_Darken_Search(int flag);
extern void Button_Press_Search(int flag);
extern void Message_Title_Draw(int x0,int y0,int age,int zone);
extern void UserMessage_Light_Search(int people,int age,int zone,
int point,people_basic *person_basic,people_competition *person_c
ompetition,int your_number,int *your_name);
extern void GenMessage_Light_Search(int people,int age,int zone,i
nt clock_hour_start,int clock_minute_start,people_basic *person_b
asic,people_competition *person_competition);
extern void PerMessage_Light_Search(int flag,int numorrnk,int pe
ople,int age,int zone,int point,people_basic *person_basic,people
_competition *person_competition);
extern void ChooseButton_Light_Per(int flag);
extern void ChooseButton_Darken_Per(int flag,int flag_choose);
extern void ChooseButton_Press_Per(int flag);
extern void PerPage_Light_Search(int flag);
extern void ButtonNumRank_Light_Search(int flag);
```

```

extern void ButtonNumRank_Darken_Search(int flag);
extern void Search(int *page,int *people,int *age,int *zone,int *
weather,int *point,int clock_hour_start,int clock_minute_start,in
t clock_hour_end,int clock_minute_end,people_basic *person_basic,
people_competition *person_competition,int* your_number,char *you
r_name);

```

```

#endif

```

30. setting.h

```

#ifndef __setting_h__
#define __setting_h__

extern void Setting(int *page,int *people, int *age, int *zone, i
nt *weather,int *point,int* point_x,int* point_y,int* clock_hour,
int* clock_minute,int *clock_hour_start,int *clock_minute_start,p
eople_basic *person_basic,people_competition *person_competition)
;
extern void PageLeft_Draw_Setting();
extern void ButtonLeft_Light_Setting(int flag);
extern void ButtonLeft_Darken_Setting(int flag);
extern void ButtonLeft_Press_Setting(int flag);

//driver
#define Revert_x0    5
#define Revert_y0    5

#define Start_x0      12
#define Start_y0      70
/*
#define Text_xplus    160
#define Text_yplus    24
*/

//follower
#define Revert_x1    (Revert_x0+39)
#define Revert_y1    (Revert_y0+39)

#define Start_x1      (Start_x0+383)
#define Start_y1      (Start_y0+85)

#define Text1_x0      12
#define Text1_y0      (Start_y0+99)

```

```

#define People_x0      12
#define People_y0      (Start_y0+145)
#define People_x1      (Start_x0+383)
#define People_y1      (Start_y0+231)
#define Age_x0         12
#define Age_y0         (Start_y0+237)
#define Age_x1         (Start_x0+383)
#define Age_y1         (Start_y0+323)

#define Text2_x0       12
#define Text2_y0       (Start_y0+337)
#define Zone_x0        12
#define Zone_y0        (Start_y0+383)
#define Zone_x1        (Start_x0+383)
#define Zone_y1        (Start_y0+469)
#define Weather_x0     12
#define Weather_y0     (Start_y0+475)
#define Weather_x1     (Start_x0+383)
#define Weather_y1     (Start_y0+562)
#define Point_x0       12
#define Point_y0       (Start_y0+568)
#define Point_x1       (Start_x0+383)
#define Point_y1       (Start_y0+654)

//driver
#define NumMinus_x0     429
#define NumMinus_y0     70

//follower
#define NumMinus_x1     (NumMinus_x0+187)
#define NumMinus_y1     (NumMinus_y0+85)
#define PeopleNum_x0    (NumMinus_x0+193)
#define PeopleNum_y0    70
#define PeopleNum_x1    (NumMinus_x0+381)
#define PeopleNum_y1    (NumMinus_y0+85)
#define NumPlus_x0      (NumMinus_x0+387)
#define NumPlus_y0      70
#define NumPlus_x1      (NumMinus_x0+575)
#define NumPlus_y1      (NumMinus_y0+85)
/*
#define Text_xplus      62
#define Text_yplus      27
*/

```



```

//driver
#define PointMinus_x0      429
#define PointMinus_y0      70

//follower
#define PointMinus_x1      (PointMinus_x0+187)
#define PointMinus_y1      (PointMinus_y0+85)
#define PointNum_x0        (PointMinus_x0+193)
#define PointNum_y0        70
#define PointNum_x1        (PointMinus_x0+381)
#define PointNum_y1        (PointMinus_y0+85)
#define PointPlus_x0        (PointMinus_x0+387)
#define PointPlus_y0        70
#define PointPlus_x1        (PointMinus_x0+575)
#define PointPlus_y1        (PointMinus_y0+85)

//driver
#define General_x0 344
#define General_y0 230
#define General_x1 680
#define General_y1 610

//follower
#define Message_x0 (General_x0+8)
#define Message_y0 (General_y0+19)
#define Message_x1 (General_x0+326)
#define Message_y1 (General_y0+191)
#define Confirm_x0 (General_x0+8)
#define Confirm_y0 (General_y0+197)
#define Confirm_x1 (General_x0+326)
#define Confirm_y1 (General_y0+282)
#define Cancel_x0  (General_x0+8)
#define Cancel_y0  (General_y0+286)
#define Cancel_x1  (General_x0+326)
#define Cancel_y1  (General_y0+371)

#endif

```

31. setting1.h

```

#ifndef __setting1_h__
#define __setting1_h__

```

```

extern void ButtonRight_Draw_Setting1(int *people);
extern void ButtonPeopleNum_Light_Setting1(int flag);
extern void ButtonPeopleNum_Darken_Setting1(int flag);
extern void PeopleNum_Draw_Setting1(int people,int color);

```

```

#endif

```

32. setting2.h

```

#ifndef __setting2_h__
#define __setting2_h__

```

```

extern void ButtonRight_Draw_Setting345(int flag);
extern void ButtonRight3_Light_Setting345(int flag);
extern void ButtonRight3_Darken_Setting345(int flag);
extern void ButtonRight3_Press_Setting345(int flag);
extern void ButtonRight4_Light_Setting345(int flag);
extern void ButtonRight4_Darken_Setting345(int flag);
extern void ButtonRight4_Press_Setting345(int flag);
extern void ButtonRight5_Light_Setting345(int flag);
extern void ButtonRight5_Darken_Setting345(int flag);
extern void ButtonRight5_Press_Setting345(int flag);

```

```

#endif

```

33. setting3.h

```

#ifndef __setting3_h__
#define __setting3_h__

```

```

extern void ButtonRight_Draw_Setting6(int *point);
extern void ButtonPointNum_Light_Setting6(int flag);
extern void ButtonPointNum_Darken_Setting6(int flag);
extern void PointNum_Draw_Setting6(int point,int color);

```

```

#endif

```

34. sign.h

```

#ifndef __sign_h__
#define __sign_h__

```

```

extern void ChooseButtonSign(int x0,int y0,int x1,int y1,int *flag_choose_sign);
extern void ChooseButtonSign_Draw(int x0,int y0,int x1,int y1,int

```

```

    flag_choose_sign);
extern void ChooseButton_Light_Sign(int x0,int y0,int x1,int y1,int flag);
extern void ChooseButton_Darken_Sign(int x0,int y0,int x1,int y1,int flag,int flag_choose_sign);
extern void ChooseButton_Press_Sign(int x0,int y0,int x1,int y1,int flag);

#endif

```

35. surface.h

```

#ifndef _SURFACE_H_
#define _SURFACE_H_

extern void Button_Darken_Register(int flag);
extern void Button_Light_Register(int flag);
extern void Draw_Register();
extern void Darken_Register();
extern void Reg_Input_Password(char* pass_buffer);
extern void Reg_Input_Username(char* user_buffer);
extern void Reg_String_Display(int x,int y,int flag,char* p);
extern void Reg_Confirm_Password(char* confirm_buffer);
extern void Uni_Msgbox(char* s);
extern void Uni_Msgbox_1();
extern void Uni_Msgbox_2();
extern void Uni_Msgbox_3();
extern void Uni_Msgbox_4();
extern void Uni_Msgbox_5();
extern void Uni_Msgbox_6();

#endif

```

36. svga.h

```

#ifndef _svga_h
#define _svga_h

/*BMP 文件头结构*/
typedef struct tagBITMAPFILEHEADER{
    int bfType;
    long bfbSize; // 文件大小, 单位为字节
    int bfReserved1; // 保留, 必须为0
    int bfReserved2; // 保留, 必须为0
    long bfOffBits;

```

```

}BITMAPFILEHEADER;

/*BMP 信息头结构*/
typedef struct tagBITMAPINFOHEADER{
    long biSize; /*信息头大小*/
    long biWidth; /*图像宽度*/
    long biHeight; /*图像高度*/
    int biPlanes; /*必须为1*/
    int biBitCount; /*每像素位数, 必须为1, 4, 8, 24*/
    long biCompression; /* 压缩方法 */
    long biSizeImage; /* 实际图像大小, 必须是 4 的倍数 */
    long biXPelsPerMeter; /* 水平方向每米像素数 */
    long biYPelsPerMeter; /* 垂直方向每米像素数*/
    long biClrUsed; /* 所用颜色数*/
    long biClrImportant; /* 重要的颜色数 */
}BITMAPINFOHEADER;

typedef struct
{
    unsigned char B; /*蓝色分量, BLUE 缩写*/
    unsigned char G; /*绿色分量, GREEN 缩写*/
    unsigned char R; /*红色分量, RED 缩写*/
}WESHEN;

extern void SetSVGA64k();
extern void ReturnMode();
extern unsigned int GetSVGA();
extern unsigned int SelectPage(unsigned char page);
extern void Putpixel64k(int x, int y, int color);
extern unsigned int Getpixel64k(int x, int y);
extern void Horizline(int x, int y, int width, int color);
extern void Vertiline(int x, int y, int width, int color);
extern void Obliqline(int x0, int y0, int x1, int y1, int color);
extern int Putbmp64k(int x,int y,const char *path);
extern void get_image(int x0,int y0,int x1,int y1,unsigned int far *save);
extern void put_image(int x0,int y0,int x1,int y1,unsigned int far *save);
extern void save_image(int x0, int y0, int x1, int y1);
extern void printf_image(int x0, int y0, int x1, int y1);
extern void save_titlt(int x0, int y0, int x1, int y1);
extern void printf_title(int x0, int y0, int x1, int y1);
extern void save_image0(int x0, int y0, int x1, int y1);

```

```
extern void printf_image0(int x0, int y0, int x1, int y1);
```

```
#endif
```

37. universe.h

```
#ifndef _UNIVERSE_H_
```

```
#define _UNIVERSE_H_
```

```
extern void Button_Light_Login(int flag);
```

```
extern void Button_Darken_Login(int flag);
```

```
extern void Draw_Login();
```

```
extern void Darken_Login();
```

```
extern void Input_Username(char* user_buffer);
```

```
extern void Input_Password(char* pass_buffer);
```

```
extern void String_Display(int x,int y,int flag,char* p);
```

```
extern void Uni_Msgbox_7();
```

```
extern void Uni_Msgbox_8();
```

```
#endif
```

38. weather.h

```
#ifndef _WEATHER_H_
```

```
#define _WEATHER_H_
```

```
#define SUNNY 1
```

```
#define CLOUDY 2
```

```
#define RAINY 3
```

```
extern void Sky(int weather,int clock_hour);
```

```
extern void Cloud_Setup(int weather,int *cloud,int *speed,int *forward);
```

```
extern void Animate_Cloud(int weather,int *cloud,int speed,int forward);
```

```
extern void Draw_MapRainDrop(int x,int y);
```

```
extern void Random_RainDrop();
```

```
#endif
```

```

39. welcome.h
#ifndef _welcome_h
#define _welcome_h

extern void Welcome();

#endif

```

二、 源代码

```

1. about.c
#include "headfile.h"

/*****
*****
* @author          ytm
* @date            2022-4-16
*****
*****/

//driver
#define Revert_x0    5
#define Revert_y0    5

#define Intro_x0      12
#define Intro_y0      70
#define Text_xplus    160
#define Text_yplus    24

#define x_first_left   16
#define x_first_right  628
#define y_first        75
#define y_second       115
#define y_third        156
#define y_fourth       196
#define y_fifth        237
#define y_sixth        278
#define y_seventh      319
#define y_eighth       360
#define y_ninth        401
#define y_tenth        442
#define y_eleventh     483

```

```

#define y_twelfth 524
#define y_thirteenth 565

//follower
#define Revert_x1 (Revert_x0+39)
#define Revert_y1 (Revert_y0+39)

/*****
*****
* 函数名称      About
* 函数作用      菜单界面
* 函数输入      page 界面地址
* 函数输出      无
*****
*****/

void About(int *page)
{
    /*按钮状态变量*/
    int flag_revert,flag_mouse,flag_mouse_light;
    /*右键菜单变量*/
    int xm1,ym1,xm2,ym2;
Refresh_About:
    //SetSVGA64k();
    Solid_Bar(0,0,1023,767,BLACK);
    flag_revert=0,flag_mouse=0,flag_mouse_light=0;
    Draw_About();
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        /*刷新界面*/
        if(mouse_press(1,1,1023,767)==3)
        {
            mousehide(MouseX,MouseY);
            if(flag_mouse==0)
            {
                flag_mouse=1;
                /*根据鼠标位置绘制右键菜单*/
                if(MouseX>0&&MouseX<864&&MouseY>0&&MouseY<728)
                {
                    xm1=MouseX,ym1=MouseY,xm2=MouseX+160,ym2=Mous
eY+40;
                }
            }
        }
    }
}

```

```

else if(MouseX>863&&MouseX<1024&&MouseY>0&&MouseY
<728)
{
    xm1=MouseX-
160,ym1=MouseY,xm2=MouseX,ym2=MouseY+40;
}
else if(MouseX>0&&MouseX<864&&MouseY>727&&MouseY<
768)
{
    xm1=MouseX,ym1=MouseY-
40,xm2=MouseX+160,ym2=MouseY;
}
else
{
    xm1=MouseX-160,ym1=MouseY-
40,xm2=MouseX,ym2=MouseY;
}
save_image(xm1,ym1,xm2+3,ym2+3);
Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
}
resetMouse(MouseX,MouseY);
continue;
}
if(flag_mouse==1)
{
    if(mouse_out_press(xm1,ym1,xm2,ym2)==1)
    {
        mousehide(MouseX,MouseY);
        flag_mouse=0;
        printf_image(xm1,ym1,xm2+3,ym2+3);
        resetMouse(MouseX,MouseY);
    }
    if(mouse_press(xm1,ym1,xm2,ym2)==2)
    {
        shape=3;
        puthz(xm1+56,ym1+8,24,24,WHITE,"刷新");
        flag_mouse_light=1;
        continue;
    }
    else if(mouse_press(xm1,ym1,xm2,ym2)==1)
    {
        press=0;

```



```

        shape=0;
        //refresh page
        delay(100);
        goto Refresh_About;
    }
    if(flag_mouse_light!=0)
    {
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        flag_mouse_light=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
//button revert
if(MouseX>Revert_x0&&MouseX<Revert_x1&&MouseY>Revert_y0&&
MouseY<Revert_y1)
{
    if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Revert_y
1)==2)
    {
        shape=3;
        if(flag_revert==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_About(1);
            resetMouse(MouseX,MouseY);
            flag_revert=1;
        }
        continue;
    }
    else if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Rev
ert_y1)==1)
    {
        *page=0;
        //go to menu page
        delay(100);
        break;
    }
}
if(flag_revert!=0)
{

```

```

        mousehide(MouseX,MouseY);
        Button_Darken_About(flag_revert);
        if(flag_mouse==1)
        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        resetMouse(MouseX,MouseY);
        flag_revert=0;
    }
    //reset mouse
    if(shape!=0)
    {
        shape=0;
    }
}

/*****
*****

* 函数名称      Draw_About
* 函数作用      绘制关于界面
* 函数输入      无
* 函数输出      无
*****
*****/

void Draw_About()
{
    //upperBar: 1024*50 DARK_GRAY + LIGHT_GRAY
    UpperBar_Draw(50,4,LIGHT_GRAY,DARK_GRAY);
    //BoundaryBar: 612-615 DARK_GRAY
    Boundary_Draw(612,53,4,DARK_GRAY);
    //buttons 1: 40*40 BLACK + LIGHT_GRAY
    RevertImage_Draw(25,25,BLACK);
    puthz(480,9,32,32,BLACK,"关于");
    //message left
    puthz(x_first_left+48,y_first,24,24,WHITE,"欢迎使用定向越野模
拟软件!");
    puthz(x_first_left+48,y_second,24,24,WHITE,"本软件基于");
    putzm(x_first_left+168,y_second,24,14,WHITE,"BC3.1");
    puthz(x_first_left+244,y_second,24,24,WHITE,"搭建, 采用");
    putzm(x_first_left+361,y_second,24,16,WHITE,"SVGA");
    puthz(x_first_left+430,y_second,24,24,WHITE,"显示模式, 旨");

```

```

    puthz(x_first_left,y_third,24,24,WHITE,"在模拟定向越野陆地区域
比赛。");
    puthz(x_first_left+48,y_fourth,24,24,WHITE,"受限于性能，可能会
出现卡顿情况，敬请谅解！");
    puthz(x_first_left+48,y_fifth,24,24,WHITE,"本软件不用作任何商
业用途，版权归原作者所有，");
    puthz(x_first_left,y_sixth,24,24,WHITE,"请勿随意转载。");
    puthz(x_first_left+48,y_seventh,24,24,WHITE,"如果想要了解更多
信息，请查阅位于软件根目录下");
    putzm(x_first_left,y_eighth,24,13,WHITE,"readme.md");
    puthz(x_first_left+123,y_eighth,24,24,WHITE,"与");
    putzm(x_first_left+141,y_eighth,24,12,WHITE,"version.md");
    puthz(x_first_left+269,y_eighth,24,24,WHITE,"。");
    //date
    putzm(484,724,24,12,WHITE,"2022.3.23");
    //measage right
    puthz(x_first_right+128,y_first,32,32,WHITE,"开发人员");
    puthz(x_first_right,y_second+8,24,24,WHITE,"华中科技大学人工智
能与自动化学院");
    puthz(x_first_right,y_third+8,24,24,WHITE,"自动化");
    putzm(x_first_right+72,y_third+8,24,16,WHITE,"2109");
    puthz(x_first_right+136,y_third+8,24,24,WHITE,"班：叶庭茂、梁
忻健");
    puthz(x_first_right+128,y_fourth+16,32,32,WHITE,"指导教师");
    puthz(x_first_right,y_fifth+24,24,24,WHITE,"华中科技大学人工智
能与自动化学院");
    puthz(x_first_right,y_sixth+24,24,24,WHITE,"教授、副教授");
}

```

```

/*****
*****
* 函数名称      Button_Light_About
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void Button_Light_About(int flag)
{
    switch(flag)
    {
        case 1:
        {

```

```

        RevertButton_Draw(25,25,DARK_GRAY);
        RevertImage_Draw(25,25,WHITE);
        //revert button on
        break;
    }
}
}

/*****
*****
* 函数名称      Button_Darken_About
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Darken_About(int flag)
{
    switch(flag)
    {
        case 1:
        {
            RevertButton_Draw(25,25,LIGHT_GRAY);
            RevertImage_Draw(25,25,BLACK);
            //revert button off
            break;
        }
    }
}

```

2. advance.c

```

#include "headfile.h"

#define PI 3.1415926

/*****
*****
* @edictor      ytm
* @date          2022-4-4
* @discription   2022-4-4: 优化了SVGA 模式下的性能，完善了程序
注释
*****
*****/

```

```

/*****
*****

* 函数名称      Arc_up
* 函数作用      画圆心为(xr,yr), 半径为r 的180°弧线
* 函数输入      (x0,y0)圆心坐标, r 为半径长度, color 绘制颜色
* 函数注意      基于Pythagorean 算法
* 函数输出      无
*****
*****/

void Arc_up(int x0,int y0,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*Pythagorean 算法所需变量*/
    int x, y, d;
    y = r;
    d = 3 - r << 1;

    for (x = 0; x <= y; x++)
    {
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 + y) << 10) + x0;
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page - x) = color;
        *(video_buffer + page + x) = color;

        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 + x) << 10) + x0;
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page - y) = color;
        *(video_buffer + page + y) = color;
    }
}

```

```

        if (d < 0)
        {
            d += x * 4 + 6;
        }
        else
        {
            d += (x - y) * 4 + 10;
            y--;
        }
    }
}

/*****
*****
* 函数名称      Arc_down
* 函数作用      画圆心为(xr,yr), 半径为r 的180°弧线
* 函数输入      (x0,y0)圆心坐标, r 为半径长度, color 绘制颜色
* 函数注意      基于Pythagorean 算法
* 函数输出      无
*****
*****/
void Arc_down(int x0,int y0,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    //    unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*Pythagorean 算法所需变量*/
    int x, y, d;
    y = r;
    d = 3 - r << 1;

    for (x = 0; x <= y; x++)
    {
        /*计算显存地址偏移量 and 对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 - y) << 10) + x0;

```

```

new_page = page >> 15;
SelectPage(new_page);
*(video_buffer + page - x) = color;
*(video_buffer + page + x) = color;

/* 计算显存地址偏移量 and 对应的页面号, 做换页操作*/
page = ((unsigned long int)(y0 - x) << 10) + x0;
new_page = page >> 15;
SelectPage(new_page);
*(video_buffer + page - y) = color;
*(video_buffer + page + y) = color;

if (d < 0)
{
    d += x * 4 + 6;
}

else
{
    d += (x - y) * 4 + 10;
    y--;
}
}
}

/*****
*****
* 函数名称      Arc_right_up
* 函数作用      画圆心为(x,y), 半径为r 的90°弧线
* 函数输入      (x,y) 圆心坐标, r 为半径长度, color 绘制颜色
* 函数注意      基于Pythagorean 算法
* 函数输出      无
*****
*****/
void Arc_right_up(int x,int y,int r,int color)
{
    /* 显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /* 要切换的页面号*/
    unsigned char new_page = 0;

```

```

//    unsigned char old_page = 0;
/* 对应显存地址偏移量*/
unsigned long int page;

/*Pythagorean 算法所需变量*/
int tx=0,d;

while(tx<=r)
{
    d = sqrt(r*r-tx*tx);

    /* 计算显存地址偏移量和对应的页面号, 做换页操作*/
    page = ((unsigned long int)(y - tx) << 10) + (x + d);
    new_page = page >> 15;
    SelectPage(new_page);
    *(video_buffer + page) = color;

    tx++;
}
}

/*****
*****
* 函数名称      Arc_right_down
* 函数作用      画圆心为(x,y), 半径为r 的90°弧线
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数注意      基于Pythagorean 算法
* 函数输出      无
*****
*****/
void Arc_right_down(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    //    unsigned char old_page = 0;
    /* 对应显存地址偏移量*/
    unsigned long int page;

```



```

/*Pythagorean 算法所需变量*/
int tx=0,d;

while(tx<=r)
{
    d = sqrt(r*r-tx*tx);

    /*计算显存地址偏移量和对应的页面号, 做换页操作*/
    page = ((unsigned long int)(y + tx) << 10) + (x + d);
    new_page = page >> 15;
    SelectPage(new_page);
    *(video_buffer + page) = color;

    tx++;
}
}

/*****
*****
* 函数名称      Arc_left_up
* 函数作用      画圆心为(x,y), 半径为r 的90°弧线
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数注意      基于Pythagorean 算法
* 函数输出      无
*****
*****/
void Arc_left_up(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*Pythagorean 算法所需变量*/
    int tx=0,d;

    while(tx<=r)

```

```

    {
        d = sqrt(r*r-tx*tx);

        /* 计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y - tx) << 10) + (x - d);
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page) = color;

        tx++;
    }
}

/*****
*****

* 函数名称      Arc_Left_down
* 函数作用      画圆心为(x,y), 半径为r 的90°弧线
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数注意      基于Pythagorean 算法
* 函数输出      无
*****
*****/

void Arc_left_down(int x,int y,int r,int color)
{
    /* 显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /* 要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /* 对应显存地址偏移量*/
    unsigned long int page;

    /*Pythagorean 算法所需变量*/
    int tx=0,d;

    while(tx<=r)
    {
        d = sqrt(r*r-tx*tx);

        /* 计算显存地址偏移量和对应的页面号, 做换页操作*/

```

```

        page = ((unsigned long int)(y + tx) << 10) + (x - d);
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page) = color;

        tx++;
    }
}

/*****
*****
* 函数名称      Solid_QuarterSector_right_up
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y) 圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无
*****
*****/
void Solid_QuarterSector_right_up(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int tx = 0, ty = r, d = 3 - 2 * r, i;

    while( tx < ty)
    {
        // 画水平两点连线(<45 度)

        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y - tx) << 10);
        new_page = page >> 15;
        SelectPage(new_page);

        for (i = x; i <= x + ty; i++)

```

```

    {
        *(video_buffer + page + i) = color;
    }

    if (d < 0)                // 取上面的点
    {
        d += 4 * tx + 6;
    }
    else                      // 取下面的点
    {
        // 画水平两点连线(>45 度)

        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y - ty) << 10);
        new_page = page >> 15;
        SelectPage(new_page);

        for (i = x; i <= x + tx; i++)
        {
            *(video_buffer + page + i) = color;
        }
        d += 4 * (tx - ty) + 10, ty--;
    }
    tx++;
}
if (tx == ty)                // 画水平两点连线(=45 度)
{
    /*计算显存地址偏移量和对应的页面号, 做换页操作*/
    page = ((unsigned long int)(y - tx) << 10);
    new_page = page >> 15;
    SelectPage(new_page);

    for (i = x; i <= x + ty; i++)
    {
        *(video_buffer + page + i) = color;
    }
}
}

/*****
*****
* 函数名称      Solid_QuarterSector_Left_up
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形

```

```

* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无
*****
*****/
void Solid_QuarterSector_left_up(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    //    unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int tx=0,ty=r,i;
    double sx;

    while(tx<ty)
    {
        sx = sqrt(r*r-tx*tx);

        /*计算显存地址偏移量 and 对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y - tx) << 10);
        new_page = page >> 15;
        SelectPage(new_page);

        for(i=x-sx;i<=x;i++)
        {
            *(video_buffer + page + i) = color;
        }
        tx++;
    }
}

/*****
*****
* 函数名称      Solid_QuarterSector_right_down
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色

```

```

* 函数输出      无
*****
*****/
void Solid_QuarterSector_right_down(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    //    unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int tx=0,ty=r,i;
    double sx;

    while(tx<ty)
    {
        sx = sqrt(r*r-tx*tx);

        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y + tx) << 10);
        new_page = page >> 15;
        SelectPage(new_page);

        for(i=x;i<=x+sx;i++)
        {
            *(video_buffer + page + i) = color;
        }
        tx++;
    }
}

/*****
*****
* 函数名称      Solid_QuarterSector_Left_down
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无

```

```

*****
*****/
void Solid_QuarterSector_left_down(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int tx=0,ty=r,i;
    double sx;

    while(tx<ty)
    {
        sx = sqrt(r*r-tx*tx);

        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y + tx) << 10);
        new_page = page >> 15;
        SelectPage(new_page);

        for(i=x-sx;i<=x;i++)
        {
            *(video_buffer + page + i) = color;
        }
        tx++;
    }
}

/*****
*****
* 函数名称      Solid_QuarterSector_down
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无
*****

```

```

*****/
void Solid_QuarterSector_down(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int tx = 0, ty = r,i;
    float d = r/1.414;
    double sx;

    while(tx<ty)
    {
        if(tx<d)
        {
            /*计算显存地址偏移量和对应的页面号, 做换页操作*/
            page = ((unsigned long int)(y + tx) << 10);
            new_page = page >> 15;
            SelectPage(new_page);

            for(i=x-tx;i<=x+tx;i++)
            {
                *(video_buffer + page + i) = color;
            }
        }
        else
        {
            /*计算显存地址偏移量和对应的页面号, 做换页操作*/
            page = ((unsigned long int)(y + tx) << 10);
            new_page = page >> 15;
            SelectPage(new_page);

            for(i=x-sx;i<=x+sx;i++)
            {
                *(video_buffer + page + i) = color;
            }
        }
    }
}

```



```

        }
    }
    tx++;
    sx = sqrt(r*r-tx*tx);
}
}

/*****
*****
* 函数名称      Solid_QuarterSector_up
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y) 圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无
*****
*****/
void Solid_QuarterSector_up(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int tx = 0, ty = r,i;
    float d = r/1.414;
    double sx;

    while(tx<ty)
    {
        if(tx<d)
        {
            /*计算显存地址偏移量和对应的页面号, 做换页操作*/
            page = ((unsigned long int)(y - tx) << 10);
            new_page = page >> 15;
            SelectPage(new_page);

            for(i=x-tx;i<=x+tx;i++)

```

```

        {
            *(video_buffer + page + i) = color;
        }
    }
    else
    {
        /* 计算显存地址偏移量和对应的页面号，做换页操作*/
        page = ((unsigned long int)(y - tx) << 10);
        new_page = page >> 15;
        SelectPage(new_page);

        for(i=x-sx;i<=x+sx;i++)
        {
            *(video_buffer + page + i) = color;
        }
    }
    tx++;
    sx = sqrt(r*r-tx*tx);
}
}

/*****
*****
* 函数名称      Solid_QuarterSector_Left
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y)圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无
*****
*****/
void Solid_QuarterSector_left(int x,int y,int r,int color)
{
    /*显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/

```

```

int ty = 0, tx = r,i;
float d = r/1.414;
double sy;

while(ty<tx)
{
    if(ty<d)
    {
        for(i=y-ty;i<=y+ty;i++)
        {
            /* 计算显存地址偏移量 and 对应的页面号, 做换页操作*/
            page = ((unsigned long int)(i) << 10) + (x - ty);
            new_page = page >> 15;
            SelectPage(new_page);

            *(video_buffer + page) = color;
        }
    }
    else
    {
        for(i=y-sy;i<=y+sy;i++)
        {
            /* 计算显存地址偏移量 and 对应的页面号, 做换页操作*/
            page = ((unsigned long int)(i) << 10) + (x - ty);
            new_page = page >> 15;
            SelectPage(new_page);

            *(video_buffer + page) = color;
        }
    }
    ty++;
    sy = sqrt(r*r-ty*ty);
}

}

/*****
*****
* 函数名称      Solid_QuarterSector_right
* 函数作用      画圆心为(x,y), 半径为r 的90°实心扇形
* 函数输入      (x,y) 圆心坐标, r 为半径长度, color 绘制颜色
* 函数输出      无
*****
*****/

```

```

void Solid_QuarterSector_right(int x,int y,int r,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环所需变量*/
    int ty = 0, tx = r,i;
    float d = r/1.414;
    double sy;

    while(ty<tx)
    {
        if(ty<d)
        {
            for(i=y-ty;i<=y+ty;i++)
            {
                /*计算显存地址偏移量和对应的页面号, 做换页操作*/
                page = ((unsigned long int)(i) << 10) + (x + ty);
                new_page = page >> 15;
                SelectPage(new_page);

                *(video_buffer + page) = color;
            }
        }
        else
        {
            for(i=y-sy;i<=y+sy;i++)
            {
                /*计算显存地址偏移量和对应的页面号, 做换页操作*/
                page = ((unsigned long int)(i) << 10) + (x + ty);
                new_page = page >> 15;
                SelectPage(new_page);

                *(video_buffer + page) = color;
            }
        }
    }
}

```

```

        }
        ty++;
        sy = sqrt(r*r-ty*ty);
    }
}

3. animate.c
#include "headfile.h"

/*****
*****

* 函数名称      Map_Data_Save
* 函数作用      储存整幅地图图片缓存
* 函数输入      无
* 函数输出      无
*****
*****/

void Map_Data_Save()
{
    int i=0,j=0,k=0,part=0;
    char start[]={"\\.\\TEMP\\MAP"}; // 缓存地址前缀
    char part_add[3]={0}; // 缓存序号
    char end[]={"DAT"}; // 缓存地址后缀
    char add[30]={0}; // 生成的地址
    Map_Head_Save();
    // 地图分块成 8*7 份
    for(i=0;i<8;i++)
    {
        for(j=0;j<7;j++)
        {
            // 地址的生成
            itoa(part,part_add,10);
            strcpy(add,start);
            strcat(add,part_add);
            strcat(add,end);
            Map_Part_Save(i,j,add); // 保存地图块
            part++;
            for(k=0;k<30;k++)
            {
                *(add+k)=0; // 清空地址缓存
            }
        }
    }
}

```

```

}

/*****
*****
* 函数名称      Map_Part_Save
* 函数作用      储存分块地图图片缓存
* 函数输入      partx 分块行, party 分块列, address 文件保存路径
* 函数输出      无
*****
*****/

void Map_Part_Save(int partx,int party,char* address)
{
    FILE * fp;                      /*定义文件指针*/
    int x0=63+120*partx,y0=95+96*party;
    int i = 0;                      /*循环变量*/
    int j = 0;
    int wide = 120;//区块地图大小120*96
    int high = 96;
    int save = 0;

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
    xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    // char fg[30] = ".\\DATA\\MAP.DAT";
    fp = fopen(address, "wb+");      /*建立保存背景的文件*/
    if (fp == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<high; i++)
    {

```

```

        /* 计算显存地址偏移量和对应的页面号，做换页操作*/
        page = ((unsigned long int)(y0 + i) << 10) + x0;
        new_page = page >> 15;    /*32k 个点一换页，除以 32k 的替代算
法*/

        SelectPage(new_page);

        for (j = 0; j<wide; j++)
        {
            /*得到颜色*/
            save = video_buffer[page + j];

            fwrite(&save,sizeof(unsigned int),1,fp);
        }
    }

    fclose(fp);
}

/*****
*****
* 函数名称      Map_Head_Save
* 函数作用      储存地图顶部图片缓存
* 函数输入      无
* 函数输出      无
*****
*****/

void Map_Head_Save()
{
    FILE * fp;                /*定义文件指针*/
    int x0=63,y0=95-40;
    int i = 0;                /*循环变量*/
    int j = 0;
    int wide = 40*24; //顶部条状地图大小 960*24
    int high = 40;
    int save = 0;

    //    char fg[30] = ".\\DATA\\MAP.DAT";
    fp = fopen(".\\TEMP\\MAPH.DAT", "wb+");    /*建立保存背景的文件*/

```

```

    if (fp == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<high; i++)
    {
        for (j = 0; j<wide; j++)
        {

            save = Getpixel64k(x0+j,y0+i);
            fwrite(&save,sizeof(unsigned int),1,fp);

        }
    }

    fclose(fp);

}

/*****
*****
* 函数名称      Map_Part_Print
* 函数作用      地图区块输出
* 函数输入      parti 分块行, partj 分块列
* 函数输出      无
*****
*****/

void Map_Part_Print(int parti, int partj)
{
    FILE * fpo;                      /*定义文件指针*/
    int partx=parti/5;//将地图坐标转化为区块坐标
    int party=partj/4;
    int x0=63+120*partx;//计算区块的绝对坐标
    int y0=95+96*party;
    int i = 0;                        /*循环变量*/
    int j = 0;
    int wide =120;
    int high =96;
    int n=7*partx+party;//计算区块的序号

```



```

    unsigned    int save = 0;
    //生成区块地址
    char add[30]={"\\.\\TEMP\\MAP"};
    char part[3];
    char end[]={"DAT"};

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    itoa(n,part,10);
    strcat(add,part);
    strcat(add,end);

//    char fas[20]="saveim0.dat";
    fpo = fopen(add, "rb+");                                /*建立保存背景的文件*/
    /*
    if (fpo == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<high; i++)
    {
        /*计算显存地址偏移量 and 对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 + i) << 10) + x0;
        new_page = page >> 15;

        SelectPage(new_page);

        for (j = 0; j<wide; j++)
        {
            fread(&save,sizeof(unsigned int),1,fpo);

            /*写入颜色*/
            *(video_buffer + page + j) = save;

```

```

    }
}
fclose(fpo);
}

/*****
*****
* 函数名称      Map_Head_Print
* 函数作用      地图顶部图像输出
* 函数输入      无
* 函数输出      无
*****
*****/

void Map_Head_Print()
{
    FILE * fpo;                /*定义文件指针*/
    int x0=63;
    int y0=95-40;
    int i = 0;                  /*循环变量*/
    int j = 0;
    int wide =40*24;
    int high =40;
    unsigned int save = 0;

    // char fas[20]="saveim0.dat";
    fpo = fopen(".\\TEMP\\MAPH.DAT", "rb+"); /*
建立保存背景的文件*/
    if (fpo == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<high; i++)
    {
        for (j = 0; j<wide; j++)
        {
            fread(&save,sizeof(unsigned int),1,fpo);
            Putpixel64k(x0 + j, y0 + i, save);
        }
    }
}

```

```

        fclose(fpo);
    }

    /*****
    *****/

    * 函数名称      Ani_Test
    * 函数作用      动画调试函数
    * 函数输入      无
    * 函数输出      无
    *****/

    *****/

void Ani_Test()
{
    int px=3,py=0;
    int x=63+24*px;
    int y=95+24*py;
    int move=WALK;
    int color=RED;
    while(1)
    {
        Teen_Movement(x,y,MOVE_LEFT,move,1,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,2,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,3,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,2,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,1,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,2,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,3,color);
        delay(500);
        Map_Refresh(px,py);
        Teen_Movement(x,y,MOVE_LEFT,move,2,color);
    }
}

```

```

        delay(500);
        Map_Refresh(px,py);
    }
}

/*****
*****
* 函数名称      Map_Refresh
* 函数作用      地图区块刷新
* 函数输入      i, j 为当前所需刷新地图坐标
* 函数输出      无
*****
*****/

void Map_Refresh(int i,int j)
{
    Map_Part_Print(i,j);//刷新人物当前所在区块
    //left
    if(i%5==0&&i!=0)
    {
        Map_Part_Print(i-1,j);//若人物位于区块左边缘,刷新左边区块
    }
    //right
    else if(i%5==4&&i!=39)
    {
        Map_Part_Print(i+1,j);//若人物位于区块右边缘,刷新右边区块
    }
    //up
    if((j%4==0||j%4==1)&&j!=0)
    {
        Map_Part_Print(i,j-1);//若人物位于区块上边缘,刷新上边区块
    }
    //down
    else if(j%4==3&&j!=27)
    {
        Map_Part_Print(i,j+1);//若人物位于区块下边缘,刷新下边区块
    }
}

/*****
*****
* 函数名称      Animate
* 函数作用      动画模拟

```

```

* 函数输入      direction 人物运动方向, people 人数, age 年龄,
clock_hour 时钟时, clock_minute 时钟分, person_basic 人物基础信息
*
      person_competition 人物竞赛信息, point 点位
数, point_x[] 点位x 坐标, point_y[] 点位y 坐标, your_number 玩家序号
* 函数输出      无
*****
*****/

```

```

void Animate(int **direction,int people,int age,int *clock_hour,i
nt *clock_minute,people_basic *person_basic,people_competition *p
erson_competition,int point,int point_x[],int point_y[],int your_
number,int weather)
{
    int i=0,x=63,y=95,t=0,p=0;//i 为人数扫描变量, xy 是地图左上角坐
标, p 是点位扫描坐标
    int pi=0,pj=0;
    int forward;//人物运动方向缓存
    int move[6];//人物动作缓存
    int lastmove[6];//人物上一动作缓存
    const int NOT_CLIMB_LR[5]={1,2,3,2,0};//各运动动作的动画帧顺序
    const int NOT_CLIMB_UD[9]={1,2,3,2,1,4,5,4,0};
    const int CLIMB_LR[3]={1,2,0};
    const int CLIMB_UD[5]={1,2,3,2,0};
    int *order;//指向动画帧顺序的指针
    int *change[6];//记录下一指针原始数据的指针
    int *frame[6]; //指向动画帧顺序的指针
    int k[6];//动画帧扫描变量
    int finish[6]={0};//完成比赛变量
    int finish_num=0;//完成比赛人数
    int finish_point[6]={0};//人物完成点位的数量
    int time_hour[6]={0};//人物出发的时、分
    int time_minute[6]={0};
    int color[6]={0};//人物衣着颜色
    int map[6]={0};//人物比赛路程
    int distance[6]={0};//人物当前路程
    int start[6]={0};//人物出发状态
    int location_i[6]={0};//人物当前位置
    int location_j[6]={0};
    int last_location_i[6]={0};//人物上一位置
    int last_location_j[6]={0};
    double plus[6]={0};//人物当前时序前进路程
    int advance;//人物前进格数缓存
    int j[6]={0};//direction 数组扫描变量

```

```

int cloud[10];
int Cloud_Speed=0;
int Cloud_Forward=0;
Cloud_Setup(weather,cloud,&Cloud_Speed,&Cloud_Forward);
Color_Random(person_basic,people);//给人物的衣服颜色随机
for(i=0;i<people;i++)
{
    frame[i]=NOT_CLIMB_LR;
    change[i]=frame[i];
} //初始化关键帧顺序
while(finish_num<people)
{
    //时序, 每次循环时钟进行1 分钟
    if(*clock_minute==59)
    {
        SystemClock_Plus(clock_hour,clock_minute,1,0);
    }
    SystemClock_Plus(clock_hour,clock_minute,0,1);
    Solid_Bar(929,13,1023,13+24,LIGHT_GRAY);
    Clock_Display(929,13,(*clock_hour),(*clock_minute),BLACK)
;

    //人物的状态初始化, 以及时间的计算
    for(i=0;i<people;i++)
    {
        color[i]=person_basic[i].color;
        //consume[i]=person_basic[i].ps_consume;
        Time_Clac(time_hour+i,time_minute+i,person_competition[i],*clock_hour,*clock_minute);
    }
    //人物运动信息的显示
    Distance_Clac(map,direction,j,people);
    Info(finish_point,point,time_hour,time_minute,map,age,color,people);
    Info_Player(your_number);
    //start scan
    for(i=0;i<people;i++)
    {
        if(Clock_Compare(*clock_hour,*clock_minute,person_competition[i].hour_point[0],person_competition[i].minute_point[0])!=0)
        {
            start[i]=1;
            finish_point[i]=1;//当人物出发时间与当前时钟一致, 设

```

定出发状态

```
    }  
    }  
    //display  
    for(i=0;i<people;i++)  
    {  
        if(start[i]==1&&finish[i]==0&&location_i[i]<=39&&location_j[i]<=27)  
        {  
            Map_Refresh(last_location_i[i],last_location_j[i]  
);// 当人物运动时，刷新当前区块地图  
        }  
    }  
    Map_Head_Print();//刷新顶部地图
```

Animate_Cloud(weather,cloud,Cloud_Speed,Cloud_Forward);//
云朵动画

```
// 人物运动主循环  
for(i=0;i<people;i++)  
{  
    if(location_i[i]>=39&&location_j[i]>=27)  
    {  
        if(finish[i]==0)  
        {  
            finish_num++;  
            finish_point[i]++;  
            finish[i]=1;  
        }    // 当人物运动到终点，设置完成比赛状态  
        continue;  
    }  
    if(direction[i][j[i]]==0)  
    {  
        if(finish[i]==0)  
        {  
            person_competition[i].hour_point[point-  
1]=*clock_hour;  
            person_competition[i].minute_point[point-  
1]=*clock_minute;  
            finish_num++;  
            finish_point[i]++;  
            finish[i]=1;  
        }    // 当人物不再运动，也设置完成比赛状态  
        continue;  
    }  
}
```

动画

```
}
if(finish_point[i]==point)
{
    if(finish[i]==0)
    {
        finish_num++;
        finish_point[i]++;
        finish[i]=1;
    }    // 当人物运动到终点, 设置完成比赛状态
    continue;
}
if(start[i]==1&&finish[i]==0)
{

    if(direction[i][j[i]]!=0)
    {
        // 记录当前瞬间下, 人物的运动信息
        x=63+24*location_i[i];
        y=95+24*location_j[i]-24;
        move[i]=direction[i][j[i]]/10;
        forward=direction[i][j[i]]%10;
        // 将动画关键帧指针指向对应的顺序数组, 以播放对应

        if(move[i]==CLIMB)
        {
            if(forward==MOVE_UP||forward==MOVE_DOWN)
            {
                frame[i]=CLIMB_UD;
            }
            else
            {
                frame[i]=CLIMB_LR;
            }
        }
        else
        {
            if(forward==MOVE_UP||forward==MOVE_DOWN)
            {
                frame[i]=NOT_CLIMB_UD;
            }
            else
            {
                frame[i]=NOT_CLIMB_LR;
            }
        }
    }
}
```



```

    }
}
// 若人物的运动动作发生改变, 重新初始化关键帧指针
if(change[i]!=frame[i])
{
    k[i]=0;
}
change[i]=frame[i];
if(lastmove[i]!=move[i])
{
    k[i]=0;
}
lastmove[i]=move[i];
// 每次时序滚动到下一个关键帧, 若滚动完毕, 从头开
始

k[i]++;
if(*(frame[i]+k[i])==0)
{
    k[i]=0;
}
// 扫描年龄, 绘制出对应的关键帧
switch(age)
{
    case CHILD:
        Child_Movement(x,y,forward,move[i],*(frame[i]+k[i]),person_basic[i].color);
        break;
    case TEEN:
        Teen_Movement(x,y,forward,move[i],*(frame[i]+k[i]),person_basic[i].color);
        break;
    case MID:
        Mid_Movement(x,y,forward,move[i],*(frame[i]+k[i]),person_basic[i].color);
        break;
    case OLD:
        Old_Movement(x,y,forward,move[i],*(frame[i]+k[i]),person_basic[i].color);
        break;
}
// 记录上一次运动的信息
last_location_i[i]=location_i[i];
last_location_j[i]=location_j[i];

```

```

//move
// 读取对应的运动动作，根据当前速度，计算当前时序
前进的路程

switch(move[i])
{
    case WALK:
        plus[i]+=24*person_basic[i].verb_walk
;
        break;
    case RUN:
        plus[i]+=24*person_basic[i].verb_run;
        break;
    case CLIMB:
        plus[i]+=24*person_basic[i].verb_clim
b;
        break;
}
//advance
// 计算当前时序前进
while(Scan_Advance(direction[i][j[i]]%10,plus
[i]))
{
    forward=direction[i][j[i]]%10;
    switch(forward)
    {
        case 1:
            location_j[i]-=1;
            plus[i]-=24;
            break;
        case 2:
            location_j[i]+=1;
            plus[i]-=24;
            break;
        case 3:
            location_i[i]-=1;
            plus[i]-=24;
            break;
        case 4:
            location_i[i]+=1;
            plus[i]-=24;
            break;
        case 5:// 扫描到斜着走时，重新确定运动

```

```

        location_i[i]-=1;
        location_j[i]-=1;
        plus[i]-=34;
        break;
    case 6:
        location_i[i]-=1;
        location_j[i]+=1;
        plus[i]-
=34;

        break;
    case 7:
        location_i[i]+=1;
        location_j[i]-=1;
        plus[i]-
=34;

        break;
    case 8:
        location_i[i]+=1;
        location_j[i]+=1;
        plus[i]-=34;
        break;
    }
    //point scan
    if(finish[i]==0)
    {
        for(p=1;p<point;p++)
        {
            //人物到达每个点位, 记录对应时间
            if(location_i[i]==point_x[p]&&location_j[i]==point_y[p]&&finish_point[i]==p)
            {
                person_competition[i].hour_point[p]=*clock_hour;
                person_competition[i].minute_point[p]=*clock_minute;
                finish_point[i]++;
            }
        }
        j[i]+=1;
    }
}
}
}

```

```

    }
    delay(500);          // 动画播放速率
}
}

/*****
*****
* 函数名称      Animate_Help
* 函数作用      帮助界面动画
* 函数输入      x, y 为播放位置, forward 运动方向, move 动作, age 年
                  龄, color 衣服颜色
* 函数输出      无
*****
*****/

```

```

void Animate_Help(int x,int y,char forward,char move,int age,int
color)
{
    // 动画播放顺序数组
    const int NOT_CLIMB_LR[9]={1,2,3,2,1,2,3,2,0};
    const int NOT_CLIMB_UD[9]={1,2,3,2,1,4,5,4,0};
    const int CLIMB_LR[5]={1,2,1,2,0};
    const int CLIMB_UD[9]={1,2,3,2,1,2,3,2,0};
    int *order;// 动画关键帧指针
    int frame;// 关键帧扫描变量
    x+=20;
    y+=8;
    // 读取动作, 确定指向的关键帧顺序数组
    if(move==CLIMB)
    {
        if(forward==MOVE_UP || forward==MOVE_DOWN)
        {
            order=CLIMB_UD;
        }
        else
        {
            order=CLIMB_LR;
        }
    }
    else
    {
        if(forward==MOVE_UP || forward==MOVE_DOWN)
        {

```

```

        order=NOT_CLIMB_UD;
    }
    else
    {
        order=NOT_CLIMB_LR;
    }
}
// 读取年龄，从关键帧库播放对应动画
switch(age)
{
    case CHILD:
        for(frame=0;order[frame]!=0;frame++)
        {
            Child_Movement(x,y,forward,move,order[frame],color);

            delay(250);
            Solid_Bar(x-20,y-8,x+43,y+55,BLACK);
        }
        break;
    case TEEN:
        for(frame=0;order[frame]!=0;frame++)
        {
            Teen_Movement(x,y,forward,move,order[frame],color);

            delay(250);
            Solid_Bar(x-20,y-8,x+43,y+55,BLACK);
        }
        break;
    case MID:
        for(frame=0;order[frame]!=0;frame++)
        {
            Mid_Movement(x,y,forward,move,order[frame],color);

            delay(250);
            Solid_Bar(x-20,y-8,x+43,y+55,BLACK);
        }
        break;
    case OLD:
        for(frame=0;order[frame]!=0;frame++)
        {
            Old_Movement(x,y,forward,move,order[frame],color);

            delay(250);

```

```

        Solid_Bar(x-20,y-8,x+43,y+55,BLACK);
    }
    break;
}
}

/*****
*****
* 函数名称      Color_Random
* 函数作用      人物衣服颜色随机初始化
* 函数输入      person_basic 人物基础信息, people 人数
* 函数输出      无
*****
*****/

void Color_Random(people_basic *person_basic,int people)
{
    const int color[6]={RED,SKY_BLUE,YELLOW,ORANGE,DARK_GREEN,HOT
_PINK};//人物衣服颜色库
    int point[6]={1,2,3,4,5,6};//记录color 数组顺序的数组
    int i=0;//次数变量
    int d1,d2;//随机变量
    int temp;//缓存
    for(i=0;i<10;i++)
    {
        d1=random(6);
        d2=random(6);
        //随机交换顺序数组10次，使颜色打乱
        temp=point[d1];
        point[d1]=point[d2];
        point[d2]=temp;
    }
    for(i=0;i<people;i++)
    {
        person_basic[i].color=color[point[i]-1];
        //将随机结果记录到人物信息
    }
}

int Scan_Advance(int direction,int plus)
{
    if(direction<=4)
    {

```

```

        if(plus>=24)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
    else if(direction<=8)
    {
        if(plus>=34)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
}

```

4. asc.c

```
#include"headfile.h"
```

```

/*****
*****
* @author          ytm
* @date            2022-4-4
* @notice          此版本为SVGA 64k(24 位)显示版本(专门为SVGA 优
化)
*                  若ASC 库的文件不存在，将在按键后等待6s 后退出
*                  32*32 的显示采用放大技术显示，显示效果有所减损
*                  putsz 函数中数字默认取绝对值显示
*                  putsz_double 函数中数字默认取绝对值显示，默认
最多取三位小数
* @discription     2022/3/15 putsz 函数加入了对于多位数字显示的支
持，修复了0 不显示的问题
*                  2022/4/3 新增了putsz_double 的函数，加入了
对double 型数字输出的支持，最多支持6 位整数
*****
*****/

/*****

```

```

*****
* 函数名称          putzm
* 函数作用          输出字母
* 函数输入          x 输出位置横坐标, y 输出位置纵坐标, flag 输出大小,
part 输出偏移, color 输出颜色, s 输出内容
* 函数输出          无
*****
*****/
void putzm(int x, int y,int flag,int part,int color,char *s)
{
    FILE *asc_p=NULL;                                //定
    义ASC 库文件指针
    unsigned long offset;                            //定义字母在字
    库中的偏移量
    unsigned char mask[] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x
    01}; // 功能数组, 用于显示字母点阵中的亮点
    int i,j,pos;
    unsigned int far * const video_buffer = (unsigned int far *
    )0xa0000000L; // 显存指针常量, 指向显存首地址, 指针本身不允许修改
    unsigned char new_page = 0;                      //要切换的页面号
    unsigned long int page;                          //对应显存地址偏移量

    switch(flag) //不同的flag 对应不同的ASC 库, 实现了字母的大小
    可根据需要改变
    {
        case 16 :
            {
                char mat[32]; //16*16 的字母需要 32 个字节的数组来
                存储

                int y0=y;
                int x0=x;
                asc_p = fopen(".\\ASC\\ASC16.DZK","rb"); //
                路径自行修改

                if(asc_p==NULL)
                {
                    getch();
                    delay(6000);
                    exit(1);
                }
                while(*s!=NULL)
                {
                    while (x<1024-flag && (*s!=NULL))
                    {

```



```

        y=y0;
        offset=((int)(s[0]))*32L;
        fseek(asc_p,offset,SEEK_SET);          //
重定位文件指针
        fread(mat,32,1,asc_p);                // 读出
该字母的具体点阵数据,1 为要读入的项数
        for(i=0;i<16;i++)
        {
            /* 计算显存地址偏移量和对应的页面号, 做换
页操作*/
            page = ((unsigned long int)y << 10) +
x;
            new_page = page >> 15;
            SelectPage(new_page);
            pos=2*i;                          //16*16 矩阵中有
每一行有两外字节
            for(j=0;j<16;j++)                // 一行一行地扫
描, 将位上为1 的点显示出来
            {
                if((mask[j%8]&mat[pos+j/8])!=NULL
) //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环
                {
                    *(video_buffer + page + j) =
color;
                }
            }
            y++;
        }
        /*****
*****
        以上一个字母显示完
        *****/
        *****/
        x+=part;                            //给 x 一个偏移量 part
        s+=1;                                //字母的 AscII 码占 1 个字节
    }
    x=x0;y0+=flag+10; //一行字母显示完后, 重新从左侧
开始输出字母, 给 y 一个偏移量, 10 为行间距
    }
    break;
}
case 24 :
{

```

```

char mat[48];    //16*24 的字母需要 48 个字节的数组来
存储

int y0=y;
int x0=x;
int diviate;    //实现 16*24 转换为 part*24 模式输出
asc_p = fopen(".\\ASC\\ASC24", "rb");    //路径

自行修改

if(asc_p==NULL)
{
    getch();
    delay(6000);
    exit(1);
}
diviate=(24-part)/2;
while(*s!=NULL)
{
    while (x<1024-flag && (*s!=NULL))
    {
        y=y0;
        offset=((int)(s[0]))*48L;
        fseek(asc_p,offset,SEEK_SET);    //

        重定位文件指针

        fread(mat,48,1,asc_p);    // 读出
        该字母的具体点阵数据,1 为要读入的项数

        for(i=0;i<24;i++)
        {
            /* 计算显存地址偏移量和对应的页面号, 做换
            页操作*/

            page = ((unsigned long int)y << 10) +
            (x + diviate);    //利用偏移使 24*24 调整为 24*part 的格式输出
            new_page = page >> 15;
            SelectPage(new_page);
            pos=2*i;    //16*24 矩阵中有

            每一行有两外字节

            for(j=0;j<16;j++)    //一行一行地扫
            描, 将位上为 1 的点显示出来

            {
                if((mask[j%8]&mat[pos+j/8])!=NULL
            )    //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环
                {
                    *(video_buffer + page + j) =
            color;

                }
            }
        }
    }
}

```

```

        }
        y++;
    }
    /*****
*****
    以上一个字母显示完
    *****/
*****/

        x+=part;          // 给 x 一个偏移量 part
        s+=1;             // 字母的 AscLL 码占 1 个字节
    }
    x=x0;y0+=flag+10; // 一行字母显示完后, 重新从左侧
开始输出字母, 给 y 一个偏移量, 10 为行间距
    }
    break;
}
case 32 :
{
    char mat[32]; //16*16 的字母需要 32 个字节的数组来
存储

    int y0=y;
    int x0=x;
    asc_p = fopen(".\\ASC\\ASC16.DZK", "rb"); //
路径自行修改

    if(asc_p==NULL)
    {
        getch();
        delay(6000);
        exit(1);
    }
    while(*s!=NULL)
    {
        while (x<1024-flag && (*s!=NULL))
        {
            y=y0;
            offset=((int)(s[0]))*32L;
            fseek(asc_p,offset,SEEK_SET); //
重定位文件指针

            fread(mat,32,1,asc_p); // 读出
该字母的具体点阵数据, 1 为要读入的项数
            for(i=0;i<16;i++)
            {
                pos=2*i; //16*16 矩阵中有

```

每一行有两外字节

```

                                for(j=0;j<16;j++)    //一行一行地扫描，将位上为1的点显示出来
                                {
                                    if((mask[j%8]&mat[pos+j/8])!=NULL
)    //j%8 只能在0-8之间循环，j/8在0,1之间循环
                                    {
                                        Solid_Bar(x+2*j,y,x+2*j+1,y+1,color);    //x轴与y轴2倍显示放大
                                    }
                                    y=y+2;    //y轴位置移动两倍放大
                                }
                                /*****
*****
                                以上一个字母显示完
                                *****/
*****/
                                x+=part;    //给x一个偏移量part
                                s+=1;    //字母的AscLL码占1个字节
                                }
                                x=x0;y0+=flag+10; //一行字母显示完后,重新从左侧开始输出字母,给y一个偏移量,10为行间距
                                }
                                break;
                            }
                        default:
                        {
                            break;
                        }
                    }
                    fclose(asc_p);
                }

/*****
*****
* 函数名称      putsz
* 函数作用      输出数字
* 函数输入      x 输出位置横坐标, y 输出位置纵坐标, flag 输出大小,
part 输出偏移, color 输出颜色, sz 输出数字
* 函数输出      无
*****
*****/
```

```

void putsz(int x, int y,int flag,int part,int color,int sz)
{
    FILE *asc_p=NULL;                                //定义ASC 库文件指针
    unsigned long offset;                             //定义数字在字库中的偏移量
    unsigned char mask[] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01}; //功能数组, 用于显示数字点阵中的亮点
    int i,j=sz,pos,temp[5],weishu=0;
    unsigned int far * const video_buffer = (unsigned int far *)0xa0000000L; //显存指针常量, 指向显存首地址, 指针本身不允许修改
    unsigned char new_page = 0;                       //要切换的页面号
    unsigned long int page;                            //对应显存地址偏移量

    if(j==0)
    {
        weishu=1;
        temp[4]=0;
    }
    else
    {
        if(j<0)
        {
            j=-j;                                     //默认取绝对值
        }
        while(j!=0)
        {
            temp[4-weishu]=j%10;                      //temp 数组储存要输出的数字
            j=(j-temp[4-weishu])/10;
            weishu++;                                  //weishu 储存数字的位数
        }
    }

    switch(flag) //不同的flag 对应不同的ASC 库, 实现了数字的大小可根据需要改变
    {
        case 16 :
        {
            char mat[32]; //16*16 的数字需要 32 个字节的数组来存储

            int y0=y;
            int x0=x;
            int s=5-weishu; //s 储存了取出数字在数组中初始的

```

位置

路径自行修改

48 是 0 的 ASC 码值

重定位文件指针

该数字的具体点阵数据,1 为要读入的项数

页操作*/

每一行有两外字节

描, 将位上为 1 的点显示出来

) //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环

color;

```
asc_p = fopen(".\\ASC\\ASC16.DZK", "rb");    //
if(asc_p==NULL)
{
    getch();
    delay(6000);
    exit(1);
}
while(weishu>0)
{
    while (x<1024-flag&&(weishu>0))
    {
        y=y0;
        offset=(temp[s]+48)*32L;                //
        fseek(asc_p,offset,SEEK_SET);           //
        fread(mat,32,1,asc_p);                  // 读出
        for(i=0;i<16;i++)
        {
            /* 计算显存地址偏移量 and 对应的页面号, 做换
            页操作*/
            page = ((unsigned long int)y << 10) +
            x;
            new_page = page >> 15;
            SelectPage(new_page);
            pos=2*i;                               //16*16 矩阵中有
            for(j=0;j<16;j++)                      //一行一行地扫
            {
                if((mask[j%8]&mat[pos+j/8])!=NULL
                ) //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环
                {
                    *(video_buffer + page + j) =
                    color;
                }
            }
            y++;
        }
    }
}
/*****
```

以上一个数字显示完

*****/

x+=part; //给x一个偏移量part
s+=1; //取用数组的指针向下移动一位

位

weishu--; //仍需输出位数减小一位

}

x=x0;y0+=flag+10; //一行数字显示完后,重新从左侧
开始输出数字,给y一个偏移量,10为行间距

}

break;

}

case 24 :

{

char mat[48]; //16*24的数字需要48个字节的数组来
存储

int y0=y;

int x0=x;

int s=5-weishu; //s储存了取出数字在数组中初始的
位置

int diviate; //数模的偏移

asc_p = fopen(".\\ASC\\ASC24","rb"); //路径
自行修改

if(asc_p==NULL)

{

getch();

delay(6000);

exit(1);

}

while(weishu>0)

{

while (x<1024-flag&&(weishu>0))

{

y=y0;

offset=(temp[s]+48)*48L;

fseek(asc_p,offset,SEEK_SET); //重定位文件指针

fread(mat,48,1,asc_p); //读出

该字母的具体点阵数据,1为要读入的项数

diviate=(24-part)/2;

for(i=0;i<24;i++)

```

{
    /* 计算显存地址偏移量和对应的页面号，做换
页操作*/
    page = ((unsigned long int)y << 10) +
(x + diviate); // 调整为part*24 的格式输出
    new_page = page >> 15;
    SelectPage(new_page);
    pos=2*i; //16*24 矩阵中有
每一行有两外字节
    for(j=0;j<16;j++) // 一行一行地扫
描，将位上为1 的点显示出来
    {
        if((mask[j%8]&mat[pos+j/8])!=NULL
) //j%8 只能在0-8 之间循环，j/8 在0，1 之间循环
        {
            *(video_buffer + page + j) =
color;
        }
    }
    y++;
}
/*****
*****
以上一个数字显示完
*****
******/
x+=part; // 给x 一个偏移量part
s+=1; // 取用数组的指针向下移动一
位
    weishu--; // 仍需输出位数减小一位
}
x=x0;y0+=flag+10; // 一行数字显示完后，重新从左侧
开始输出数字，给y 一个偏移量，10 为行间距
}
break;
}
case 32 :
{
    char mat[32]; //16*16 的数字需要32 个字节的数组来
存储
    int y0=y;
    int x0=x;
    int s=5-weishu; //s 储存了取出数字在数组中初始的

```


位置

```
asc_p = fopen(".\\ASC\\ASC16.DZK","rb");    //
```

路径自行修改

```
if(asc_p==NULL)
```

```
{
```

```
    getch();
```

```
    delay(6000);
```

```
    exit(1);
```

```
}
```

```
while(weishu>0)
```

```
{
```

```
    while (x<1024-flag&&(weishu>0))
```

```
    {
```

```
        y=y0;
```

```
        offset=(temp[s]+48)*32L;
```

```
        fseek(asc_p,offset,SEEK_SET);    //
```

重定位文件指针

```
        fread(mat,32,1,asc_p);    // 读出
```

该数字的具体点阵数据,1 为要读入的项数

```
        for(i=0;i<16;i++)
```

```
        {
```

```
            pos=2*i;    //16*16 矩阵中有
```

每一行有两外字节

```
            for(j=0;j<16;j++)    // 一行一行地扫
```

描, 将位上为1 的点显示出来

```
            {
```

```
                if((mask[j%8]&mat[pos+j/8])!=NULL
```

) //j%8 只能在0-8 之间循环, j/8 在0, 1 之间循环

```
                {
```

```
                    Solid_Bar(x+2*j,y,x+2*j+1,y+1
```

,color); //x 轴与y 轴2 倍显示放大

```
                }
```

```
            }
```

```
            y=y+2;    //y 轴位置移动两倍放大
```

```
        }
```

```
    /*****
```

以上一个数字显示完

```
    *****/
```

*****/

```
    x+=part;
```

// 给x 一个偏移量part

```
    s+=1;
```

// 取用数组的指针向下移动一

位

```

        weishu--; // 仍需输出位数减小一位
    }
    x=x0;y0+=flag+10; // 一行数字显示完后, 重新从左侧
    开始输出数字, 给 y 一个偏移量, 10 为行间距
}
break;
}
default:
{
    break;
}
}
fclose(asc_p);
}

```

```

/*****
*****

```

```

* 函数名称      putzm
* 函数作用      输出浮点数字
* 函数输入      x 输出位置横坐标, y 输出位置纵坐标, flag 输出大小,
part 输出偏移, color 输出颜色, sz 输出数字
* 函数输出      无

```

```

*****
*****/

```

```

void putsz_double(int x, int y,int flag,int part,int color,double
sz)
{
    int i=0,j,weishu_zs=0,temp;

    // 默认取绝对值
    if(sz<0)
    {
        sz=-sz;
    }

    // 输出整数和获取位数
    temp=(floor)(sz);
    putsz(x,y,flag,part,color,temp);
    if(temp==0)
    {
        weishu_zs==1;
    }
    else

```

```

    {
        while(temp!=0)
        {
            temp=(floor)(temp/10);
            weishu_zs++;
        }
    }

    // 输出小数点
    putzm(x+weishu_zs*part,y,flag,part,color,".");

    // 输出小数
    temp=(floor)((sz-(floor)(sz))*1000+0.5);
    putsz(x+(weishu_zs+1)*part,y,flag,part,color,temp);
}

```

5. back.c

```

#include "headfile.h"

#define BODYCOLOR_B OCHER
#define BODYCOLOR_C MAROON

/*****
*****

* 函数名称      Back_People_A
* 函数作用      绘制主界面A 人
* 函数输入      x,y 坐标, zoom 放大倍数, color 衣服颜色
* 函数输出      无
*****
*****/

void Back_People_A(int x,int y,int zoom,int color)
{
    zoom=5;
    //Putpixel64k(x,y,WHITE);
    Solid_Circle(x+11*zoom,y+11*zoom,5*zoom,BODYCOLOR); //head
    Scarf(x+11*zoom,y+11*zoom,5*zoom,20,15,RED);
    Solid_Bar(x+7*zoom,y+18*zoom,x+16*zoom,y+31*zoom,color); //bod
y
    Solid_Quadrangle(x+18*zoom,y+19*zoom,x+19*zoom,y+18*zoom,x+23
*zoom,y+24*zoom,x+24*zoom,y+23*zoom,color); //R
    Solid_Quadrangle(x+4*zoom,y+18*zoom,x+5*zoom,y+19*zoom,x+0*zo
om,y+26*zoom,x+1*zoom,y+27*zoom,color);
    Solid_Quadrangle(x+1*zoom,y+29*zoom,x+0*zoom,y+30*zoom,x+4*zo

```

```

om,y+34*zoom,x+3*zoom,y+35*zoom,BODYCOLOR);//L
    Solid_Quadrangle(x+14*zoom,y+35*zoom,x+16*zoom,y+33*zoom,x+19
*zoom,y+40*zoom,x+21*zoom,y+38*zoom,color);
    Solid_Quadrangle(x+21*zoom,y+40*zoom,x+24*zoom,y+41*zoom,x+19
*zoom,y+46*zoom,x+22*zoom,y+47*zoom,BODYCOLOR);//R
    Solid_Quadrangle(x+7*zoom,y+33*zoom,x+10*zoom,y+34*zoom,x+4*z
oom,y+40*zoom,x+8*zoom,y+41*zoom,color);//L
    Solid_Quadrangle(x+6*zoom,y+43*zoom,x+3*zoom,y+42*zoom,x-
1*zoom,y+46*zoom,x+2*zoom,y+48*zoom,BODYCOLOR);
    Solid_Quadrangle(x+125,y+121,x+122,y+126,x+151,y+141,x+154,y+
134,BODYCOLOR);
}

```

```

/*****
*****
* 函数名称      Back_People_B
* 函数作用      绘制主界面B人
* 函数输入      x,y 坐标, zoom 放大倍数, color 衣服颜色
* 函数输出      无
*****
*****/

```

```

void Back_People_B(int x,int y,int zoom,int color)
{
    zoom=5;
    //Putpixel64k(x,y,WHITE);
    Solid_Circle(x+11*zoom,y+11*zoom,5*zoom,BODYCOLOR_B);//head
    Solid_Bar(x+7*zoom,y+18*zoom,x+16*zoom,y+31*zoom,color);//bod
y
    Solid_Quadrangle(x+18*zoom,y+19*zoom,x+19*zoom,y+18*zoom,x+23
*zoom,y+24*zoom,x+24*zoom,y+23*zoom,color);//R
    Solid_Quadrangle(x+4*zoom,y+18*zoom,x+5*zoom,y+19*zoom,x+0*zo
om,y+26*zoom,x+1*zoom,y+27*zoom,color);
    Solid_Quadrangle(x+1*zoom,y+29*zoom,x+0*zoom,y+30*zoom,x+4*zo
om,y+34*zoom,x+3*zoom,y+35*zoom,BODYCOLOR_B);//L
    Solid_Quadrangle(x+14*zoom,y+35*zoom,x+16*zoom,y+33*zoom,x+19
*zoom,y+40*zoom,x+21*zoom,y+38*zoom,color);
    Solid_Quadrangle(x+21*zoom,y+40*zoom,x+24*zoom,y+41*zoom,x+19
*zoom,y+46*zoom,x+22*zoom,y+47*zoom,BODYCOLOR_B);//R
    Solid_Quadrangle(x+7*zoom,y+33*zoom,x+10*zoom,y+34*zoom,x+4*z
oom,y+40*zoom,x+8*zoom,y+41*zoom,color);//L
    Solid_Quadrangle(x+6*zoom,y+43*zoom,x+3*zoom,y+42*zoom,x-
1*zoom,y+46*zoom,x+2*zoom,y+48*zoom,BODYCOLOR_B);
}

```

```

        Solid_Quadrangle(x+125,y+121,x+122,y+126,x+151,y+141,x+154,y+
134,BODYCOLOR_B);
    }

```

```

/*****
*****
* 函数名称      Back_People_C
* 函数作用      绘制主界面C人
* 函数输入      x,y 坐标, zoom 放大倍数, color 衣服颜色
* 函数输出      无
*****
*****/

```

```

void Back_People_C(int x,int y,int zoom,int color)
{
    zoom=5;
    //Putpixel64k(x,y,WHITE);
    Solid_Circle(x-11*zoom,y+11*zoom,5*zoom,BODYCOLOR_C);//head
    Scarf(x-11*zoom,y+11*zoom,5*zoom+1,5*zoom,17,GRAY);
    Solid_Bar(x-7*zoom,y+18*zoom,x-
16*zoom,y+31*zoom,color);//body
    Solid_Quadrangle(x-18*zoom,y+19*zoom,x-19*zoom,y+18*zoom,x-
23*zoom,y+24*zoom,x-24*zoom,y+23*zoom,color);//R
    Solid_Quadrangle(x-4*zoom,y+18*zoom,x-5*zoom,y+19*zoom,x-
0*zoom,y+26*zoom,x-1*zoom,y+27*zoom,color);
    Solid_Quadrangle(x-1*zoom,y+29*zoom,x-0*zoom,y+30*zoom,x-
4*zoom,y+34*zoom,x-3*zoom,y+35*zoom,BODYCOLOR_C);//L
    Solid_Quadrangle(x-14*zoom,y+35*zoom,x-16*zoom,y+33*zoom,x-
19*zoom,y+40*zoom,x-21*zoom,y+38*zoom,color);
    Solid_Quadrangle(x-21*zoom,y+40*zoom,x-24*zoom,y+41*zoom,x-
19*zoom,y+46*zoom,x-22*zoom,y+47*zoom,BODYCOLOR_C);//R
    Solid_Quadrangle(x-7*zoom,y+33*zoom,x-10*zoom,y+34*zoom,x-
4*zoom,y+40*zoom,x-8*zoom,y+41*zoom,color);//L
    Solid_Quadrangle(x-6*zoom,y+43*zoom,x-
3*zoom,y+42*zoom,x+1*zoom,y+46*zoom,x-
2*zoom,y+48*zoom,BODYCOLOR_C);
    Solid_Quadrangle(x-125,y+121,x-122,y+126,x-151,y+141,x-
154,y+134,BODYCOLOR_C);
}

```

```

/*****
*****
* 函数名称      Back_Flag

```

```

* 函数作用      绘制主界面旗帜
* 函数输入      x,y 坐标, zoom 放大倍数
* 函数输出      无
*****
*****/

```

```

void Back_Flag(int x,int y,int zoom)
{
    Solid_Ellipse(x+8*zoom,y+15*zoom,6*zoom,3*zoom,BROWN);
    Solid_Bar(x+8*zoom,y+1*zoom,x+9*zoom,y+15*zoom,WHITE);
    Solid_Triangle(x+10*zoom,y+1*zoom,x+10*zoom,y+9*zoom,x+17*zoom,y+5*zoom,RED);
}

```

```

/*****
*****
* 函数名称      Back_Tree
* 函数作用      绘制主界面树
* 函数输入      x,y 坐标, zoom 放大倍数
* 函数输出      无
*****
*****/

```

```

void Back_Tree(int x,int y,int zoom)
{
    int i;
    Solid_Bar(x+8*zoom,y+21*zoom,x+15*zoom,y+32*zoom,BROWN);
    Solid_Quadrangle(x+8*zoom,y+32*zoom,x+15*zoom,y+32*zoom,x+7*zoom,y+40*zoom,x+16*zoom,y+40*zoom,BROWN);
    Solid_Quadrangle(x+7*zoom,y+40*zoom,x+16*zoom,y+40*zoom,x+6*zoom,y+44*zoom,x+17*zoom,y+44*zoom,BROWN);
    Solid_Circle(x+17*zoom,y+22*zoom,7*zoom,GREEN);
    Solid_Circle(x+6*zoom,y+22*zoom,7*zoom,GREEN);
    Solid_Circle(x+11*zoom,y+12*zoom,8*zoom,GREEN);
    Back_Redball(x+3*zoom,y+20*zoom);
    Back_Redball(x+7*zoom,y+14*zoom);
    Back_Redball(x+9*zoom,y+9*zoom);
    Back_Redball(x+11*zoom,y+20*zoom);
    Back_Redball(x+13*zoom,y+15*zoom);
    Back_Redball(x+16*zoom,y+10*zoom);
    Back_Redball(x+18*zoom,y+23*zoom);
}

```

```

/*****
*****

* 函数名称      Back_Redball
* 函数作用      绘制主界面苹果
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Back_Redball(int x,int y)
{
    int zoom=4;
    Solid_Circle(x,y,5,RED);
}

/*****
*****

* 函数名称      Back_Compass
* 函数作用      绘制主界面指南针
* 函数输入      x0,y0 坐标
* 函数输出      无
*****
*****/

void Back_Compass(int x0,int y0)
{
    int zoom=4;
    Solid_Circle(x0+12*zoom,y0+12*zoom,12*zoom,GOLDEN);
    Solid_Circle(x0+12*zoom,y0+12*zoom,10*zoom,BLACK);
    Solid_Triangle(x0+12*zoom,y0+1*zoom,x0+10*zoom,y0+11*zoom,x0+
15*zoom,y0+11*zoom,RED);
    Solid_Triangle(x0+13*zoom,y0+22*zoom,x0+10*zoom,y0+12*zoom,x0
+15*zoom,y0+12*zoom,WHITE);
}

/*****
*****

* 函数名称      Back_Point
* 函数作用      绘制主界面点位图像
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Back_Point(int x,int y)
{
    int ring_color=YELLOW;
    int one_color=LIME;
    int two_color=SKY_BLUE;
    int three_color=RED;
    int zoom=1;
    Ring(x+31*zoom,y+68*zoom,20*zoom,27*zoom,ring_color);
    Ring(x+90*zoom,y+27*zoom,20*zoom,27*zoom,ring_color);
    Ring(x+172*zoom,y+70*zoom,20*zoom,27*zoom,ring_color);
    Solid_Quadrangle(x+50*zoom,y+51*zoom,x+53*zoom,y+55*zoom,x+68
*zoom,y+39*zoom,x+71*zoom,y+43*zoom,ring_color);
    Solid_Quadrangle(x+114*zoom,y+37*zoom,x+111*zoom,y+41*zoom,x+
148*zoom,y+60*zoom,x+151*zoom,y+57*zoom,ring_color);
    Solid_Bar(x+29*zoom,y+55*zoom,x+33*zoom,y+80*zoom,one_color);
//1
    Solid_Bar(x+83*zoom,y+14*zoom,x+93*zoom,y+17*zoom,two_color);
    Solid_Bar(x+94*zoom,y+14*zoom,x+97*zoom,y+22*zoom,two_color);
    Solid_Bar(x+83*zoom,y+23*zoom,x+97*zoom,y+26*zoom,two_color);
    Solid_Bar(x+83*zoom,y+27*zoom,x+86*zoom,y+36*zoom,two_color);
    Solid_Bar(x+87*zoom,y+33*zoom,x+97*zoom,y+36*zoom,two_color);
//2
    Solid_Bar(x+165*zoom,y+59*zoom,x+176*zoom,y+62*zoom,three_col
or);
    Solid_Bar(x+165*zoom,y+69*zoom,x+176*zoom,y+72*zoom,three_col
or);
    Solid_Bar(x+165*zoom,y+79*zoom,x+176*zoom,y+82*zoom,three_col
or);
    Solid_Bar(x+177*zoom,y+59*zoom,x+180*zoom,y+82*zoom,three_col
or);//3
}

```

```

/*****
*****
* 函数名称      Back
* 函数作用      绘制主界面图像
* 函数输入      无
* 函数输出      无
*****
*****/

```

void Back()


```

{
    Back_People_A(74,421,5,RED);
    Back_People_B(174,441,5,BLUE);

    Back_Flag(65,72,10);
    Back_Point(464,60);
    Back_Compass(844,44);
    Back_Tree(846,315,4);
    Back_People_C(900,441,5,YELLOW);
}

/*****
*****
* 函数名称      Scarf
* 函数作用      绘制头巾
* 函数输入      x,y 坐标, radius 半径, up, down 头巾上下沿距离圆心位置, color 颜色
* 函数输出      无
*****
*****/

void Scarf(int x,int y,int radius,int up,int down,int color)
{
    double i,j;
    for(i=x-radius;i<=x+radius;i++)
    {
        for(j=y-radius;j<=y+radius;j++)
        {
            if((x-i)*(x-i)+(y-j)*(y-j)<=(double)radius*(double)radius&&j>=y-up&&j<=y-down)
            {
                Putpixel64k((int)i,(int)j,color);
            }
        }
    }
}

```

6. clock.c

```

#include"headfile.h"

/*****
*****
* @author      ytm
* @date        2022-4-15

```

```

*****
*****/

/*****
*****

* 函数名称      Clock_Random
* 函数作用      生成随机时间
* 函数输入      clock_hour, clock_minute 当前时间,
clock_hour_start, clock_minute_start 比赛开始时间
* 函数输出      无
*****
*****/

void Clock_Random(int* clock_hour,int* clock_minute,int *clock_ho
ur_start,int *clock_minute_start)
{
    randomize();
    (*clock_hour)=random(11)+5;
    (*clock_hour_start)=(*clock_hour);
    //make sure time reasonable
    (*clock_minute)=random(60);
    (*clock_minute_start)=(*clock_minute);
}

/*****
*****

* 函数名称      Clock_Display
* 函数作用      显示当前时间
* 函数输入      x0, y0 显示位置, hour, minute 当前时间, color 显示颜
色
* 函数注意      单字符大小 24*16
* 函数输出      无
*****
*****/

void Clock_Display(int x0,int y0,int hour,int minute,int color)
{
    int shiwei;
    int gewei;
    if(hour==0)
    {
        putsz(x0,y0,24,16,color,0);
        putsz(x0+16,y0,24,16,color,0);
    }
    else if(hour<10)

```

```

    {
        putsz(x0,y0,24,16,color,0);
        putsz(x0+16,y0,24,16,color,hour);
    }
    else
    {
        gewei=hour%10;
        shiwei=(hour-gewei)/10;
        putsz(x0,y0,24,16,color,shiwei);
        putsz(x0+16,y0,24,16,color,gewei);
    }
    putzm(x0+32,y0,24,16,color,":");
    if(minute==0)
    {
        putsz(x0+48,y0,24,16,color,0);
        putsz(x0+64,y0,24,16,color,0);
    }
    else if(minute<10)
    {
        putsz(x0+48,y0,24,16,color,0);
        putsz(x0+64,y0,24,16,color,minute);
    }
    else
    {
        gewei=minute%10;
        shiwei=(minute-gewei)/10;
        putsz(x0+48,y0,24,16,color,shiwei);
        putsz(x0+64,y0,24,16,color,gewei);
    }
}

/*****
*****
* 函数名称      SystemClock_Plus
* 函数作用      当前时间增加
* 函数输入      clock_hour, clock_minute 当前时间地址, plus_hour,
plus_minute 增加时间
* 函数输出      无
*****
*****/
void SystemClock_Plus(int* clock_hour,int* clock_minute,int plus_
hour,int plus_minute)
{

```

```

        (*clock_minute)+=plus_minute;
        if((*clock_minute)>=60)
        {
            (*clock_minute)-=60;
            clock_hour++;
        }
        (*clock_hour)+=plus_hour;
    }

/*****
*****
* 函数名称      Clock_Minus
* 函数作用      计算时间差值
* 函数输入      hour1, minute1 时间1, hour2, minute2 时间2
* 函数输出      时间差值绝对值
*****
*****/
int Clock_Minus(int hour1,int minute1,int hour2,int minute2)
{
    int temp;
    temp=(hour1-hour2)*60+minute1-minute2;
    if(temp<0)
    {
        temp=-temp;
    }
    return temp;
}

/*****
*****
* 函数名称      Clock_Compare
* 函数作用      比较时间先后
* 函数输入      hour1, minute1 时间1, hour2, minute2 时间2
* 函数输出      如果时间1 比时间2 早, 返回1
*                如果时间1 比时间2 晚, 返回-1
*                否则返回0
*****
*****/
int Clock_Compare(int hour1,int minute1,int hour2,int minute2)
{
    if(hour1<hour2)
    {
        return 1;
    }

```

```

    }
    else if(hour1>hour2)
    {
        return -1;
    }
    else
    {
        if(minute1<minute2)
        {
            return 1;
        }
        else if(minute1>minute2)
        {
            return -1;
        }
        else
        {
            return 0;
        }
    }
}

/*****
*****
* 函数名称      ClockEnd_Get
* 函数作用      得到比赛结束时间
* 函数输入      people 人数, point 点位数量, clock_hour_end,
clock_minute_end 比赛结束时间
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无
*****
*****/
void ClockEnd_Get(int people,int point,int *clock_hour_end,int *c
lock_minute_end, people_basic *person_basic, people_competition *
person_competition)
{
    int hour_end=person_competition[0].hour_point[point-
1],minute_end=person_competition[0].minute_point[point-1],i;
    for(i=1;i<people;i++)
    {
        if(Clock_Compare(hour_end,minute_end,person_competition[i
].hour_point[point-1],person_competition[i].minute_point[point-

```

```

1])<0)
    {
        hour_end=person_competition[i].hour_point[point-1];
        minute_end=person_competition[i].minute_point[point-
1];
    }
}
(*clock_hour_end)=hour_end,(*clock_minute_end)=minute_end;
}

```

7. create.c

```
#include"headfile.h"
```

```

/*****
*****
* 函数名称      Map_Create
* 函数作用      地图生成
* 函数输入      x0,y0 地图生成坐标, zone 地区模式
* 函数输出      无
*****
*****/

```

```

void Map_Create(int x0,int y0,int zone)
{
    int i,j; //扫描变量
    int draw_done[40][28]; //绘制状态数组
    int tree_type; //树的形式
    //Erase_Map();
    Draw_Compass(12,15); //指南针绘制
    switch(zone)
    {
        case 1:
            Solid_Bar(63,95,1024,768,YELLOW_GREEN);
            break;
        case 2:
            Solid_Bar(63,95,1024,768,LIME);
            break;
        case 3:
            Solid_Bar(63,95,1024,768,GRAY);
            break;
    } //选择区域对应的底板

    for(j=0;j<28;j++)
    {

```

```

        for(i=0;i<40;i++)
        {
            draw_done[i][j]=0;
        }
    } // 初始化绘制状态数组
    //check if draw
    if(zone==1)
    {
        //城区地图绘制
        for(j=0;j<28;j++)
        {
            for(i=0;i<40;i++)
            {
                //道路网络生成
                if(map_display[i][j]==3&&draw_done[i][j]==0)
                {
                    Urban_Road_Create(x0,y0,i,j,draw_done);
                    //Draw_H_Narrow_Highway_Part(x0+24*i,y0+24*j)
;

                }
                //河流生成
                if(map_display[i][j]==4)
                {
                    Draw_River_Icon(x0+24*i,y0+24*j);
                }
                //湖的生成
                if(map_display[i][j]==5)
                {
                    Draw_Urban_Lake(x0+24*i,y0+24*j);
                }
                //桥网络生成
                if(map_display[i][j]==6&&draw_done[i][j]==0)
                {
                    Urban_Bridge_Create(x0,y0,i,j,draw_done);
                    //    Draw_Wide_Bridge_Part(x0+i*24,y0+j*24,V
_ERTICAL);
                    //    Draw_Narrow_Bridge_Connection(x0+24*i,y
0+24*j,H_ORIZONTAL);
                }
                //建筑生成
                if(map_display[i][j]==1)
                {
                    Urban_Building_Create(x0,y0,i,j,draw_done);

```

```

    }
    //树的生成
    if(map_display[i][j]==2)
    {
        Urban_Tree_Create(x0,y0,i,j,draw_done);
    }
}
}
if(zone==2)
{
    //郊区地图绘制
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            //草地生成
            if(map_display[i][j]==2)
            {
                Draw_Suburban_Grass(x0+24*i,y0+24*j);
            }
            //田地生成
            if(map_display[i][j]==4)
            {
                Draw_Suburban_Field(x0+24*i,y0+24*j);
            }
            //河流生成
            if(map_display[i][j]==6)
            {
                Draw_River_Icon(x0+24*i,y0+24*j);
            }
            //桥网络生成
            if(map_display[i][j]==7&&draw_done[i][j]==0)
            {
                Suburban_Bridge_Create(x0,y0,i,j,draw_done);
            }
            //小径生成
            if(map_display[i][j]==5)
            {
                Path_Create(x0,y0,i,j,draw_done);
            }
            //道路网络生成
            if(map_display[i][j]==8)

```



```

        {
            Suburban_Road_Create(x0,y0,i,j,draw_done);
        }
        //泥土底板生成
        if(map_display[i][j]==3)
        {
            if(Field_Convert(i,j,0,0,1))
            {
                Solid_Bar(x0+24*i,y0+24*j,x0+24*i+23,y0+2
4*j+23,KHAKI);
            }
            Draw_Suburban_House(x0+24*i,y0+24*j);
        }
        //树生成
        if(map_display[i][j]==1)
        {
            Suburban_Tree_Create(x0,y0,i,j,draw_done);
        }
    }
}
if(zone==3)
{
    //山区地图绘制
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            //山区草地绘制
            if(map_display[i][j]==3)
            {
                Solid_Bar(x0+24*i,y0+24*j,x0+24*i+23,y0+24*j+
23,LIME);
                Draw_On_Mountainous_Grass(x0+24*i,y0+24*j);
            }
            //河流绘制
            if(map_display[i][j]==4)
            {
                Draw_River_Icon(x0+24*i,y0+24*j);
            }
            //湖绘制
            if(map_display[i][j]==5)
            {

```

```

        Draw_Mountainous_Lake(x0+24*i,y0+24*j);
    }
    // 沼泽绘制
    if(map_display[i][j]==6)
    {
        Draw_Mountainous_Swamp(x0+24*i,y0+24*j);
    }
    // 桥网络绘制
    if(map_display[i][j]==7&&draw_done[i][j]==0)
    {
        Mountainous_Bridge_Create(x0,y0,i,j,draw_done
);
        //Draw_Narrow_Bridge_Connection(x0+24*i,y0+24
*j,H_ORIZONTAL);
    }
    // 小屋绘制
    if(map_display[i][j]==8)
    {
        Solid_Bar(x0+24*i,y0+24*j,x0+24*i+23,y0+24*j+
23,LIME);
        Draw_Mountainous_House(x0+24*i,y0+24*j);
    }
    // 平地树绘制
    if(map_display[i][j]==2)
    {
        tree_type=random(100);
        Solid_Bar(x0+24*i,y0+24*j,x0+24*i+23,y0+24*j+
23,LIME);
        if(tree_type<=50)
        {
            Draw_Mountainous_Bush(x0+24*i,y0+24*j);
        }
        else
        {
            Draw_Mountainous_Tree(x0+24*i,y0+24*j);
        }
    }
    // 山区树绘制
    if(map_display[i][j]==12)
    {
        //Draw_Mountain(x0+24*i,y0+24*j);
        tree_type=random(2);
        if(tree_type==0)

```

```

        {
            Draw_Mountainous_Bush(x0+24*i,y0+24*j);
        }
        else
        {
            Draw_Mountainous_Tree(x0+24*i,y0+24*j);
        }
    }
    // 山区草绘制
    if(map_display[i][j]==13)
    {
        Draw_On_Mountainous_Grass(x0+24*i,y0+24*j);
    }
}
}
//getch();
}

```

```

/*****
*****
* 函数名称      Urban_Building_Create
* 函数作用      城区建筑生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制
                  完成数组
* 函数输出      无
*****
*****/

```

```

void Urban_Building_Create(int x0,int y0,int i,int j,int (*draw_d
one)[28])
{
    int scanx,scany;//区块扫描坐标
    int housesize=0,housecolor=0;//屋子性质
    int randomcolor=0;//随机颜色数
    const int hospital=5;//医院比例数
    for(scanx=0;scanx<3;scanx++)
    {
        for(scany=0;scany<3;scany++)
        {
            if(map_display[i+scanx][j+scany]==1&&draw_done[i+scan
x][j+scany]==0)
            {

```

```

        housesize++;
    }
}
} // 扫描可放置建筑的位置

if(housesize==9)//big house
{
    randomcolor=4;
    housecolor=random(randomcolor);
    if(housecolor==0)
    {
        Draw_Urban_Store(x0+24*i,y0+24*j,GRAY);
    }
    if(housecolor==1)
    {
        Draw_Urban_Store(x0+24*i,y0+24*j,HONEY_ORANGE);
    }
    if(housecolor==2)
    {
        Draw_Urban_Store(x0+24*i,y0+24*j,DIMGRAY);
    }
    if(housecolor==3)
    {
        Draw_Urban_Store(x0+24*i,y0+24*j,POWDER_BLUE);
    }

    for(scanx=0;scanx<3;scanx++)
    {
        for(scany=0;scany<3;scany++)
        {
            draw_done[i+scanx][j+scany]=1;
        }
    }
} // 商店绘制
else
{
    housesize=0;
    for(scanx=0;scanx<2;scanx++)
    {
        for(scany=0;scany<2;scany++)
        {
            if(map_display[i+scanx][j+scany]==1&&draw_done[i+
scanx][j+scany]==0)

```

```

        {
            housesize++;
        }
    }
}
if(housesize==4)//mid house or hospital
{
    if(random(hospital)==0)
    {
        Draw_Urban_Hospital(x0+24*i,y0+24*j);
    }
    else
    {
        randomcolor=3;
        housecolor=random(randomcolor);
        if(housecolor==0)
        {
            Draw_Urban_Midhouse(x0+24*i,y0+24*j,ORANGE,WH
ITE);
        }
        if(housecolor==1)
        {
            Draw_Urban_Midhouse(x0+24*i,y0+24*j,BROWN,GRAY);
        }
        if(housecolor==2)
        {
            Draw_Urban_Midhouse(x0+24*i,y0+24*j,HOT_PINK,MARIGOLD);
        }
    }
}
for(scanx=0;scanx<2;scanx++)
{
    for(scany=0;scany<2;scany++)
    {
        draw_done[i+scanx][j+scany]=1;
    }
}
} // 中型房屋
else if(draw_done[i][j]==0)
{
    randomcolor=6;
    housecolor=random(randomcolor);

```

```

        if(housecolor==0)
        {
            Draw_Urban_House(x0+24*i,y0+24*j,GRAY);
        }
        if(housecolor==1)
        {
            Draw_Urban_House(x0+24*i,y0+24*j,MARIGOLD);
        }
        if(housecolor==2)
        {
            Draw_Urban_House(x0+24*i,y0+24*j,WHITE);
        }
        if(housecolor==3)
        {
            Draw_Urban_House(x0+24*i,y0+24*j,POWDER_BLUE);
        }
        if(housecolor==4)
        {
            Draw_Urban_House(x0+24*i,y0+24*j,BROWN);
        }
        if(housecolor==5)
        {
            Draw_Urban_House(x0+24*i,y0+24*j,OLIVE);
        }
        draw_done[i][j]=1;
        // 小房屋
    }
}

}

/*****
*****
* 函数名称      Urban_Road_Create
* 函数作用      城区道路生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制
                  完成数组
* 函数输出      无
*****
*****/

void Urban_Road_Create(int x0,int y0,int i,int j,int (*draw_done)
[28])
{

```

```

int main_V_x,main_H_y;
int added_V_x,added_H_y;
int up_x,down_x,left_y,right_y; //记录道路信息变量
int bridge_length,bridge_scan,count; //扫描用变量
int forward,edge; //基础信息变量
main_V_x=Main_Road_Convert(0,V_ERTICAL,1);
main_H_y=Main_Road_Convert(0,H_ORIZONTAL,1);
up_x=Narrow_Road_Convert(0,U_P,1);
down_x=Narrow_Road_Convert(0,D_OWN,1);
left_y=Narrow_Road_Convert(0,L_EFT,1);
right_y=Narrow_Road_Convert(0,R_IGHT,1);
added_V_x=Added_Road_Convert(0,V_ERTICAL,1);
added_H_y=Added_Road_Convert(0,H_ORIZONTAL,1); //道路信息接口
传入

```

```

if(i==main_V_x&&j==main_H_y)
{
    Draw_Main_Crossing(x0+i*24,y0+j*24);
    draw_done[i][j]=1;
    draw_done[i+1][j]=1;
    draw_done[i][j+1]=1;
    draw_done[i+1][j+1]=1;
} // 十字路口扫描
else if(i==main_V_x)
{
    Draw_Urban_V_Wide_Highway_Part(x0+i*24,y0+j*24);
    draw_done[i][j]=1;
    draw_done[i+1][j]=1;
}
else if(j==main_H_y)
{
    Draw_Urban_H_Wide_Highway_Part(x0+i*24,y0+j*24);
    draw_done[i][j]=1;
    draw_done[i][j+1]=1;
} // 两主路绘制
else if(Scan_Small_Crossing(i,j)==1&&Scan_Edge(i,j)==0)
{
    Draw_Small_Crossing(x0+i*24,y0+j*24);
} // 小路十字路口扫描
else
{
    edge=Scan_Edge(i,j);
    if(edge)

```

```

    {
        forward=Scan_Edge_Road(i,j,edge);
        if(forward==H_ORIZONTAL)
        {
            Draw_H_Narrow_Highway_Part(x0+i*24,y0+j*24);
        }
        if(forward==V_ERTICAL)
        {
            Draw_V_Narrow_Highway_Part(x0+i*24,y0+j*24);
        }
    } // 边缘道路扫描
    else if(Scan_Narrow_Road(i,j,V_ERTICAL)==1)
    {
        Draw_V_Narrow_Highway_Part(x0+i*24,y0+j*24);
    }
    else if(Scan_Narrow_Road(i,j,H_ORIZONTAL)==1)
    {
        Draw_H_Narrow_Highway_Part(x0+i*24,y0+j*24);
    } // 普通小路绘制
}
}

/*****
*****
* 函数名称      Main_Road_Convert
* 函数作用      主路坐标传入
* 函数输入      point 坐标输入, forward 水平与垂直输入, mode 模式
* 函数输出      模式1 返回寄存坐标, 模式2 返回读取是否成功
*****
*****/

```

```

int Main_Road_Convert(int point,int forward,int mode)
{
    static int x_for_H=-1,y_for_V=-1;//main Horizontal
    //mode 1 is read, mode 2 is write
    if(mode==2)
    {
        if(forward==H_ORIZONTAL)
        {
            x_for_H=point;
            return -1;
        }
        if(forward==V_ERTICAL)

```



```

        {
            y_for_V=point;
            return -1;
        }
        else return -2;
    }
    if(mode==1)
    {
        if(forward==H_ORIZONTAL)
        {
            return x_for_H;
        }
        if(forward==V_ERTICAL)
        {
            return y_for_V;
        }
        else return -2;
    }
}

/*****
*****
* 函数名称      Suburban_Road_Convert
* 函数作用      郊区道路传入
* 函数输入      point 坐标输入, forward 水平与垂直输入, mode 模式
* 函数输出      模式1 返回寄存坐标, 模式2 返回读取是否成功
*****
*****/

```

```

int Suburban_Road_Convert(int point,int forward,int mode)
{
    static int temp=0;//main Horizontal
    //mode 1 is read, mode 2 is write
    if(mode==2)
    {
        if(forward==H_ORIZONTAL)
        {
            temp=point;
            return -1;
        }
        if(forward==V_ERTICAL)
        {
            temp=-point;

```

```

        return -1;
    }
    else return -2;
}
if(mode==1)
{
    return temp;
}
}

/*****
*****
* 函数名称      Narrow_Road_Convert
* 函数作用      小路传入
* 函数输入      point 坐标输入, forward 水平与垂直输入, mode 模式
* 函数输出      模式1 返回寄存坐标, 模式2 返回读取是否成功
*****
*****/

int Narrow_Road_Convert(int point,int forward,int mode)
{
    static int x_for_UP=-1,x_for_DOWN=-1,y_for_LEFT=-
1,y_for_RIGHT;//main Horizontal
    //mode 1 is read, mode 2 is write
    if(mode==2)
    {
        if(forward==U_P)
        {
            x_for_UP=point;
            return -1;
        }
        if(forward==D_OWN)
        {
            x_for_DOWN=point;
            return -1;
        }
        if(forward==L_EFT)
        {
            y_for_LEFT=point;
            return -1;
        }
        if(forward==R_IGHT)
        {

```

```

        y_for_RIGHT=point;
        return -1;
    }
    else return -2;
}
if(mode==1)
{
    if(forward==U_P)
    {
        return x_for_UP;
    }
    if(forward==D_DOWN)
    {
        return x_for_DOWN;
    }
    if(forward==L_LEFT)
    {
        return y_for_LEFT;
    }
    if(forward==R_RIGHT)
    {
        return y_for_RIGHT;
    }
    else return -2;
}
}

/*****
*****
* 函数名称      Added_Road_Convert
* 函数作用      附加路传入
* 函数输入      point 坐标输入, forward 水平与垂直输入, mode 模式
* 函数输出      模式1 返回寄存坐标, 模式2 返回读取是否成功
*****
*****/

```

```

int Added_Road_Convert(int point,int forward,int mode)
{
    static int x_for_H=-1,y_for_V=-1;//main Horizontal
    //mode 1 is read, mode 2 is write
    if(mode==2)
    {
        if(forward==H_ORIZONTAL)

```

```

        {
            x_for_H=point;
            return -1;
        }
        if(forward==V_ERTICAL)
        {
            y_for_V=point;
            return -1;
        }
        else return -2;
    }
    if(mode==1)
    {
        if(forward==H_ORIZONTAL)
        {
            return x_for_H;
        }
        if(forward==V_ERTICAL)
        {
            return y_for_V;
        }
        else return -2;
    }
}

/*****
*****
* 函数名称      Urban_Tree_Create
* 函数作用      城区树生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制
                完成数组
* 函数输出      无
*****
*****/

void Urban_Tree_Create(int x0,int y0,int i,int j,int (*draw_done)
[28])
{
    int treemode=3,rantree;
    rantree=random(treemode);
    if(rantree==0)
    {
        Draw_Urban_Flower(x0+24*i,y0+24*j);
    }
}

```

```

    }
    else
    {
        Draw_Urban_Tree(x0+24*i,y0+24*j);
    }
}

/*****
*****
* 函数名称      Suburban_Road_Create
* 函数作用      郊区道路生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制
                完成数组
* 函数输出      无
*****
*****/

void Suburban_Road_Create(int x0,int y0,int i,int j,int (*draw_done)[28])
{
    int x_for_V=0,y_for_H=0;
    if(Suburban_Road_Convert(0,0,1)<0)
    {
        x_for_V=-Suburban_Road_Convert(0,0,1);
        if(i==x_for_V)
        {
            Draw_Urban_V_Wide_Highway_Part(x0+24*i,y0+24*j);
        }
    }
    else
    {
        y_for_H=Suburban_Road_Convert(0,0,1);
        if(j==y_for_H)
        {
            Draw_Urban_H_Wide_Highway_Part(x0+24*i,y0+24*j);
        }
    }
}

/*****
*****
* 函数名称      Path_Create
* 函数作用      小径生成

```

* 函数输入 x_0, y_0 地图生成坐标, i, j 地图相对坐标, $draw_done$ 绘制完成数组

* 函数输出 无

 *****/

```
void Path_Create(int x0,int y0,int i,int j,int (*draw_done)[28])
{
    int x_for_V=0,y_for_H=0;
    x_for_V=Main_Road_Convert(0,V_ERTICAL,1);
    y_for_H=Main_Road_Convert(0,H_ORIZONTAL,1);
    if(i==x_for_V)
    {
        Draw_Suburban_Path_Part(x0+24*i,y0+24*j,2);
    }
    else if(j==y_for_H)
    {
        Draw_Suburban_Path_Part(x0+24*i,y0+24*j,1);
    }
}
```

 *****/

* 函数名称 *Suburban_Tree_Create*

* 函数作用 郊区树生成

* 函数输入 x_0, y_0 地图生成坐标, i, j 地图相对坐标, $draw_done$ 绘制完成数组

* 函数输出 无

 *****/

```
void Suburban_Tree_Create(int x0,int y0,int i,int j,int (*draw_done)[28])
{
    int treemode=3,rantree;
    rantree=random(treemode);
    if(rantree==0)
    {
        Draw_Suburban_Bush(x0+24*i,y0+24*j);
    }
    else if(rantree==1)
    {
        Draw_Suburban_Appletree(x0+24*i,y0+24*j);
    }
}
```

```

    }
    else if(rantree==2)
    {
        Draw_Suburban_Sakura(x0+24*i,y0+24*j);
    }
}

/*****
*****
* 函数名称      Scan_Small_Crossing
* 函数作用      扫描小路十字路口
* 函数输入      i,j 地图相对坐标
* 函数输出      是返回1，否返回0
*****
*****/

int Scan_Small_Crossing(int i,int j)
{
    int road_count=0;
    if(map_display[i+1][j]==3)
    {
        road_count++;
    }
    if(map_display[i][j+1]==3)
    {
        road_count++;
    }
    if(map_display[i-1][j]==3)
    {
        road_count++;
    }
    if(map_display[i][j-1]==3)
    {
        road_count++;
    }
    // 扫描接入道路的路口，超过3 即为交叉路口
    if(road_count==3||road_count==4)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

```

```

    }
}

/*****
*****

* 函数名称      Scan_Narrow_Road
* 函数作用      扫描周围小路
* 函数输入      i,j 地图相对坐标
* 函数输出      是返回1, 否返回0
*****
*****/

int Scan_Narrow_Road(int i,int j,int forward)
{
    //扫描道路延续性
    if(forward==V_ERTICAL)
    {
        if((map_display[i][j+1]!=3&&map_display[i][j+1]!=6)&&(map_display[i][j-1]!=3&&map_display[i][j-1]!=6))
        {
            return 0;
        }
        else return 1;
    }
    if(forward==H_ORIZONTAL)
    {
        if((map_display[i+1][j]!=3&&map_display[i+1][j]!=6)&&(map_display[i-1][j]!=3&&map_display[i-1][j]!=6))
        {
            return 0;
        }
        else return 1;
    }
}

/*****
*****

* 函数名称      Scan_Edge
* 函数作用      扫描边界
* 函数输入      i,j 地图相对坐标
* 函数输出      是返回1, 否返回0
*****
*****/

```



```

int Scan_Edge(int i,int j)
{
    //扫描是否为边界, 且返回边界信息
    if(j==0)
    {
        return U_P;
    }
    else if(j==27)
    {
        return D_DOWN;
    }
    else if(i==0)
    {
        return L_LEFT;
    }
    else if(i==39)
    {
        return R_RIGHT;
    }
    else
    {
        return 0;
    }
}

/*****
*****
* 函数名称      Urban_Bridge_Create
* 函数作用      城区桥生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制
                  完成数组
* 函数输出      无
*****
*****/

void Urban_Bridge_Create(int x0,int y0,int i,int j,int (*draw_done)
[28])
{
    int main_V_x,main_H_y;
    int H_length,V_length;//桥信息传入
    int count;//桥长扫描变量
    int forward;//方向信息

```

```

main_V_x=Main_Road_Convert(0,V_ERTICAL,1);
main_H_y=Main_Road_Convert(0,H_ORIZONTAL,1);// 桥信息接口传入
if(i==main_V_x&&j==main_H_y)
{
    // 交叉路口传入
    Draw_Main_Crossing(x0+24*i,y0+24*j);
    draw_done[i][j]=1;
    draw_done[i+1][j]=1;
    draw_done[i][j+1]=1;
    draw_done[i+1][j+1]=1;
    Draw_Wide_Bridge_End(i,j,L_EFT);
}
else if(j==main_H_y)
{
    // 主路桥扫描
    H_length=Scan_Bridge(i,j,H_ORIZONTAL);// 桥长扫描
    if(H_length==1)
    {
        Draw_Urban_H_Wide_Highway_Part(x0+24*i,y0+24*j);
        draw_done[i][j]=1;
        draw_done[i][j+1]=1;
    } // 单长度桥直接绘制
    else
    {
        Draw_River_Icon(x0+24*i,y0+24*j);
        Draw_River_Icon(x0+24*i,y0+24*j+24);
        Draw_Wide_Bridge_End(x0+24*i,y0+24*j,L_EFT);// 绘制桥

        端点
        for(count=1;count<H_length-1;count++)
        {
            Draw_River_Icon(x0+24*(i+count),y0+24*j);
            Draw_River_Icon(x0+24*(i+count),y0+24*j+24);
            Draw_Wide_Bridge_Part(x0+24*(i+count),y0+24*j,H_O
RIZONTAL);// 根据桥长绘制桥中间部分
        }
        Draw_River_Icon(x0+24*(i+H_length-1),y0+24*j);
        Draw_River_Icon(x0+24*(i+H_length-1),y0+24*j+24);
        Draw_Wide_Bridge_End(x0+24*(i+H_length-
1),y0+24*j,R_IGHT);// 绘制桥另一端点
        for(count=0;count<H_length;count++)
        {
            draw_done[i+count][j]=1;
            draw_done[i+count][j+1]=1;

```

```

        }// 记录绘制信息
    }
}
else if(i==main_V_x)
{
    V_length=Scan_Bridge(i,j,V_ERTICAL);// 桥长扫描
    if(V_length==1)
    {
        Draw_Urban_V_Wide_Highway_Part(x0+24*i,y0+24*j);
        draw_done[i][j]=1;
        draw_done[i+1][j]=1;
    }// 单长度桥直接绘制
    else
    {
        Draw_River_Icon(x0+24*i,y0+24*j);
        Draw_River_Icon(x0+24*i+24,y0+24*j);
        Draw_Wide_Bridge_End(x0+24*i,y0+24*j,U_P);// 绘制桥端

        for(count=1;count<V_length-1;count++)
        {
            Draw_River_Icon(x0+24*i,y0+24*(j+count));
            Draw_River_Icon(x0+24*i+24,y0+24*(j+count));
            Draw_Wide_Bridge_Part(x0+24*i,y0+24*(j+count),V_
RTICAL);// 根据桥长绘制桥中间部分
        }
        Draw_River_Icon(x0+24*i,y0+24*(j+V_length-1));
        Draw_River_Icon(x0+24*i+24,y0+24*(j+V_length-1));
        Draw_Wide_Bridge_End(x0+24*i,y0+24*(j+V_length-
1),D_OWN);// 绘制桥另一端点
        for(count=0;count<V_length;count++)
        {
            draw_done[i][j+count]=1;
            draw_done[i+1][j+count]=1;
        }// 记录绘制信息
    }
}
else
{
    // 小桥绘制
    if(Scan_Bridge_Crossing==1)
    {
        Draw_Wide_Bridge_Part(x0+24*i,y0+24*j,C_ENTER);
    } // 桥交叉绘制
}

```

```

    if(Scan_Edge(i,j))
    {
        forward=Scan_Edge_Road(i,j,Scan_Edge(i,j));
    } //扫描边界
    else
    {
        if(Scan_Narrow_Road(i,j,H_ORIZONTAL))
        {
            forward=H_ORIZONTAL;
        }
        else
        {
            forward=V_ERTICAL;
        }
    } //扫描桥方向

    if(forward==H_ORIZONTAL)
    {
        H_length=Scan_Bridge(i,j,H_ORIZONTAL); //扫描桥长
        if(H_length==1)
        {
            Draw_Narrow_Bridge_Connection(x0+24*i,y0+24*j,H_0
RIZONTAL);
        } //单桥长直接绘制
        else
        {
            if(draw_done[i][j]==0)
            {
                Draw_River_Icon(x0+24*i,y0+24*j);
                //Draw_River_Icon(x0+24*i+24,y0+24*j);
                Draw_Narrow_Bridge_End(x0+24*i,y0+24*j,R_IGHT
);
            }
            //桥端点绘制
            for(count=1;count<H_length-1;count++)
            {
                if(draw_done[i+count][j])
                {
                    Draw_River_Icon(x0+24*(i+count),y0+24*j);
                    //Draw_River_Icon(x0+24*(i+count),y0+24*j
+24);
                    Draw_Narrow_Bridge_Connection(x0+24*(i+co
unt),y0+24*j,H_ORIZONTAL);
                }
            }
        }
    }
}

```

```

        } //根据桥长绘制桥中部
        if(draw_done[i+H_length-1][j]==0)
        {
            Draw_River_Icon(x0+24*(i+H_length-
1),y0+24*j);
            //Draw_River_Icon(x0+24*(i+H_length-
1),y0+24*j+24);
            Draw_Narrow_Bridge_End(x0+24*(i+H_length-
1),y0+24*j,L_EFT);
        } //绘制桥另一端点
        for(count=0;count<H_length;count++)
        {
            draw_done[i+count][j]=1;
        } //记录绘制数据
    }
}
else if(forward==V_ERTICAL)
{
    V_length=Scan_Bridge(i,j,V_ERTICAL); //扫描桥长
    if(V_length==1)
    {
        Draw_Narrow_Bridge_Connection(x0+24*i,y0+24*j,V_E
RTICAL);
    } //单桥长直接绘制
    else
    {
        if(draw_done[i][j]==0)
        {
            Draw_River_Icon(x0+24*i,y0+24*j);
            //Draw_River_Icon(x0+24*i+24,y0+24*j);
            Draw_Narrow_Bridge_End(x0+24*i,y0+24*j,D_OWN)
;
        } //桥端点绘制
        for(count=1;count<V_length-1;count++)
        {
            if(draw_done[i][j+count]==0)
            {
                Draw_River_Icon(x0+24*i,y0+24*(j+count));
                //Draw_River_Icon(x0+24*i+24,y0+24*(j+cou
nt));
                Draw_Narrow_Bridge_Connection(x0+24*i,y0+
24*(j+count),V_ERTICAL);
            }
        }
    }
}

```

```

        } //根据桥长绘制桥中部
        if(draw_done[i][j+V_length-1]==0)
        {
            Draw_River_Icon(x0+24*i,y0+24*(j+V_length-
1));

            //Draw_River_Icon(x0+24*i+24,y0+24*(j+V_Lengt
h-1));

            Draw_Narrow_Bridge_End(x0+24*i,y0+24*(j+V_len
gth-1),U_P);
        } //绘制桥另一端点
        for(count=0;count<V_length;count++)
        {
            draw_done[i][j+count]=1;
        } //记录绘制数据
    }
}
}
}
}

```

```

/*****
*****
* 函数名称      Scan_Bridge
* 函数作用      桥长扫描
* 函数输入      i,j 地图相对坐标
* 函数输出      桥长
*****
*****/

```

```

int Scan_Bridge(int i,int j,int forward)
{
    int length=1;
    if(forward==H_ORIZONTAL)
    {
        for(length=1;map_display[i+length][j]==6;length++);
        return length;
    }
    if(forward==V_ERTICAL)
    {
        for(length=1;map_display[i][j+length]==6;length++);
        return length;
    }
}

```

```

/*****
*****
* 函数名称      Scan_Edge_Road
* 函数作用      扫描边界路
* 函数输入      i,j 地图相对坐标, forward 扫描方向
* 函数输出      水平或垂直
*****
*****/

```

```

int Scan_Edge_Road(int i,int j,int forward)
{
    //根据边界情况, 扫描返回路的方向
    if(forward==U_P)
    {
        if(map_display[i][j+1]!=3&&map_display[i][j+1]!=6)
        {
            if(Scan_Narrow_Road(i,j,H_ORIZONTAL))
            {
                return H_ORIZONTAL;
            }
        }
        else
        {
            return V_ERTICAL;
        }
    }
    if(forward==D_DOWN)
    {
        if(map_display[i][j-1]!=3&&map_display[i][j-1]!=6)
        {
            if(Scan_Narrow_Road(i,j,H_ORIZONTAL))
            {
                return H_ORIZONTAL;
            }
        }
        else
        {
            return V_ERTICAL;
        }
    }
    if(forward==L_LEFT)
    {
        if(map_display[i+1][j]!=3&&map_display[i+1][j]!=6)

```

```

        {
            if(Scan_Narrow_Road(i,j,V_ERTICAL))
            {
                return V_ERTICAL;
            }
        }
        else
        {
            return H_ORIZONTAL;
        }
    }
    if(forward==R_IGHT)
    {
        if(map_display[i-1][j]!=3&&map_display[i-1][j]!=6)
        {
            if(Scan_Narrow_Road(i,j,V_ERTICAL))
            {
                return V_ERTICAL;
            }
        }
        else
        {
            return H_ORIZONTAL;
        }
    }
}

/*****
*****
* 函数名称      Scan_Bridge_Crossing
* 函数作用      扫描桥交叉
* 函数输入      i,j 地图相对坐标
* 函数输出      是返回1, 否返回0
*****
*****/

int Scan_Bridge_Crossing(int i,int j)
{
    //扫描桥交叉情况
    int road_count=0;
    if(map_display[i+1][j]==3||map_display[i+1][j]==6)
    {
        road_count++;
    }
}

```



```

    }
    if(map_display[i][j+1]==3||map_display[i][j+1]==6)
    {
        road_count++;
    }
    if(map_display[i-1][j]==3||map_display[i-1][j]==6)
    {
        road_count++;
    }
    if(map_display[i][j-1]==3||map_display[i][j-1]==6)
    {
        road_count++;
    }
    // 超过 3 个路口即返回交叉桥信息
    if(road_count==3||road_count==4)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

/*****
*****
* 函数名称      Erase_Map
* 函数作用      地图擦除
* 函数输入      无
* 函数输出      无
*****
*****/

void Erase_Map()
{
    Solid_Bar(63,71,1024,95,BLACK);
}

/*****
*****
* 函数名称      Suburban_Bridge_Create
* 函数作用      郊区桥生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制

```

完成数组

* 函数输出 无

```
*****  
*****/
```

```
void Suburban_Bridge_Create(int x0,int y0,int i,int j,int (*draw_  
done)[28])
```

```
{  
    int x_for_V=0,y_for_H=0;  
    int H_length,V_length; //桥长信息  
    int count; //桥长扫描变量  
    int x_Narrow,y_Narrow; //桥缓存  
    if(Suburban_Road_Convert(0,0,1)<0)  
    {  
        x_for_V=-Suburban_Road_Convert(0,0,1); //扫描桥长  
        if(i==x_for_V)  
        {  
            V_length=Scan_Suburban_Bridge(i,j,V_ERTICAL);  
            if(V_length==1)  
            {  
                Draw_Urban_V_Wide_Highway_Part(x0+24*i,y0+24*j);  
            } //单桥长直接绘制  
            else  
            {  
                Draw_River_Icon(x0+24*i,y0+24*j);  
                Draw_River_Icon(x0+24*i+24,y0+24*j);  
                Draw_Wide_Bridge_End(x0+24*i,y0+24*j,U_P); //绘制  
                for(count=1;count<V_length-1;count++)  
                {  
                    Draw_River_Icon(x0+24*i,y0+24*(j+count));  
                    Draw_River_Icon(x0+24*i+24,y0+24*(j+count));  
                    Draw_Wide_Bridge_Part(x0+24*i,y0+24*(j+count)  
                    ,V_ERTICAL); //根据桥长绘制桥中部  
                }  
                Draw_River_Icon(x0+24*i,y0+24*(j+V_length-1));  
                Draw_River_Icon(x0+24*i+24,y0+24*(j+V_length-1));  
                Draw_Wide_Bridge_End(x0+24*i,y0+24*(j+V_length-  
                1),D_OWN); //绘制桥另一端点  
                for(count=0;count<V_length;count++)  
                {  
                    draw_done[i][j+count]=1;  
                    draw_done[i+1][j+count]=1;  
                }  
            }  
        }  
    }  
}
```

桥一端点

```

        } //记录绘制信息
    }
}
else
{
    y_for_H=Suburban_Road_Convert(0,0,1); //位置传入
    if(j==y_for_H)
    {
        H_length=Scan_Suburban_Bridge(i,j,H_ORIZONTAL); //扫描桥长
        if(H_length==1)
        {
            Draw_Urban_H_Wide_Highway_Part(x0+24*i,y0+24*j);
        } //单桥长直接绘制
        else
        {
            Draw_River_Icon(x0+24*i,y0+24*j);
            Draw_River_Icon(x0+24*i,y0+24*j+24);
            Draw_Wide_Bridge_End(x0+24*i,y0+24*j,L_EFT); //绘制桥左端点
            for(count=1;count<H_length-1;count++)
            {
                Draw_River_Icon(x0+24*(i+count),y0+24*j);
                Draw_River_Icon(x0+24*(i+count),y0+24*j+24);
                Draw_Wide_Bridge_Part(x0+24*(i+count),y0+24*j,H_ORIZONTAL); //根据桥长绘制桥中部
            }
            Draw_River_Icon(x0+24*(i+H_length-1),y0+24*j);
            Draw_River_Icon(x0+24*(i+H_length-1),y0+24*j+24);
            Draw_Wide_Bridge_End(x0+24*(i+H_length-1),y0+24*j,R_IGHT); //绘制桥右端点
            for(count=0;count<H_length;count++)
            {
                draw_done[i+count][j]=1;
                draw_done[i+count][j+1]=1;
            } //记录绘制信息
        }
    }
}
x_Narrow=Main_Road_Convert(0,V_ERTICAL,1);
y_Narrow=Main_Road_Convert(0,H_ORIZONTAL,1); //传入道路信息
if(i==x_Narrow&&j==y_Narrow)

```

```

{
    if(Scan_Sub_Bridge_Crossing==1)
    {
        Draw_Wide_Bridge_Part(x0+24*i,y0+24*j,C_ENTER);
    }
} // 绘制桥交叉
else if(i==x_Narrow)
{
    V_length=Scan_Suburban_Bridge(i,j,V_ERTICAL);
    if(V_length==1)
    {
        Draw_Narrow_Bridge_Connection(x0+24*i,y0+24*j,V_ERTIC
AL);
    } // 单桥长绘制
    else
    {
        Draw_River_Icon(x0+24*i,y0+24*j);
        Draw_River_Icon(x0+24*i+24,y0+24*j);
        Draw_Narrow_Bridge_End(x0+24*i,y0+24*j,D_DOWN); // 绘制
桥一端点
        for(count=1;count<V_length-1;count++)
        {
            Draw_River_Icon(x0+24*i,y0+24*(j+count));
            Draw_River_Icon(x0+24*i+24,y0+24*(j+count));
            Draw_Narrow_Bridge_Connection(x0+24*i,y0+24*(j+co
unt),V_ERTICAL); // 根据桥长绘制桥中部
        }
        Draw_River_Icon(x0+24*i,y0+24*(j+V_length-1));
        Draw_River_Icon(x0+24*i+24,y0+24*(j+V_length-1));
        Draw_Narrow_Bridge_End(x0+24*i,y0+24*(j+V_length-
1),U_P); // 绘制桥另一端点
        for(count=0;count<V_length;count++)
        {
            draw_done[i][j+count]=1;
        } // 记录绘制信息
    }
}
else if(j==y_Narrow)
{
    H_length=Scan_Suburban_Bridge(i,j,H_ORIZONTAL); // 桥长扫
描
    if(H_length==1)
    {

```

```

        Draw_Narrow_Bridge_Connection(x0+24*i,y0+24*j,H_ORIZO
NTAL);
    } // 单桥长绘制
    else
    {
        Draw_River_Icon(x0+24*i,y0+24*j);
        Draw_River_Icon(x0+24*i+24,y0+24*j);
        Draw_Narrow_Bridge_End(x0+24*i,y0+24*j,R_IGHT); // 绘
制桥一 endpoint
        for(count=1;count<H_length-1;count++)
        {
            Draw_River_Icon(x0+24*(i+count),y0+24*j);
            Draw_River_Icon(x0+24*(i+count),y0+24*j+24);
            Draw_Narrow_Bridge_Connection(x0+24*(i+count),y0+
24*j,H_ORIZONTAL); // 根据桥长绘制桥中部
        }
        Draw_River_Icon(x0+24*(i+H_length-1),y0+24*j);
        Draw_River_Icon(x0+24*(i+H_length-1),y0+24*j+24);
        Draw_Narrow_Bridge_End(x0+24*(i+H_length-
1),y0+24*j,L_EFT); // 绘制桥另一端点
        for(count=0;count<H_length;count++)
        {
            draw_done[i+count][j]=1;
        } // 记录绘制信息
    }
}
}

```

```

/*****
*****
* 函数名称      Scan_Suburban_Bridge
* 函数作用      郊区桥长扫描
* 函数输入      i,j 地图相对坐标, forward 方向
* 函数输出      桥长
*****
*****/

```

```

int Scan_Suburban_Bridge(int i,int j,int forward)
{
    // 直到没有扫描到桥, 返回扫描变量即桥长
    int length=1;
    if(forward==H_ORIZONTAL)
    {

```

```

        for(length=1;map_display[i+length][j]==7;length++);
        return length;
    }
    if(forward==V_ERTICAL)
    {
        for(length=1;map_display[i][j+length]==7;length++);
        return length;
    }
}

/*****
*****
* 函数名称      Scan_Sub_Bridge_Crossing
* 函数作用      扫描郊区桥交叉
* 函数输入      i,j 地图相对坐标
* 函数输出      是返回1, 否返回0
*****
*****/

int Scan_Sub_Bridge_Crossing(int i,int j)
{
    int road_count=0;
    if(map_display[i+1][j]==3||map_display[i+1][j]==7)
    {
        road_count++;
    }
    if(map_display[i][j+1]==3||map_display[i][j+1]==7)
    {
        road_count++;
    }
    if(map_display[i-1][j]==3||map_display[i-1][j]==7)
    {
        road_count++;
    }
    if(map_display[i][j-1]==3||map_display[i][j-1]==7)
    {
        road_count++;
    }
    if(road_count==3||road_count==4)
    {
        return 1;
    }
    else

```

```

    {
        return 0;
    }
}

/*****
*****
* 函数名称      Field_Convert
* 函数作用      田地扫描
* 函数输入      x0,y0 左上角坐标, x, y 右下角坐标, mode 模式
* 函数输出      模式一返回是否为田地, 模式二返回写入是否成功
*****
*****/

int Field_Convert(int x0,int y0,int x,int y,int mode)
{
    static int x_l,y_l,x_r,y_r; //储存田地信息
    //mode 1 is read ,mode 2 is write
    if(mode==1)
    {
        if(x0>=x_l&&x0<=x_r&&y0>=y_l&&y0<=y_r)
        {
            return 1;
        }
        else return 0;
    }
    if(mode==2)
    {
        x_l=x0;
        y_l=y0;
        x_r=x;
        y_r=y;
        return 0;
    }
}

/*****
*****
* 函数名称      Mountainous_Bridge_Create
* 函数作用      山区桥生成
* 函数输入      x0,y0 地图生成坐标, i,j 地图相对坐标, draw_done 绘制
                完成数组
* 函数输出      无

```

```

*****
*****/

```

```

void Mountainous_Bridge_Create(int x0,int y0,int i,int j,int (*draw_done)[28])
{
    int length,count;
    length=Scan_Mountainous_Bridge(i,j);
    if(length==1)
    {
        Draw_Narrow_Bridge_Connection(x0+24*i,y0+24*j,H_ORIZONTAL);
    } // 单桥长直接绘制
    else
    {
        Draw_River_Icon(x0+24*i,y0+24*j);
        //Draw_River_Icon(x0+24*i+24,y0+24*j);
        Draw_Narrow_Bridge_End(x0+24*i,y0+24*j,R_IGHT); // 绘制桥
端点
        for(count=1;count<length-1;count++)
        {
            Draw_River_Icon(x0+24*(i+count),y0+24*j);
            //Draw_River_Icon(x0+24*(i+count),y0+24*j+24);
            Draw_Narrow_Bridge_Connection(x0+24*(i+count),y0+24*j,H_ORIZONTAL); // 根据桥长绘制桥中部
        }
        Draw_River_Icon(x0+24*(i+length-1),y0+24*j);
        //Draw_River_Icon(x0+24*(i+length-1),y0+24*j+24);
        Draw_Narrow_Bridge_End(x0+24*(i+length-1),y0+24*j,L_LEFT); // 绘制桥另一端点
        for(count=0;count<length;count++)
        {
            draw_done[i+count][j]=1;
        } // 记录绘制信息
    }
}

```

```

/*****
*****

```

```

* 函数名称      Scan_Mountainous_Bridge
* 函数作用      山区桥长扫描
* 函数输入      i,j 地图相对坐标
* 函数输出      桥长

```



```

*****
*****/

```

```

int Scan_Mountainous_Bridge(int i,int j)
{
    int length=1;
    for(length=1;map_display[i+length][j]==7;length++);
    return length;
}

```

```

/*****
*****

```

```

* 函数名称      Flag_Create
* 函数作用      点位绘制
* 函数输入      x, y 点位坐标数组, point 点位数量
* 函数输出      无

```

```

*****
*****/

```

```

void Flag_Create(int x[],int y[],int point)
{
    int i;
    for(i=0;i<point;i++)
    {
        Draw_Flag(63+24*x[i]+2,95+24*y[i]+2);
    }
}

```

8. find.c

```

#include"headfile.h"

```

```

/*****
*****

```

```

* @author      ytm
* @date        2022-4-19
* @content     PS-analysis algorithm

```

```

*****
*****/

```

```

/*****
*****

```

```

* 函数名称      Direction_Calculate
* 函数作用      随机生成参赛成员的行动路径
* 函数输入      people 人数, zone 区域类型, weather 天气类型, point

```

```

    点位数量, point_x 点位x 值地址, point_y 点位y 值地址
    *          person_basic 比赛成员基本信息结构体地址,
    person_competition 比赛成员参赛信息结构体地址
    *          direction 方向二维数组地址, your_number 用户参赛序
    号, flag_gui 过程可视化开关, flag_debug 调试模式开关
    * 函数注意      flag_debug 开启调试功能
    *          flag_gui 开启过程可视化
    * 函数输出      无
    *****
    *****/
void Direction_Calculate(int people,int zone,int weather,int poin
t,int* point_x,int* point_y,int **direction,people_basic *person_
basic,people_competition *person_competition,int your_number,int
flag_gui,int flag_debug)
{
    int i,j,t,x,y,h;
    int *flag=(int*)malloc(sizeof(int)*people);
    int *b=(int*)malloc(sizeof(int)*people);
    int addedpoint_x[4],addedpoint_y[4];
    FILE *fp;
    randomize();
    for(i=0;i<people;i++)
    {
        flag[i]=i+1;
        b[i]=random(100);
    }
    for(i=0;i<people;i++)
    {
        for(j=i+1;j<people;j++)
        {
            if(b[i]<b[j])
            {
                t = b[i];
                b[i] = b[j];
                b[j] = t;
                t = flag[i];
                flag[i] = flag[j];
                flag[j] = t;
            }
        }
    }
    //caculate standard way
    if(flag_gui)

```

```

{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Start Findpath0");
}
Findpath0(people,zone,weather,point,point_x,point_y,direction
[0],addedpoint_x,addedpoint_y,flag_gui,flag_debug);
for(i=0;i<people;i++)
{
    for(j=0;j<270;j++)
    {
        direction[i][j]=0;
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Start Findpath");
}
for(i=0;i<people;i++)
{
    if(flag_gui)
    {
        Solid_Bar(229,5,246,29,BLACK);
        putsz(229,5,24,16,RED,flag[i]);
    }
    if(i!=your_number)
    {
        Findpath(flag[i],point,point_x,point_y,direction[i],a
ddedpoint_x,addedpoint_y,flag_gui,flag_debug);
    }
}
//analysis direction and fix bugs
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Start Analysis");
}
switch(zone)
{
case 3:
    {
        for(i=0;i<people;i++)
        {

```

```

if(i!=your_number)
{
    //caculate ps and action
    t=2*person_basic[i].ps_consume/person_basic[i
].ps_recover;

    x=0,y=0,j=0,h=0;
    if(map_display[0][0]>10)
    {
        while(direction[i][j]!=0)
        {
            if(direction[i][j]==1)
            {
                y--;
            }
            else if(direction[i][j]==2)
            {
                y++;
            }
            else if(direction[i][j]==3)
            {
                x--;
            }
            else if(direction[i][j]==4)
            {
                x++;
            }
            else if(direction[i][j]==5)
            {
                x--;
                y--;
            }
            else if(direction[i][j]==6)
            {
                x--;
                y++;
            }
            else if(direction[i][j]==7)
            {
                x++;
                y--;
            }
            else if(direction[i][j]==8)
            {

```

```

        x++;
        y++;
    }
    if(map_display[x][y]<10)
    {
        break;
    }
    direction[i][j]+=30;
    j++;
}
while(direction[i][j]!=0)
{
    if(direction[i][j]==1)
    {
        y--;
    }
    else if(direction[i][j]==2)
    {
        y++;
    }
    else if(direction[i][j]==3)
    {
        x--;
    }
    else if(direction[i][j]==4)
    {
        x++;
    }
    else if(direction[i][j]==5)
    {
        x--;
        y--;
    }
    else if(direction[i][j]==6)
    {
        x--;
        y++;
    }
    else if(direction[i][j]==7)
    {
        x++;
        y--;
    }
}

```

```

        else if(direction[i][j]==8)
        {
            x++;
            y++;
        }
        if(h%t<(t/2))
        {
            direction[i][j]+=20;
        }
        else
        {
            direction[i][j]+=10;
        }
        j++,h++;
    }
}
else
{
    //caculate ps and action
    while(direction[i][j]!=0)
    {
        if(direction[i][j]==1)
        {
            y--;
        }
        else if(direction[i][j]==2)
        {
            y++;
        }
        else if(direction[i][j]==3)
        {
            x--;
        }
        else if(direction[i][j]==4)
        {
            x++;
        }
        else if(direction[i][j]==5)
        {
            x--;
            y--;
        }
        else if(direction[i][j]==6)

```

```

{
    x--;
    y++;
}
else if(direction[i][j]==7)
{
    x++;
    y--;
}
else if(direction[i][j]==8)
{
    x++;
    y++;
}
if(map_display[x][y]>10)
{
    break;
}
if(j%t<(t/2))
{
    direction[i][j]+=20;
}
else
{
    direction[i][j]+=10;
}
j++;
}
while(direction[i][j]!=0)
{
    if(direction[i][j]==1)
    {
        y--;
    }
    else if(direction[i][j]==2)
    {
        y++;
    }
    else if(direction[i][j]==3)
    {
        x--;
    }
    else if(direction[i][j]==4)

```

```

        {
            x++;
        }
        else if(direction[i][j]==5)
        {
            x--;
            y--;
        }
        else if(direction[i][j]==6)
        {
            x--;
            y++;
        }
        else if(direction[i][j]==7)
        {
            x++;
            y--;
        }
        else if(direction[i][j]==8)
        {
            x++;
            y++;
        }
        direction[i][j]+=30;
        j++;
    }
    Direction_Optimize(zone,direction[i],x,y,t,h,
j));
    }
}
break;
default:
{
    for(i=0;i<people;i++)
    {
        if(i!=your_number)
        {
            //caculate ps and action
            t=2*person_basic[i].ps_consume/person_bas
ic[i].ps_recover;
            x=0,y=0,j=0,h=0;

```



```

while(direction[i][j]!=0)
{
    if(direction[i][j]==1)
    {
        y--;
    }
    else if(direction[i][j]==2)
    {
        y++;
    }
    else if(direction[i][j]==3)
    {
        x--;
    }
    else if(direction[i][j]==4)
    {
        x++;
    }
    else if(direction[i][j]==5)
    {
        x--;
        y--;
    }
    else if(direction[i][j]==6)
    {
        x--;
        y++;
    }
    else if(direction[i][j]==7)
    {
        x++;
        y--;
    }
    else if(direction[i][j]==8)
    {
        x++;
        y++;
    }
    if(j%t<(t/2))
    {
        direction[i][j]+=20;
    }
    else

```

```

        {
            direction[i][j]+=10;
        }
        j++;
    }
    Direction_Optimize(zone,direction[i],x,y,
t,h,j);
    }
}
}
break;
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\DIRECT.txt","w+");
    fprintf(fp,"Direction Information\n");
    for(i=0;i<people;i++)
    {
        if(i!=your_number)
        {
            fprintf(fp,"\nnum %d or Findpath%d 's Direction I
nformation for value:\n",i+1,flag[i]);
            j=0;
            while(direction[i][j]!=0)
            {
                fprintf(fp,"direction[%d]=%d;\n",j,direction[
i][j]);
                j++;
            }
        }
    }
    fclose(fp);
}
//delete malloc
free(flag);
free(b);
}

```

```

/*****
*****

```

```

* 函数名称      Direction_Optimize
* 函数作用      优化寻路算法的方向

```

* 函数输入 *zone* 区域类型, *direction* 方向一维数组地址, *x*, *y* 目前位置, *t*, *h*, *j* 分析变量

* 函数输出 无

 *****/

```
void Direction_Optimize(int zone,int* direction,int x,int y,int t
,int h,int j)
```

```
{
    if(x!=39||y!=27)
    {
        switch(zone)
        {
            case 3:
            {
                if(map_display[0][0]>10)
                {
                    if(x==39&&y==26)
                    {
                        direction[j]=2;
                        if(h%t<(t/2))
                        {
                            direction[j]+=20;
                        }
                        else
                        {
                            direction[j]+=10;
                        }
                    }
                }
                else if(x==38&&y==27)
                {
                    direction[j]=4;
                    if(h%t<(t/2))
                    {
                        direction[j]+=20;
                    }
                    else
                    {
                        direction[j]+=10;
                    }
                }
                else if(x==38&&y==26)
                {
                    direction[j]=8;
```

```

        if(h%t<(t/2))
        {
            direction[j]+=20;
        }
        else
        {
            direction[j]+=10;
        }
    }
else if(x==37&&y==27)
{
    direction[j]=4;
    if(h%t<(t/2))
    {
        direction[j]+=20;
    }
    else
    {
        direction[j]+=10;
    }
    direction[j+1]=4;
    if((h+1)%t<(t/2))
    {
        direction[j+1]+=20;
    }
    else
    {
        direction[j+1]+=10;
    }
}
else if(x==39&&y==25)
{
    direction[j]=2;
    if(h%t<(t/2))
    {
        direction[j]+=20;
    }
    else
    {
        direction[j]+=10;
    }
    direction[j+1]=2;
    if((h+1)%t<(t/2))

```

```

        {
            direction[j+1]+=20;
        }
        else
        {
            direction[j+1]+=10;
        }
    }
}
else
{
    if(x!=39||y!=27)
    {
        if(x==39&&y==26)
        {
            direction[j]=2;
            direction[j]+=30;
        }
        else if(x==38&&y==27)
        {
            direction[j]=4;
            direction[j]+=30;
        }
        else if(x==38&&y==26)
        {
            direction[j]=8;
            direction[j]+=30;
        }
        else if(x==37&&y==27)
        {
            direction[j]=4;
            direction[j]+=30;
            direction[j+1]=4;
            direction[j+1]+=30;
        }
        else if(x==39&&y==25)
        {
            direction[j]=2;
            direction[j]+=30;
            direction[j+1]=2;
            direction[j+1]+=30;
        }
    }
}

```

```

    }
}
break;
default:
{
    if(x!=39||y!=27)
    {
        if(x==39&&y==26)
        {
            direction[j]=2;
            if(j%t<(t/2))
            {
                direction[j]+=20;
            }
            else
            {
                direction[j]+=10;
            }
        }
        else if(x==38&&y==27)
        {
            direction[j]=4;
            if(j%t<(t/2))
            {
                direction[j]+=20;
            }
            else
            {
                direction[j]+=10;
            }
        }
        else if(x==38&&y==26)
        {
            direction[j]=8;
            if(j%t<(t/2))
            {
                direction[j]+=20;
            }
            else
            {
                direction[j]+=10;
            }
        }
    }
}

```

```

else if(x==37&&y==27)
{
    direction[j]=4;
    if(j%t<(t/2))
    {
        direction[j]+=20;
    }
    else
    {
        direction[j]+=10;
    }
    direction[j+1]=4;
    if((j+1)%t<(t/2))
    {
        direction[j+1]+=20;
    }
    else
    {
        direction[j+1]+=10;
    }
}
else if(x==39&&y==25)
{
    direction[j]=2;
    if(j%t<(t/2))
    {
        direction[j]+=20;
    }
    else
    {
        direction[j]+=10;
    }
    direction[j+1]=2;
    if((j+1)%t<(t/2))
    {
        direction[j+1]+=20;
    }
    else
    {
        direction[j+1]+=10;
    }
}
}

```

```

        }
        break;
    }
}

/*****
*****
* 函数名称      Findpath
* 函数作用      随机生成参赛成员的行动路径
* 函数输入      flag 调用随机路径的序号, point 点位数量, point_x 点
位x 值地址, point_y 点位y 值地址, direction 方向一维数组地址
*              addedpoint_x 加入点位x 值地址, addedpoint_y 加入点
位y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数注意      flag_debug 开启调试功能
*              flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Findpath(int flag,int point,int* point_x,int* point_y,int *d
irection,int *addedpoint_x,int *addedpoint_y,int flag_gui,int fla
g_debug)
{
    switch(flag)
    {
        case 1:
        {
            Findpath1(point,point_x,point_y,direction,addedpo
int_x,addedpoint_y,flag_gui,flag_debug);
            break;
        }
        case 2:
        {
            Findpath2(point,point_x,point_y,direction,addedpo
int_x,addedpoint_y,flag_gui,flag_debug);
            break;
        }
        case 3:
        {
            Findpath3(point,point_x,point_y,direction,addedpo
int_x,addedpoint_y,flag_gui,flag_debug);
            break;
        }
    }
}

```



```

        case 4:
        {
            Findpath4(point,point_x,point_y,direction,addedpoint_x,addedpoint_y,flag_gui,flag_debug);
            break;
        }
        case 5:
        {
            Findpath5(point,point_x,point_y,direction,addedpoint_x,addedpoint_y,flag_gui,flag_debug);
            break;
        }
        case 6:
        {
            Findpath6(point,point_x,point_y,direction,addedpoint_x,addedpoint_y,flag_gui,flag_debug);
            break;
        }
    }
}

```

```

/*****
*****/

```

```

* 函数名称      Findpath1
* 函数作用      生成特殊路径1
* 函数输入      point 点位数量, point_x 点位x 值地址, point_y 点位y
                  值地址, direction 方向一维数组地址
*               addedpoint_x 加入点位x 值地址, addedpoint_y 加入点
                  位y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*               flag_gui 开启过程可视化
* 函数输出      无

```

```

*****/

```

```

void Findpath1(int point,int* point_x,int* point_y,int *direction
,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step=0;
    int j,x=0,y=0,way[40][28];
    int odd;
    FILE *fp;
    //init
    randomize();

```

```

odd=random(2);
//caculate way 1
if(odd)
{
    starterX=point_x[0];
    starterY=point_y[0];
    destinationX=addedpoint_x[1];
    destinationY=addedpoint_y[1];
    Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    starterX=destinationX;
    starterY=destinationY;
    destinationX=point_x[1];
    destinationY=point_y[1];
    Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    for(i=0;i<point-2;i++)
    {
        starterX=point_x[i+1];
        starterY=point_y[i+1];
        destinationX=point_x[i+2];
        destinationY=point_y[i+2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    }
}
else
{
    for(i=0;i<point-2;i++)
    {
        starterX=point_x[i];
        starterY=point_y[i];
        destinationX=point_x[i+1];
        destinationY=point_y[i+1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    }
    if(point==2)
    {
        starterX=point_x[0];
        starterY=point_y[0];
    }
    else

```

```

    {
        starterX=destinationX;
        starterY=destinationY;
    }
    destinationX=addedpoint_x[3];
    destinationY=addedpoint_y[3];
    Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    starterX=destinationX;
    starterY=destinationY;
    destinationX=point_x[point-1];
    destinationY=point_y[point-1];
    Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\WAY1.txt","w+");
    fprintf(fp,"\nWay 1 Information\n");
    fprintf(fp,"\nReference\n1:up 2:down 3:left 4:right\n5:left-up 6:left-down 7:right-up 8:right-down\n");
    //generate way[40][28]: get 0 walkable, 1 obscale, 8 way, 9 point
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            way[i][j]=map_process[i][j];
        }
    }
    //get 8 way and 9 point
    i=0;
    while(direction[i]!=0)
    {
        fprintf(fp,"direction[%d] = %d\n",i,direction[i]);
        if(direction[i]==1)
        {
            y--;
        }
        else if(direction[i]==2)
        {
            y++;
        }
    }
}

```

```

    }
    else if(direction[i]==3)
    {
        x--;
    }
    else if(direction[i]==4)
    {
        x++;
    }
    else if(direction[i]==5)
    {
        x--;
        y--;
    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)
    {
        x++;
        y++;
    }
    way[x][y]=8;
    i++;
}
for(i=0;i<4;i++)
{
    way[point_x[i]][point_y[i]]=9;
}
fprintf(fp,"\nReference\n0:walkable  1:obstacle  8:way  9
:point\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)

```

```

        {
            fprintf(fp," %d \n",way[i][j]);
        }
        else
        {
            fprintf(fp," %d ",way[i][j]);
        }
    }
}
fclose(fp);
}
}

/*****
*****
* 函数名称      Findpath2
* 函数作用      生成特殊路径 2
* 函数输入      point 点位数量, point_x 点位 x 值地址, point_y 点位 y
值地址, direction 方向一维数组地址
*               addedpoint_x 加入点位 x 值地址, addedpoint_y 加入点
位 y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*               flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Findpath2(int point,int* point_x,int* point_y,int *direction
,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step=0;
    int j,x=0,y=0,way[40][28];
    int odd;
    FILE *fp;
    //init
    randomize();
    odd=random(2);
    //caculate way 2
    if(odd)
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
    }
}

```

```

        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        for(i=0;i<point-2;i++)
        {
            starterX=point_x[i+1];
            starterY=point_y[i+1];
            destinationX=point_x[i+2];
            destinationY=point_y[i+2];
            Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        }
    }
    else
    {
        for(i=0;i<point-2;i++)
        {
            starterX=point_x[i];
            starterY=point_y[i];
            destinationX=point_x[i+1];
            destinationY=point_y[i+1];
            Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        }
        if(point==2)
        {
            starterX=point_x[0];
            starterY=point_y[0];
        }
        else
        {
            starterX=destinationX;
            starterY=destinationY;
        }
        destinationX=addedpoint_x[2];
        destinationY=addedpoint_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    }
}

```

```

        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[point-1];
        destinationY=point_y[point-1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
    }
    //debug output
    if(flag_debug)
    {
        fp = fopen (".\\WAY\\WAY2.txt","w+");
        fprintf(fp,"\nWay 2 Information\n");
        fprintf(fp,"\nReference\n1:up 2:down 3:left 4:right\n5:left-up 6:left-down 7:right-up 8:right-down\n");
        //generate way[40][28]: get 0 walkable, 1 obscale, 8 way, 9 point
        for(j=0;j<28;j++)
        {
            for(i=0;i<40;i++)
            {
                way[i][j]=map_process[i][j];
            }
        }
        //get 8 way and 9 point
        i=0;
        while(direction[i]!=0)
        {
            fprintf(fp,"direction[%d] = %d\n",i,direction[i]);
            if(direction[i]==1)
            {
                y--;
            }
            else if(direction[i]==2)
            {
                y++;
            }
            else if(direction[i]==3)
            {
                x--;
            }
            else if(direction[i]==4)
            {
                x++;
            }
        }
    }

```

```

    }
    else if(direction[i]==5)
    {
        x--;
        y--;
    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)
    {
        x++;
        y++;
    }
    way[x][y]=8;
    i++;
}
for(i=0;i<4;i++)
{
    way[point_x[i]][point_y[i]]=9;
}
fprintf(fp,"\nReference\n0:walkable  1:obstacle  8:way  9
:point\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)
        {
            fprintf(fp," %d \n",way[i][j]);
        }
        else
        {
            fprintf(fp," %d ",way[i][j]);
        }
    }
}

```



```

    }
    fclose(fp);
}
}

/*****
*****
* 函数名称      Findpath3
* 函数作用      生成特殊路径3
* 函数输入      point 点位数量, point_x 点位x 值地址, point_y 点位y
                  值地址, direction 方向一维数组地址
*                addedpoint_x 加入点位x 值地址, addedpoint_y 加入点
                  位y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*                flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Findpath3(int point,int* point_x,int* point_y,int *direction
,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step=0;
    int j,x=0,y=0,way[40][28];
    int odd;
    FILE *fp;
    //init
    randomize();
    odd=random(6);
    //caculate way 3
    if(odd)
    {
        switch (point)
        {
            case 2:
            {
                starterX=point_x[0];
                starterY=point_y[0];
                destinationX=addedpoint_x[1];
                destinationY=addedpoint_y[1];
                Searchpath(starterX,starterY,destinationX,destina
tionY,direction,&step,flag_gui);
                starterX=destinationX;
                starterY=destinationY;
            }
        }
    }
}

```

```

        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
    case 3:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
    case 4:
    {
        starterX=point_x[0];

```

```

        starterY=point_y[0];
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[3];
        destinationY=point_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
}
else //shortest way!
{
    for(i=0;i<(point-1);i++)
    {
        starterX=point_x[i];
        starterY=point_y[i];
        destinationX=point_x[i+1];
        destinationY=point_y[i+1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);

```

```

    }
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\WAY3.txt","w+");
    fprintf(fp,"\\nWay 3 Information\\n");
    fprintf(fp,"\\nReference\\n1:up 2:down 3:left 4:right\\n5
:left-up 6:left-down 7:right-up 8:right-down\\n");
    //generate way[40][28]: get 0 walkable, 1 obscale, 8 way,
9 point
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            way[i][j]=map_process[i][j];
        }
    }
    //get 8 way and 9 point
    i=0;
    while(direction[i]!=0)
    {
        fprintf(fp,"direction[%d] = %d\\n",i,direction[i]);
        if(direction[i]==1)
        {
            y--;
        }
        else if(direction[i]==2)
        {
            y++;
        }
        else if(direction[i]==3)
        {
            x--;
        }
        else if(direction[i]==4)
        {
            x++;
        }
        else if(direction[i]==5)
        {
            x--;
            y--;
        }
    }
}

```

```

    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)
    {
        x++;
        y++;
    }
    way[x][y]=8;
    i++;
}
for(i=0;i<4;i++)
{
    way[point_x[i]][point_y[i]]=9;
}
fprintf(fp,"\nReference\n0:walkable  1:obstacle  8:way  9
:point\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)
        {
            fprintf(fp," %d \n",way[i][j]);
        }
        else
        {
            fprintf(fp," %d ",way[i][j]);
        }
    }
}
fclose(fp);
}
}

```

```

/*****
*****

* 函数名称      Findpath4
* 函数作用      生成特殊路径4
* 函数输入      point 点位数量, point_x 点位x 值地址, point_y 点位y
值地址, direction 方向一维数组地址
*              addedpoint_x 加入点位x 值地址, addedpoint_y 加入点
位y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*              flag_gui 开启过程可视化
* 函数输出      无
*****
*****/

void Findpath4(int point,int* point_x,int* point_y,int *direction
,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step=0;
    int j,x=0,y=0,way[40][28];
    int odd;
    FILE *fp;
    //init
    randomize();
    odd=random(6);
    //caculate way 4
    if(odd)
    {
        switch (point)
        {
            case 2:
            {
                starterX=point_x[0];
                starterY=point_y[0];
                destinationX=addedpoint_x[0];
                destinationY=addedpoint_y[0];
                Searchpath(starterX,starterY,destinationX,destina
tionY,direction,&step,flag_gui);
                starterX=destinationX;
                starterY=destinationY;
                destinationX=addedpoint_x[2];
                destinationY=addedpoint_y[2];
                Searchpath(starterX,starterY,destinationX,destina
tionY,direction,&step,flag_gui);
                starterX=destinationX;

```

```

        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
    case 3:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[2];
        destinationY=addedpoint_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
    case 4:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);

```

```

        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[2];
        destinationY=addedpoint_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[3];
        destinationY=point_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
}
else //Longest way!
{
    switch (point)
    {
        case 2:
        {
            starterX=point_x[0];
            starterY=point_y[0];
            destinationX=addedpoint_x[0];
            destinationY=addedpoint_y[0];
            Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
            starterX=destinationX;
            starterY=destinationY;
            destinationX=addedpoint_x[1];

```



```

        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
    case 3:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);

```

```

        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
    case 4:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[3];

```

```

        destinationY=point_y[3];
        Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
        break;
    }
}
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\WAY4.txt","w+");
    fprintf(fp,"\nWay 4 Information\n");
    fprintf(fp,"\nReference\n1:up 2:down 3:left 4:right\n5:left-up 6:left-down 7:right-up 8:right-down\n");
    //generate way[40][28]: get 0 walkable, 1 obscale, 8 way, 9 point
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            way[i][j]=map_process[i][j];
        }
    }
    //get 8 way and 9 point
    i=0;
    while(direction[i]!=0)
    {
        fprintf(fp,"direction[%d] = %d\n",i,direction[i]);
        if(direction[i]==1)
        {
            y--;
        }
        else if(direction[i]==2)
        {
            y++;
        }
        else if(direction[i]==3)
        {
            x--;
        }
        else if(direction[i]==4)
        {
            x++;
        }
    }
}

```

```

    }
    else if(direction[i]==5)
    {
        x--;
        y--;
    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)
    {
        x++;
        y++;
    }
    way[x][y]=8;
    i++;
}
for(i=0;i<4;i++)
{
    way[point_x[i]][point_y[i]]=9;
}
fprintf(fp,"\nReference\n0:walkable  1:obstacle  8:way  9
:point\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)
        {
            fprintf(fp," %d \n",way[i][j]);
        }
        else
        {
            fprintf(fp," %d ",way[i][j]);
        }
    }
}

```

```

    }
    fclose(fp);
}
}

/*****
*****
* 函数名称      Findpath5
* 函数作用      生成特殊路径5
* 函数输入      point 点位数量, point_x 点位x 值地址, point_y 点位y
                  值地址, direction 方向一维数组地址
*                  addedpoint_x 加入点位x 值地址, addedpoint_y 加入点
                  位y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*                  flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Findpath5(int point,int* point_x,int* point_y,int *direction
,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step=0;
    int j,x=0,y=0,way[40][28];
    FILE *fp;
    //caculate way 5
    switch (point)
    {
    case 2:
        {
            starterX=point_x[0];
            starterY=point_y[0];
            destinationX=addedpoint_x[0];
            destinationY=addedpoint_y[0];
            Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
            starterX=destinationX;
            starterY=destinationY;
            destinationX=addedpoint_x[3];
            destinationY=addedpoint_y[3];
            Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
            starterX=destinationX;
            starterY=destinationY;

```

```

        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        break;
    }
    case 3:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        break;
    }
    case 4:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[0];
        destinationY=addedpoint_y[0];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;

```

```

        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[3];
        destinationY=addedpoint_y[3];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[3];
        destinationY=point_y[3];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        break;
    }
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\WAY5.txt","w+");
    fprintf(fp,"\\nWay 5 Information\\n");
    fprintf(fp,"\\nReference\\n1:up 2:down 3:left 4:right\\n5
:left-up 6:left-down 7:right-up 8:right-down\\n");
    //generate way[40][28]: get 0 walkable, 1 obscale, 8 way,
9 point
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            way[i][j]=map_process[i][j];
        }
    }
    //get 8 way and 9 point

```

```

i=0;
while(direction[i]!=0)
{
    fprintf(fp,"direction[%d] = %d\n",i,direction[i]);
    if(direction[i]==1)
    {
        y--;
    }
    else if(direction[i]==2)
    {
        y++;
    }
    else if(direction[i]==3)
    {
        x--;
    }
    else if(direction[i]==4)
    {
        x++;
    }
    else if(direction[i]==5)
    {
        x--;
        y--;
    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)
    {
        x++;
        y++;
    }
    way[x][y]=8;
    i++;
}

```



```

        for(i=0;i<4;i++)
        {
            way[point_x[i]][point_y[i]]=9;
        }
        fprintf(fp,"\nReference\n0:walkable  1:obstacle  8:way  9
:point\n");
        for(j=0;j<28;j++)
        {
            for(i=0;i<40;i++)
            {
                if(i==39)
                {
                    fprintf(fp," %d \n",way[i][j]);
                }
                else
                {
                    fprintf(fp," %d ",way[i][j]);
                }
            }
        }
        fclose(fp);
    }
}

```

```

/*****
*****

```

```

* 函数名称      Findpath6
* 函数作用      生成特殊路径6
* 函数输入      point 点位数量, point_x 点位x 值地址, point_y 点位y
值地址, direction 方向一维数组地址
*              addedpoint_x 加入点位x 值地址, addedpoint_y 加入点
位y 值地址, flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*              flag_gui 开启过程可视化
* 函数输出      无

```

```

*****
*****/

```

```

void Findpath6(int point,int* point_x,int* point_y,int *direction
,int *addedpoint_x,int *addedpoint_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step=0;
    int j,x=0,y=0,way[40][28];
    FILE *fp;

```

```

//caculate way 6
switch (point)
{
case 2:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[2];
        destinationY=addedpoint_y[2];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        break;
    }
case 3:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[2];
        destinationY=addedpoint_y[2];
    }
}

```

```

        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        break;
    }
    case 4:
    {
        starterX=point_x[0];
        starterY=point_y[0];
        destinationX=addedpoint_x[1];
        destinationY=addedpoint_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[1];
        destinationY=point_y[1];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[2];
        destinationY=point_y[2];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=addedpoint_x[2];
        destinationY=addedpoint_y[2];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        starterX=destinationX;
        starterY=destinationY;
        destinationX=point_x[3];
        destinationY=point_y[3];
        Searchpath(starterX,starterY,destinationX,destination
Y,direction,&step,flag_gui);
        break;
    }

```

```

    }
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\WAY6.txt","w+");
    fprintf(fp,"\\nWay 6 Information\\n");
    fprintf(fp,"\\nReference\\n1:up 2:down 3:left 4:right\\n5
:left-up 6:left-down 7:right-up 8:right-down\\n");
    //generate way[40][28]: get 0 walkable, 1 obscale, 8 way,
9 point
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            way[i][j]=map_process[i][j];
        }
    }
    //get 8 way and 9 point
    i=0;
    while(direction[i]!=0)
    {
        fprintf(fp,"direction[%d] = %d\\n",i,direction[i]);
        if(direction[i]==1)
        {
            y--;
        }
        else if(direction[i]==2)
        {
            y++;
        }
        else if(direction[i]==3)
        {
            x--;
        }
        else if(direction[i]==4)
        {
            x++;
        }
        else if(direction[i]==5)
        {
            x--;
            y--;
        }
    }
}

```

```

    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)
    {
        x++;
        y++;
    }
    way[x][y]=8;
    i++;
}
for(i=0;i<4;i++)
{
    way[point_x[i]][point_y[i]]=9;
}
fprintf(fp,"\nReference\n0:walkable  1:obstacle  8:way  9
:point\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)
        {
            fprintf(fp," %d \n",way[i][j]);
        }
        else
        {
            fprintf(fp," %d ",way[i][j]);
        }
    }
}
fclose(fp);
}
}

```

9. findl.c

```

#include "headfile.h"

/*****
*****
* @edictor          ytm
* @date             2022-4-19
* @content          A* way-finding algorithm
*                  Breadth-first search(BFS) algorithm
*                  Added-point generate algorithm 4.2
*****
*****/

Lstack Open = NULL;
Lstack Closed = NULL;

/*****
*****
* 函数名称          getFminNodeFromOpen
* 函数作用          选取 OPEN 表上  $f$  值(总代价) 相同  $g$  值最小的节点
* 函数输入          无
* 函数输出          该节点首地址
*****
*****/

Lnode getFminNodeFromOpen()
{
    Lstack temp = Open->next, min = Open->next, minp = Open;
    Lnode minx;
    if( temp == NULL )
    {
        return NULL;
    }
    while(temp->next!=NULL)
    {
        if((temp->next->npoint->f)<(min->npoint->f))
        {
            min = temp->next;
            minp = temp;
        }
        else if(((temp->next->npoint->f)==(min->npoint->f))&&(temp->next->npoint->g)<(min->npoint->g))
        {
            min = temp->next;
            minp = temp;
        }
    }
}

```

```

        }
        temp = temp->next;
    }
    minx = min->npoint;
    temp = minp->next;
    minp->next = minp->next->next;
    free(temp);
    return minx;
}

/*****
*****
* 函数名称      getGminNodeFromOpen
* 函数作用      选取 OPEN 表上 g 值最小的节点
* 函数输入      无
* 函数输出      该节点首地址
*****
*****/
Lnode getGminNodeFromOpen()
{
    Lstack temp = Open->next,min = Open->next,minp = Open;
    Lnode minx;
    if( temp == NULL )
    {
        return NULL;
    }
    while(temp->next!=NULL)
    {
        if((temp->next->npoint->g)<(min->npoint->g))
        {
            min = temp->next;
            minp = temp;
        }
        temp = temp->next;
    }
    minx = min->npoint;
    temp = minp->next;
    minp->next = minp->next->next;
    free(temp);
    return minx;
}

/*****
*****

```

```

*****
* 函数名称      BelongInOpen
* 函数作用      判断节点是否属于Open 表
* 函数输入      X, Y 位置坐标
* 函数输出      是则返回节点地址, 否则返回空地址
*****
*****/
Lnode BelongInOpen(int X, int Y)
{
    Lstack temp = Open -> next;
    if ( temp == NULL )
    {
        return NULL;
    }
    while (temp != NULL)
    {
        if((temp->npoint->X == X)&&(temp->npoint->Y==Y))
        {
            return temp -> npoint;
        }
        else
        {
            temp = temp->next;
        }
    }
    return NULL;
}

/*****
*****
* 函数名称      BelongInClosed
* 函数作用      判断节点是否属于Closed 表
* 函数输入      X, Y 位置坐标
* 函数输出      是则返回 TURE, 否则返回 FALSE
*****
*****/
BOOL BelongInClosed(int X, int Y)
{
    Lstack temp = Closed -> next;
    if ( temp == NULL )
    {
        return FALSE;
    }
}

```



```

while (temp != NULL)
{
    if ((temp->npoint->X == X)&&(temp->npoint->Y==Y))
    {
        return TRUE;
    }
    else
    {
        temp = temp->next;
    }
}
return FALSE;
}

/*****
*****
* 函数名称      PutintoOpen
* 函数作用      把节点放入 OPEN 表中
* 函数输入      node 节点地址
* 函数输出      无
*****
*****/
void PutintoOpen(Lnode node)
{
    Lstack temp;
    temp =(Lstack)malloc(sizeof(Stack));
    temp->npoint = node;
    temp->next = Open->next;
    Open->next = temp;
}

/*****
*****
* 函数名称      PutintoClosed
* 函数作用      把节点放入 Closed 表中
* 函数输入      node 节点地址
* 函数输出      无
*****
*****/
void PutintoClosed(Lnode node)
{
    Lstack temp;
    temp =(Lstack)malloc(sizeof(Stack));

```

```

    temp->npoint = node;
    temp->next = Closed->next;
    Closed->next = temp;
}

/*****
*****
* 函数名称      getH
* 函数作用      得到该节点的h 值(曼哈顿距离)
* 函数输入      X, Y 位置坐标, destinationX, destinationY 终点坐标
* 函数输出      h 值(曼哈顿距离)
*****
*****/
int getH(int X,int Y,int destinationX,int destinationY)
{
    return (10*(abs(destinationX - X)+abs(destinationY - Y)));
}

/*****
*****
* 函数名称      isCanMove
* 函数作用      检测该节点是否可通行或存在
* 函数输入      X, Y 位置坐标
* 函数输出      是则返回 TRUE, 否则返回 FALSE
*****
*****/
BOOL isCanMove(int X,int Y)
{
    if(X>=0&&X<40&&Y>=0&&Y<28&&map_process[X][Y]==0)
        return TRUE;
    else
        return FALSE;
}

/*****
*****
* 函数名称      creatSeccessionNode
* 函数作用      根据父节点生成子节点并传入 Open 表(边界列表) 中
* 函数输入      parentNode 父节点地址, X, Y 位置坐标,
destinationX, destinationY 终点坐标, gPlus 节点g 值增量
* 函数输出      无
*****
*****/

```

```

void creatSeccessionNode(Lnode parentNode,int X,int Y,int destinationX, int destinationY,int gPlus)
{
    Lnode Lnode1 = NULL;
    int g = parentNode->g + gPlus;
    /*gPlus: G value addition*/
    if(!BelongInClosed(X,Y))
    /*in Closed stack,desert its information*/
    {
        if((Lnode1 = BelongInOpen(X,Y)) != NULL)
            /*in Open stack(have scanned):caculate again and cover original information*/
            {
                if(Lnode1->g < g)
                {
                    Lnode1->parent = parentNode;
                    Lnode1->g = g;
                    Lnode1->f = g + Lnode1->h;
                }
            }
        else
            /*first scan: caculate its information*/
            {
                Lnode1=(Lnode)malloc(sizeof(Node));
                Lnode1->parent = parentNode;
                Lnode1->g = g;
                Lnode1->h = getH(X,Y,destinationX,destinationY);
                Lnode1->f = Lnode1->g + Lnode1->h;
                Lnode1->X = X;
                Lnode1->Y = Y;
                PutintoOpen(Lnode1);
            }
    }
}

```

```

/*****
*****
* 函数名称      seachSeccessionNode
* 函数作用      根据传入的父节点生成所有子节点
* 函数输入      parentNode 父节点地址, destinationX, destinationY
                  终点坐标
* 函数输出      无
*****

```

```

*****/
void searchSeccessionNode(Lnode parentNode, int destinationX, int
destinationY)
{
    int X,Y;
    //up
    if(isCanMove(X = parentNode->X, Y = parentNode->Y - 1))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, desti
nationY,10);
    }
    //down
    if(isCanMove( X = parentNode->X,Y = parentNode->Y + 1))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, desti
nationY,10);
    }
    //left
    if(isCanMove(X = parentNode->X - 1, Y = parentNode->Y))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, desti
nationY,10);
    }
    //right
    if(isCanMove(X = parentNode->X + 1, Y = parentNode->Y))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, desti
nationY,10);
    }
    //left-up
    if(isCanMove(X = parentNode->X - 1, Y = parentNode->Y - 1))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, desti
nationY,14);
    }
    //left-down
    if(isCanMove(X = parentNode->X - 1, Y = parentNode->Y + 1))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, desti
nationY,14);
    }
    //right-up
    if(isCanMove(X = parentNode->X + 1, Y = parentNode->Y - 1))

```

```

    {
        creatSeccessionNode(parentNode, X, Y, destinationX, destinationY,14);
    }
    //right-down
    if(isCanMove(X = parentNode->X + 1, Y = parentNode->Y + 1))
    {
        creatSeccessionNode(parentNode, X, Y, destinationX, destinationY,14);
    }
    PutintoClosed(parentNode);
}

```

```

/*****
*****

```

```

* 函数名称      creatSeccessionNodeG
* 函数作用      根据父节点生成子节点并传入 Open 表(边界列表)中
* 函数输入      parentNode 父节点地址, X, Y 位置坐标, gPlus 节点g 值增量
* 函数注意      仅仅计算g 值
* 函数输出      无

```

```

*****
*****/

```

```

void creatSeccessionNodeG(Lnode parentNode,int X,int Y,int gPlus)
{
    Lnode Lnode1 = NULL;
    int g = parentNode->g + gPlus;
    /*gPlus: G value addition*/
    if(!BelongInClosed(X,Y))
    /*in Closed stack,desert its information*/
    {
        if((Lnode1 = BelongInOpen(X,Y)) != NULL)
        /*in Open stack(have scanned):caculate again and cover original information*/
        {
            if(Lnode1->g < g)
            {
                Lnode1->parent = parentNode;
                Lnode1->g = g;
            }
        }
        else
        /*first scan: caculate its information*/

```

```

        {
            Lnode1=(Lnode)malloc(sizeof(Node));
            Lnode1->parent = parentNode;
            Lnode1->g = g;
            Lnode1->X = X;
            Lnode1->Y = Y;
            PutintoOpen(Lnode1);
        }
    }
}

/*****
*****
* 函数名称      seachSeccessionNodeG
* 函数作用      根据传入的父节点生成所有子节点
* 函数输入      parentNode 父节点地址
* 函数注意      仅仅计算g 值
* 函数输出      无
*****
*****/
void seachSeccessionNodeG(Lnode parentNode)
{
    int X,Y;
    //up
    if(isCanMove( X = parentNode->X, Y = parentNode->Y - 1))
    {
        creatSeccessionNodeG(parentNode, X, Y,10);
    }
    //down
    if(isCanMove(X = parentNode->X,Y = parentNode->Y + 1))
    {
        creatSeccessionNodeG(parentNode, X, Y,10);
    }
    //left
    if(isCanMove(X = parentNode->X - 1, Y = parentNode->Y))
    {
        creatSeccessionNodeG(parentNode, X, Y,10);
    }
    //right
    if(isCanMove(X = parentNode->X + 1, Y = parentNode->Y))
    {
        creatSeccessionNodeG(parentNode, X, Y,10);
    }
}

```

```

//left-up
if(isCanMove(X = parentNode->X - 1, Y = parentNode->Y - 1))
{
    creatSeccessionNodeG(parentNode, X, Y,14);
}
//left-down
if(isCanMove(X = parentNode->X - 1, Y = parentNode->Y + 1))
{
    creatSeccessionNodeG(parentNode, X, Y,14);
}
//right-up
if(isCanMove(X = parentNode->X + 1, Y = parentNode->Y - 1))
{
    creatSeccessionNodeG(parentNode, X, Y,14);
}
//right-down
if(isCanMove(X = parentNode->X + 1, Y = parentNode->Y + 1))
{
    creatSeccessionNodeG(parentNode, X, Y,14);
}
PutintoClosed(parentNode);
}

/*****
*****
* 函数名称      cleanStack
* 函数作用      释放Open 与Close 表的动态内存
* 函数输入      无
* 函数输出      无
*****
*****/

void cleanStack()
{
    Lstack temp = Open -> next;
    Lnode p_node;
    while(temp != NULL)
    {
        Lstack head = temp;
        temp = temp->next;
        p_node = head->npoint;
        free(p_node);
        free( head );
        Open->next = temp;
    }
}

```

```

    }
    temp = Closed -> next;
    while(temp != NULL)
    {
        Lstack head = temp;
        temp = temp->next;
        p_node = head->npoint;
        free(p_node);
        free( head );
        Closed -> next = temp;
    }
    free(Open);
    free(Closed);
}

/*****
*****
* 函数名称      getPath
* 函数作用      寻找路径信息并赋值方向给 direction 数组
* 函数输入      startX, startY 开始点坐标, endX, endY 结束点坐标,
direction 方向数组的一维指针, step 跳跃步数
*              flag_gui 过程可视化开关
* 函数提示      flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void getPath(int startX,int startY,int endX,int endY,int *direction,
int *step,int flag_gui)
{
    Lnode startNode=(Lnode)malloc(sizeof(Node));
    Lnode bestNode=NULL,bestNode1=NULL,bestNode1Parent=NULL;
    int i,j,nodeSum=0,nodeIndex=0;
    if(flag_gui)
    {
        Solid_Bar(227,5,246,29,BLACK);
        putsz(229,5,24,29,RED,2);
    }
    //init first node and Open stack
    startNode->parent= NULL;
    startNode->X = startX;
    startNode->Y = startY;
    startNode->g = 0;
    startNode->h = getH(startX,startY,endX,endY);

```



```

startNode->f = startNode->g + startNode->h;
PutintoOpen(startNode);
if(flag_gui)
{
    Solid_Bar(227,5,246,29,BLACK);
    putsz(229,5,24,29,RED,3);
    Solid_Bar(227,5,246,29,BLACK);
}
while(1)
{
    bestNode=getFminNodeFromOpen();
    if(flag_gui)
    {
        Solid_Bar(227,5,246,29,BLACK);
        putsz(229,5,24,29,RED,4);
    }
    if(bestNode->X==endX&&bestNode->Y==endY)
    {
        if(flag_gui)
        {
            Solid_Bar(227,5,246,29,BLACK);
            putsz(229,5,24,29,RED,5);
        }
        bestNode1 = bestNode,nodeSum = 0,nodeIndex = 0;
        while( bestNode1->parent != NULL )
        {
            bestNode1 = bestNode1->parent;
            nodeSum += 1;
        }
        bestNode1=bestNode,nodeIndex=nodeSum-1;
        if(flag_gui)
        {
            Solid_Bar(227,5,246,29,BLACK);
            putsz(229,5,24,29,RED,6);
        }
        while(bestNode1->parent!= NULL&&nodeIndex>=0)
        {
            if(flag_gui)
            {
                putsz(229,5,24,29,RED,7);
            }
            bestNode1Parent = bestNode1->parent;
            //output path

```

```

        if( bestNode1Parent->X - bestNode1->X == 0 && bestNode1Parent->Y - bestNode1->Y == +1)
        {
            direction[nodeIndex+(*step)] = 1;
            //up
        }
        else if( bestNode1Parent->X - bestNode1->X == 0 &
& bestNode1Parent->Y - bestNode1->Y == -1)
        {
            direction[nodeIndex+(*step)] = 2;
            //down
        }
        else if( bestNode1Parent->X - bestNode1->X == +1
&& bestNode1Parent->Y - bestNode1->Y == 0)
        {
            direction[nodeIndex+(*step)] = 3;
            //left
        }
        else if( bestNode1Parent->X - bestNode1->X == -
1 && bestNode1Parent->Y - bestNode1->Y == 0)
        {
            direction[nodeIndex+(*step)] = 4;
            //right
        }
        else if( bestNode1Parent->X - bestNode1->X == +1
&& bestNode1Parent->Y - bestNode1->Y == +1)
        {
            direction[nodeIndex+(*step)] = 5;
            //left-up
            if(direction[nodeIndex+(*step)+1]==6)
            {
                if(map_process[bestNode1->X][bestNode1Parent->Y]==0)
                {
                    direction[nodeIndex+(*step)+1]=3;
                    direction[nodeIndex+(*step)]=3;
                }
            }
            else if(direction[nodeIndex+(*step)+1]==7)
            {
                if(map_process[bestNode1Parent->X][bestNode1->Y]==0)
                {

```

```

        direction[nodeIndex+(*step)+1]=1;
        direction[nodeIndex+(*step)]=1;
    }
}
//make avenue straight
}
else if( bestNode1Parent->X - bestNode1->X == +1
&& bestNode1Parent->Y - bestNode1->Y == -1 )
{
    direction[nodeIndex+(*step)] = 6;
    //left-down
    if(direction[nodeIndex+(*step)+1]==8)
    {
        if(map_process[bestNode1Parent->X][bestNo
de1->Y]==0)
        {
            direction[nodeIndex+(*step)+1]=2;
            direction[nodeIndex+(*step)]=2;
        }
    }
    else if(direction[nodeIndex+(*step)+1]==5)
    {
        if(map_process[bestNode1->X][bestNode1Par
ent->Y]==0)
        {
            direction[nodeIndex+(*step)+1]=3;
            direction[nodeIndex+(*step)]=3;
        }
    }
    //make avenue straight
}
else if( bestNode1Parent->X - bestNode1->X == -
1 && bestNode1Parent->Y - bestNode1->Y == +1 )
{
    direction[nodeIndex+(*step)] = 7;
    //right-up
    if(direction[nodeIndex+(*step)+1]==8)
    {
        if(map_process[bestNode1->X][bestNode1Par
ent->Y]==0)
        {
            direction[nodeIndex+(*step)+1]=4;
            direction[nodeIndex+(*step)]=4;

```

```

        }
    }
    else if(direction[nodeIndex+(*step)+1]==5)
    {
        if(map_process[bestNode1Parent->X][bestNo
de1->Y]==0)
        {
            direction[nodeIndex+(*step)+1]=1;
            direction[nodeIndex+(*step)]=1;
        }
    }
    //make avenue straight
}
else if( bestNode1Parent->X - bestNode1->X == -
1 && bestNode1Parent->Y - bestNode1->Y == -1 )
{
    direction[nodeIndex+(*step)] = 8;
    //right-down
    if(direction[nodeIndex+(*step)+1]==6)
    {
        if(map_process[bestNode1Parent->X][bestNo
de1->Y]==0)
        {
            direction[nodeIndex+(*step)+1]=2;
            direction[nodeIndex+(*step)]=2;
        }
    }
    else if(direction[nodeIndex+(*step)+1]==7)
    {
        if(map_process[ bestNode1->X][bestNode1Par
ent->Y]==0)
        {
            direction[nodeIndex+(*step)+1]=4;
            direction[nodeIndex+(*step)]=4;
        }
    }
    //make avenue straight
}
nodeIndex--;
bestNode1=bestNode1->parent;
if(flag_gui)
{
    Solid_Bar(227,5,246,29,BLACK);

```

```

        }
    }
    (*step)+=nodeSum;
    if(flag_gui)
    {
        Solid_Bar(227,5,246,29,BLACK);
        putsz(229,5,24,29,RED,8);
    }
    break;
}
if(flag_gui)
{
    Solid_Bar(227,5,246,29,BLACK);
}
seachSeccessionNode(bestNode,endX,endY);
//generate son-node
}
}

/*****
*****
* 函数名称      SearchAll
* 函数作用      用BFS 算法搜索所有可行节点
* 函数输入      startX, startY 开始点坐标, flag_gui 过程可视化开关
* 函数提示      flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void SearchAll(int startX,int startY,int flag_gui)
{
    Lnode startNode=(Lnode)malloc(sizeof(Node));
    Lnode bestNode=NULL;
    Open=(Lstack)malloc(sizeof(Stack));
    Open->next = NULL;
    Closed=(Lstack)malloc(sizeof(Stack));
    Closed->next=NULL;
    startNode->parent= NULL;
    startNode->X = startX;
    startNode->Y = startY;
    startNode->g = 0;
    PutintoOpen(startNode);
    if(flag_gui)
    {

```

```

        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Start SearchAll");
    }
    while(1)
    {
        bestNode=getGminNodeFromOpen();
        if(bestNode==NULL)
        {
            if(flag_gui)
            {
                Solid_Bar(5,5,246,29,BLACK);
                putzm(5,5,24,16,RED,"FinishSearchAll");
            }
            break;
        }
        seachSeccessionNodeG(bestNode);
    }
}

/*****
*****
* 函数名称      Searchpath
* 函数作用      用A*算法搜索最短可行路径
* 函数输入      starterX, starterY 开始点坐标, destinationX,
destinationY 结束点坐标, direction 方向数组的一维指针, step 跳跃步数
*              flag_gui 过程可视化开关
* 函数提示      flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Searchpath(int starterX,int starterY,int destinationX,int destinationY,int *direction,int *step,int flag_gui)
{
    Open=(Lstack)malloc(sizeof(Stack));
    Open->next = NULL;
    Closed=(Lstack)malloc(sizeof(Stack));
    Closed->next = NULL;
    if(flag_gui)
    {
        Solid_Bar(227,5,246,29,BLACK);
        putsz(229,5,24,29,RED,1);
    }
    getPath(starterX,starterY,destinationX,destinationY,direction

```

```

,step,flag_gui);
    cleanStack();
}

/*****
*****
* 函数名称      Findpath0
* 函数作用      创建链表, 计算标准路径, 检查可行节点, 生成加入点位
* 函数输入      people 人数, zone 区域类型, weather 天气类型, point
                点位数量, point_x 点位x 值地址, point_y 点位y 值地址
*              direction 方向一维数组地址, addedpoint_x 加入点位x
                值地址, addedpoint_y 加入点位y 值地址
*              flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数提示      flag_debug 开启调试功能
*              flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Findpath0(int people,int zone,int weather,int point,int* poi
nt_x,int* point_y,int *direction,int *addedpoint_x,int *addedpoin
t_y,int flag_gui,int flag_debug)
{
    int i,starterX,starterY,destinationX,destinationY,step;
    int j,index,x,y,way[40][28];
    int wayminY[36],waymaxY[36],acupoint[36][7],acdpoint[36][7],a
cupointnum[36],acdpointnum[36];
    int sumup,sumdown,xap[4],yapY[4],px[2];
    FILE *fp;
    Lnode p_node;
    Lstack temp;
Map_illegal:
    randomize();
    step=0,x=0,y=0,sumup=0,sumdown=0;
    for(i=0;i<120;i++)
    {
        direction[i]=0;
    }
    for(i=5;i<36;i++)
    {
        wayminY[i]=0;
        waymaxY[i]=0;
        for(j=0;j<7;j++)
        {

```

```

        acupoint[i][j]=0;
        acdpoint[i][j]=0;
    }
    acupointnum[i]=0;
    acdpointnum[i]=0;
}
//caculate standard way
for(i=0;i<point-1;i++)
{
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Start Search");
        putsz(213,5,24,29,RED,i);
    }
    starterX=point_x[i];
    starterY=point_y[i];
    destinationX=point_x[i+1];
    destinationY=point_y[i+1];
    Searchpath(starterX,starterY,destinationX,destinationY,direction,&step,flag_gui);
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish Search");
}
//generate way[40][28]: 0 walkable, 1 obscale
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Start 0,1");
}
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        way[i][j]=map_process[i][j];
    }
}
//generate way[40][28]: 7 accessible
if(flag_gui)
{

```



```

        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Start Access 7");
    }
    if(zone==1)
    {
        //search all to generate 7 accessible
        SearchAll(0,0,flag_gui);
        temp = Open -> next;
        while(temp != NULL)
        {
            Lstack head = temp;
            temp = temp->next;
            p_node = head->npoint;
            free(p_node);
            free( head );
            Open->next = temp;
        }
        temp=Closed->next;
        while(temp != NULL)
        {
            Lstack head = temp;
            temp = temp->next;
            p_node = head->npoint;
            way[p_node->X][p_node->Y]=7;
            free(p_node);
            free( head );
            Closed->next = temp;
        }
        free(Open);
        free(Closed);
    }
    else
    {
        //every 0 is 7 accessible
        for(j=0;j<28;j++)
        {
            for(i=0;i<40;i++)
            {
                if(way[i][j]==0)
                {
                    way[i][j]=7;
                }
            }
        }
    }

```

```

    }
}
//generate way[40][28]: 8 way
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Start Way 8");
}
i=0;
while(direction[i]!=0)
{
    if(direction[i]==1)
    {
        y--;
    }
    else if(direction[i]==2)
    {
        y++;
    }
    else if(direction[i]==3)
    {
        x--;
    }
    else if(direction[i]==4)
    {
        x++;
    }
    else if(direction[i]==5)
    {
        x--;
        y--;
    }
    else if(direction[i]==6)
    {
        x--;
        y++;
    }
    else if(direction[i]==7)
    {
        x++;
        y--;
    }
    else if(direction[i]==8)

```

```

        {
            x++;
            y++;
        }
        way[x][y]=8;
        i++;
    }
    //generate way[40][28]: 9 point
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Start Point 9");
    }
    for(i=0;i<4;i++)
    {
        way[point_x[i]][point_y[i]]=9;
    }
    //generate way[40][28]: 6 added-point
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Adp min & max");
    }
    //get wayminY and waymaxY
    for(i=5;i<36;i++)
    {
        for(j=0;j<28;j++)
        {
            if(way[i][j]==8|way[i][j]==9)
            {
                if(wayminY[i]==0&&waymaxY[i]==0)
                {
                    wayminY[i]=j;
                    waymaxY[i]=j;
                }
                else
                {
                    if(j>waymaxY[i])
                    {
                        waymaxY[i]=j;
                    }
                    if(j<wayminY[i])
                    {

```

```

        wayminY[i]=j;
    }
}
}
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Adpoint Collect");
}
//collect available node
for(i=5;i<36;i++)
{
    for(j=0;j<28;j++)
    {
        if(way[i][j]==7)
        {
            if(j<wayminY[i])
            {
                if((wayminY[i]-j)>=3&&(wayminY[i]-j)<=9)
                {
                    acupoint[i][acupointnum[i]]=j;
                    acupointnum[i]++;
                    if(i<20)
                    {
                        sumup++;
                    }
                }
            }
            else if(j>waymaxY[i])
            {
                if((j-waymaxY[i])>=3&&(j-waymaxY[i])<=9)
                {
                    acdpoint[i][acdpointnum[i]]=j;
                    acdpointnum[i]++;
                    if(i>=20)
                    {
                        sumdown++;
                    }
                }
            }
        }
    }
}

```

```

    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Adpoint Delete");
}
//delete node between point
if(zone!=1)
{
    if(point==3)
    {
        for(i=point_x[1]-1;i<=point_x[1]+1;i++)
        {
            if(acupointnum[i]>0)
            {
                for(j=0;j<7;j++)
                {
                    acupoint[i][j]=0;
                }
                acupointnum[i]=0;
            }
            if(acdpointnum[i]>0)
            {
                for(j=0;j<7;j++)
                {
                    acdpoint[i][j]=0;
                }
                acdpointnum[i]=0;
            }
        }
    }
}
else if(point==4)
{
    for(i=point_x[1]-1;i<=point_x[1]+1;i++)
    {
        if(acupointnum[i]>0)
        {
            for(j=0;j<7;j++)
            {
                acupoint[i][j]=0;
            }
            acupointnum[i]=0;
        }
    }
}

```

```

    }
    if(acdpointnum[i]>0)
    {
        for(j=0;j<7;j++)
        {
            acdpoint[i][j]=0;
        }
        acdpointnum[i]=0;
    }
}
for(i=point_x[2]-1;i<=point_x[2]+1;i++)
{
    if(akupointnum[i]>0)
    {
        for(j=0;j<7;j++)
        {
            acupoint[i][j]=0;
        }
        acupointnum[i]=0;
    }
    if(acdpointnum[i]>0)
    {
        for(j=0;j<7;j++)
        {
            acdpoint[i][j]=0;
        }
        acdpointnum[i]=0;
    }
}
}
else
{
    if(point==3)
    {
        for(i=point_x[1]-1;i<=point_x[1]+1;i++)
        {
            if(akupointnum[i]>0)
            {
                for(j=0;j<akupointnum[i];j++)
                {
                    if(acupoint[i][j]>=point_y[1]-
1&&acupoint[i][j]<=point_y[1]-1)

```

```

        {
            for(index=j;index<acupointnum[i]-
1;index++)
            {
                acupoint[i][index]=acupoint[i][in
dex+1];
            }
            acupointnum[i]--;
        }
    }
    if(acdpointnum[i]>0)
    {
        for(j=0;j<acdpointnum[i];j++)
        {
            if(acdpoint[i][j]>=point_y[1]-
1&&acdpoint[i][j]<=point_y[1]-1)
            {
                for(index=j;index<acdpointnum[i]-
1;index++)
                {
                    acdpoint[i][index]=acdpoint[i][in
dex+1];
                }
                acdpointnum[i]--;
            }
        }
    }
}
else if(point==4)
{
    for(i=point_x[1]-1;i<=point_x[1]+1;i++)
    {
        if(acupointnum[i]>0)
        {
            for(j=0;j<acupointnum[i];j++)
            {
                if(acupoint[i][j]>=point_y[1]-
1&&acupoint[i][j]<=point_y[1]-1)
                {
                    for(index=j;index<acupointnum[i]-
1;index++)

```

```

        {
            acupoint[i][index]=acupoint[i][in
dex+1];
        }
        acupointnum[i]--;
    }
}
if(acdpointnum[i]>0)
{
    for(j=0;j<acdpointnum[i];j++)
    {
        if(acdpoint[i][j]>=point_y[1]-
1&&acdpoint[i][j]<=point_y[1]-1)
        {
            for(index=j;index<acdpointnum[i]-
1;index++)
            {
                acdpoint[i][index]=acdpoint[i][in
dex+1];
            }
            acdpointnum[i]--;
        }
    }
}
for(i=point_x[2]-1;i<=point_x[2]+1;i++)
{
    if(acupointnum[i]>0)
    {
        for(j=0;j<acupointnum[i];j++)
        {
            if(acupoint[i][j]>=point_y[2]-
1&&acupoint[i][j]<=point_y[2]-1)
            {
                for(index=j;index<acupointnum[i]-
1;index++)
                {
                    acupoint[i][index]=acupoint[i][in
dex+1];
                }
                acupointnum[i]--;
            }
        }
    }
}

```



```

        }
    }
    if(acdpointnum[i]>0)
    {
        for(j=0;j<acdpointnum[i];j++)
        {
            if(acdpoint[i][j]>=point_y[2]-
1&&acdpoint[i][j]<=point_y[2]-1)
            {
                for(index=j;index<acdpointnum[i]-
1;index++)
                {
                    acdpoint[i][index]=acdpoint[i][in
dex+1];
                }
                acdpointnum[i]--;
            }
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Adpoint Random");
}
//generate random added-point
if(point==2)
{
    //Left added-point
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Adpoint Left");
    }
    index=1;
    xap[1]=random(10)+10;
    while(acdpointnum[xap[1]]==0)
    {
        randomize();
        xap[1]=random(10)+10;
        index++;
    }
}

```

```

    if(index>21000)
    {
        //scan all to avoid time waiting
        for(j=15,i=15;i<20;i++,j--)
        {
            if(acdpointnum[i]!=0)
            {
                xap[1]=i;
                break;
            }
            else if(acdpointnum[j]!=0)
            {
                xap[1]=j;
                break;
            }
        }
        if(acdpointnum[10]!=0)
        {
            xap[1]=10;
            break;
        }
        //map without adp: regenerate a map
        Map_Random(zone,point,point_x,point_y,weather,fla
g_gui,flag_debug);
        goto Map_illegal;
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[1]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[1]=random(acdpointnum[xap[1]]);
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[1]");
}
addedpoint_y[1]=acdpoint[xap[1]][yapY[1]];

```

```

if(sumup>30)
{
    index=1;
    xap[0]=random(10)+10;
    while(acupointnum[xap[0]]==0)
    {
        randomize();
        xap[0]=random(10)+10;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            for(j=15,i=15;i<20;i++,j--)
            {
                if(acupointnum[i]!=0)
                {
                    xap[0]=i;
                    break;
                }
                else if(acupointnum[j]!=0)
                {
                    xap[0]=j;
                    break;
                }
            }
            if(acupointnum[10]!=0)
            {
                xap[0]=10;
                break;
            }
            //map without adp: regenerate a map
            Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
            goto Map_illegal;
        }
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish xap[0]");
        putzm(5,30,24,16,RED,"Time:");
        putsz(85,30,24,16,RED,index);
        delay(500);
    }
}

```

```

        Solid_Bar(5,30,165,54,BLACK);
    }
    yapY[0]=random(acupointnum[xap[0]]);
    addedpoint_y[0]=acupoint[xap[0]][yapY[0]];
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish yap[0]");
    }
}
else
{
    index=1;
    xap[0]=random(10)+10;
    while(acdpointnum[xap[0]]==0||(xap[0]==xap[1]))
    {
        randomize();
        xap[0]=random(10)+10;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            for(j=15,i=15;i<20;i++,j--)
            {
                if(acdpointnum[i]!=0&&i!=xap[1])
                {
                    xap[0]=i;
                    break;
                }
                else if(acdpointnum[j]!=0&&j!=xap[1])
                {
                    xap[0]=j;
                    break;
                }
            }
            if(acdpointnum[10]!=0&&10!=xap[1])
            {
                xap[0]=10;
                break;
            }
            //map without adp: regenerate a map
            Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);

```

```

        goto Map_illegal;
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[0]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[0]=random(acdpointnum[xap[0]]);
addedpoint_y[0]=acdpoint[xap[0]][yapY[0]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[0]");
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Adpoint Right");
}
//right added-point
index=1;
xap[2]=random(10)+20;
while(acupointnum[xap[2]]==0)
{
    randomize();
    xap[2]=random(10)+20;
    index++;
    if(index>21000)
    {
        //scan all to avoid time waiting
        for(j=25,i=25;i<30;i++,j--)
        {
            if(acupointnum[i]!=0)
            {
                xap[2]=i;
                break;
            }
        }
    }
}

```

```

        else if(acupointnum[j]!=0)
        {
            xap[2]=j;
            break;
        }
    }
    if(acupointnum[20]!=0)
    {
        xap[2]=20;
        break;
    }
    //map without adp: regenerate a map
    Map_Random(zone,point,point_x,point_y,weather,fla
g_gui,flag_debug);
    goto Map_illegal;
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[2]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[2]=random(acupointnum[xap[2]]);
addedpoint_y[2]=acupoint[xap[2]][yapY[2]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[2]");
}
if(sumdown>30)
{
    index=1;
    xap[3]=random(10)+20;
    while(acdpointnum[xap[3]]==0)
    {
        randomize();
        xap[3]=random(10)+20;
        index++;
        if(index>21000)

```

```

{
    //scan all to avoid time waiting
    for(j=25,i=25;i<30;i++,j--)
    {
        if(acdpointnum[i]!=0)
        {
            xap[3]=i;
            break;
        }
        else if(acdpointnum[j]!=0)
        {
            xap[3]=j;
            break;
        }
    }
    if(acdpointnum[20]!=0)
    {
        xap[3]=20;
        break;
    }
    //map without adp: regenerate a map
    Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
    goto Map_illegal;
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[3]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[3]=random(acdpointnum[xap[3]]);
addedpoint_y[3]=acdpoint[xap[3]][yapY[3]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[3]");
}
}

```

```

else
{
    index=1;
    xap[3]=random(10)+20;
    while(acupointnum[xap[3]]==0||(xap[3]==xap[2]))
    {
        randomize();
        xap[3]=random(10)+20;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            for(j=25,i=25;i<30;i++,j--)
            {
                if(acupointnum[i]!=0&&i!=xap[2])
                {
                    xap[3]=i;
                    break;
                }
                else if(acupointnum[j]!=0&&i!=xap[2])
                {
                    xap[3]=j;
                    break;
                }
            }
            if(acupointnum[20]!=0&&20!=xap[2])
            {
                xap[3]=20;
                break;
            }
            //map without adp: regenerate a map
            Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
            goto Map_illegal;
        }
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish xap[3]");
        putzm(5,30,24,16,RED,"Time:");
        putsz(85,30,24,16,RED,index);
        delay(500);
    }
}

```



```

        Solid_Bar(5,30,165,54,BLACK);
    }
    yapY[3]=random(acupointnum[xap[3]]);
    addedpoint_y[3]=acupoint[xap[3]][yapY[3]];
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish yap[3]");
    }
}
else
{
    if(point==3)
    {
        px[0]=point_x[1];
        px[1]=point_x[1];
    }
    else
    {
        px[0]=point_x[1];
        px[1]=point_x[2];
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Adpoint Left");
    }
    //left added-point:  $x = 5 - (px[0] - 1)$ 
    index=1;
    xap[1]=random(px[0]-5)+5;
    while(acdpointnum[xap[1]]==0)
    {
        randomize();
        xap[1]=random(px[0]-5)+5;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            if(xap[1]%2==0)
            {
                for(i=5;i<px[0];i++)
                {

```

```

        if(acdpointnum[i]!=0)
        {
            xap[1]=i;
            break;
        }
    }
}
else
{
    for(j=px[0]-1;j>=5;j--)
    {
        if(acdpointnum[j]!=0)
        {
            xap[1]=j;
            break;
        }
    }
}
//map without adp: regenerate a map
Map_Random(zone,point,point_x,point_y,weather,flag_gui,flag_debug);
    goto Map_illegal;
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[1]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[1]=random(acdpointnum[xap[1]]);
addedpoint_y[1]=acdpoint[xap[1]][yapY[1]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[1]");
}
if(sumup>30)
{
    index=1;

```

```

xap[0]=random(px[0]-5)+5;
while(akupointnum[xap[0]]==0)
{
    randomize();
    xap[0]=random(px[0]-5)+5;
    index++;
    if(index>21000)
    {
        //scan all to avoid time waiting
        if(xap[0]%2==0)
        {
            for(i=5;i<px[0];i++)
            {
                if(akupointnum[i]!=0)
                {
                    xap[0]=i;
                    break;
                }
            }
        }
        else
        {
            for(j=px[0]-1;j>=5;j--)
            {
                if(akupointnum[j]!=0)
                {
                    xap[0]=j;
                    break;
                }
            }
        }
        //map without adp: regenerate a map
        Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
        goto Map_illegal;
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[0]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
}

```

```

        delay(500);
        Solid_Bar(5,30,165,54,BLACK);
    }
    yapY[0]=random(acupointnum[xap[0]]);
    addedpoint_y[0]=acupoint[xap[0]][yapY[0]];
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish yap[0]");
    }
}
else
{
    index=1;
    xap[0]=random(px[0]-5)+5;
    while(acdpointnum[xap[0]]==0||(xap[0]==xap[1]))
    {
        randomize();
        xap[0]=random(px[0]-5)+5;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            if(xap[0]%2==0)
            {
                for(i=5;i<px[0];i++)
                {
                    if(acdpointnum[i]!=0&&i!=xap[1])
                    {
                        xap[0]=i;
                        break;
                    }
                }
            }
            else
            {
                for(j=px[0]-1;j>=5;j--)
                {
                    if(acdpointnum[j]!=0&&j!=xap[1])
                    {
                        xap[0]=j;
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    //map without adp: regenerate a map
    Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
    goto Map_illegal;
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[0]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[0]=random(acdpointnum[xap[0]]);
addedpoint_y[0]=acdpoint[xap[0]][yapY[0]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[0]");
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Adpoint Right");
}
//right added-point: x = (px[1]+1)-35
index=1;
xap[2]=random(35-px[1])+px[1]+1;
while(acupointnum[xap[2]]==0)
{
    randomize();
    xap[2]=random(35-px[1])+px[1]+1;
    index++;
    if(index>21000)
    {
        //scan all to avoid time waiting
        if(xap[2]%2==0)
        {

```

```

        for(i=px[1]+1;i<36;i++)
        {
            if(acupointnum[i]!=0)
            {
                xap[2]=i;
                break;
            }
        }
    }
    else
    {
        for(j=35;j>=px[1]+1;j--)
        {
            if(acupointnum[j]!=0)
            {
                xap[2]=j;
                break;
            }
        }
    }
    //map without adp: regenerate a map
    Map_Random(zone,point,point_x,point_y,weather,flag_gui,flag_debug);
    goto Map_illegal;
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[2]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[2]=random(acupointnum[xap[2]]);
addedpoint_y[2]=acupoint[xap[2]][yapY[2]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[2]");
}
}
if(sumdown>30)

```

```

{
    index=1;
    xap[3]=random(35-px[1])+px[1]+1;
    while(acdpointnum[xap[3]]==0)
    {
        randomize();
        xap[3]=random(35-px[1])+px[1]+1;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            if(xap[3]%2==0)
            {
                for(i=px[1]+1;i<36;i++)
                {
                    if(acdpointnum[i]!=0)
                    {
                        xap[3]=i;
                        break;
                    }
                }
            }
            else
            {
                for(j=35;j>=px[1]+1;j--)
                {
                    if(acdpointnum[j]!=0)
                    {
                        xap[3]=j;
                        break;
                    }
                }
            }
            //map without adp: regenerate a map
            Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
            goto Map_illegal;
        }
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish xap[3]");
    }
}

```

```

        putzm(5,30,24,16,RED,"Time:");
        putsz(85,30,24,16,RED,index);
        delay(500);
        Solid_Bar(5,30,165,54,BLACK);
    }
    yapY[3]=random(acdpointnum[xap[3]]);
    addedpoint_y[3]=acdpoint[xap[3]][yapY[3]];
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Finish yap[3]");
    }
}
else
{
    index=1;
    xap[3]=random(35-px[1])+px[1]+1;
    while(acupointnum[xap[3]]==0||(xap[3]==xap[2]))
    {
        randomize();
        xap[3]=random(35-px[1])+px[1]+1;
        index++;
        if(index>21000)
        {
            //scan all to avoid time waiting
            if(xap[3]%2==0)
            {
                for(i=px[1]+1;i<36;i++)
                {
                    if(acupointnum[i]!=0&&i!=xap[2])
                    {
                        xap[3]=i;
                        break;
                    }
                }
            }
            else
            {
                for(j=35;j>=px[1]+1;j--)
                {
                    if(acupointnum[j]!=0&&j!=xap[2])
                    {
                        xap[3]=j;

```



```

                break;
            }
        }
    }
    //map without adp: regenerate a map
    Map_Random(zone,point,point_x,point_y,weather
,flag_gui,flag_debug);
    goto Map_illegal;
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish xap[3]");
    putzm(5,30,24,16,RED,"Time:");
    putsz(85,30,24,16,RED,index);
    delay(500);
    Solid_Bar(5,30,165,54,BLACK);
}
yapY[3]=random(acupointnum[xap[3]]);
addedpoint_y[3]=acupoint[xap[3]][yapY[3]];
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Finish yap[3]");
}
}
}
for(i=0;i<4;i++)
{
    way[xap[i]][addedpoint_y[i]]=6;
    addedpoint_x[i]=xap[i];
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\WAY\\WAY0.txt","w+");
    fprintf(fp,"Competition Information\n");
    fprintf(fp,"\npeople = %d, zone = %d, weather = %d, point
= %d\n",people,zone,weather,point);
    fprintf(fp,"\nMap Information:\n");
    fprintf(fp,"\nmap_process[40][28] for eval\n");
    for(i=0;i<40;i++)

```

```

{
    for(j=0;j<28;j++)
    {
        if(j==0)
        {
            fprintf(fp,"%d,",map_process[i][j]);
        }
        else if(j==27)
        {
            if(i==39)
            {
                fprintf(fp,"%d}\n",map_process[i][j]);
            }
            else
            {
                fprintf(fp,"%d},\n",map_process[i][j]);
            }
        }
        else
        {
            fprintf(fp,"%d,",map_process[i][j]);
        }
    }
}
fprintf(fp,"\nStandard Way Information\n");
fprintf(fp,"\nReference\n1:up 2:down 3:left 4:right\n5
:left-up 6:left-down 7:right-up 8:right-down\n");
i=0;
while(direction[i]!=0)
{
    fprintf(fp,"direction[%d] = %d\n",i,direction[i]);
    i++;
}
fprintf(fp,"\nAdded-point Information:\n");
fprintf(fp,"\nwayminY 5-35:\n");
for(i=5;i<36;i++)
{
    if(i==36)
    {
        if(wayminY[i]<10)
        {
            fprintf(fp," 0%d \n",wayminY[i]);
        }
    }
}

```

```

        else
        {
            fprintf(fp," %d \n",wayminY[i]);
        }
    }
    else
    {
        if(wayminY[i]<10)
        {
            fprintf(fp," 0%d ",wayminY[i]);
        }
        else
        {
            fprintf(fp," %d ",wayminY[i]);
        }
    }
}
fprintf(fp,"\nwaymaxY 5-35:\n");
for(i=5;i<36;i++)
{
    if(i==36)
    {
        if(waymaxY[i]<10)
        {
            fprintf(fp," 0%d \n",waymaxY[i]);
        }
        else
        {
            fprintf(fp," %d \n",waymaxY[i]);
        }
    }
    else
    {
        if(waymaxY[i]<10)
        {
            fprintf(fp," 0%d ",waymaxY[i]);
        }
        else
        {
            fprintf(fp," %d ",waymaxY[i]);
        }
    }
}
}

```

```

fprintf(fp,"\\nacupointnum 5-35:\\n");
for(i=5;i<36;i++)
{
    if(i==36)
    {
        if(acupointnum[i]<10)
        {
            fprintf(fp," 0%d \\n",acupointnum[i]);
        }
        else
        {
            fprintf(fp," %d \\n",acupointnum[i]);
        }
    }
    else
    {
        if(acupointnum[i]<10)
        {
            fprintf(fp," 0%d ",acupointnum[i]);
        }
        else
        {
            fprintf(fp," %d ",acupointnum[i]);
        }
    }
}
fprintf(fp,"\\nacdpointnum 5-35:\\n");
for(i=5;i<36;i++)
{
    if(i==36)
    {
        if(acdpointnum[i]<10)
        {
            fprintf(fp," 0%d \\n",acdpointnum[i]);
        }
        else
        {
            fprintf(fp," %d \\n",acdpointnum[i]);
        }
    }
    else
    {
        if(acdpointnum[i]<10)

```

```

        {
            fprintf(fp," 0%d ",acdpointnum[i]);
        }
        else
        {
            fprintf(fp," %d ",acdpointnum[i]);
        }
    }
}
fprintf(fp,"\\nacupoint 5-35:\\n");
for(j=0;j<7;j++)
{
    for(i=5;i<36;i++)
    {
        if(i==35)
        {
            if(acupoint[i][j]<10)
            {
                fprintf(fp," 0%d \\n",acupoint[i][j]);
            }
            else
            {
                fprintf(fp," %d \\n",acupoint[i][j]);
            }
        }
        else
        {
            if(acupoint[i][j]<10)
            {
                fprintf(fp," 0%d ",acupoint[i][j]);
            }
            else
            {
                fprintf(fp," %d ",acupoint[i][j]);
            }
        }
    }
}
fprintf(fp,"acdpoint 5-35:\\n");
for(j=0;j<7;j++)
{
    for(i=5;i<36;i++)
    {

```

```

        if(i==35)
        {
            if(acdpoint[i][j]<10)
            {
                fprintf(fp," 0%d \n",acdpoint[i][j]);
            }
            else
            {
                fprintf(fp," %d \n",acdpoint[i][j]);
            }
        }
        else
        {
            if(acdpoint[i][j]<10)
            {
                fprintf(fp," 0%d ",acdpoint[i][j]);
            }
            else
            {
                fprintf(fp," %d ",acdpoint[i][j]);
            }
        }
    }
}
for(i=0;i<4;i++)
{
    fprintf(fp,"addedpoint_x[%d]=%d;\naddedpoint_y[%d]=%d
; \n",i,addedpoint_x[i],i,addedpoint_y[i]);
}
fprintf(fp,"\nReference\n0:walkable  1:obstacle  7:access
able 8:way\n9:point  6:addedpoint\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(j==0&& i==0)
        {
            for(i=0;i<40;i++)
            {
                if(i==39)
                {
                    if(i<10)
                    {

```

```

        fprintf(fp," %d \n",i);
    }
    else
    {
        fprintf(fp," %d \n",i%10);
    }
}
else if(i==0)
{
    if(i<10)
    {
        fprintf(fp,"    %d ",i);
    }
    else
    {
        fprintf(fp,"    %d ",i%10);
    }
}
else
{
    if(i<10)
    {
        fprintf(fp," %d ",i);
    }
    else
    {
        fprintf(fp," %d ",i%10);
    }
}
}
i=0;
}
if(i==39)
{
    if(way[i][j]<10)
    {
        fprintf(fp," %d \n",way[i][j]);
    }
    else
    {
        fprintf(fp,"%d \n",way[i][j]);
    }
}
}

```

```

        else if(i==0)
        {
            if(way[i][j]<10)
            {
                fprintf(fp,"%d %d ",j,way[i][j]);
            }
            else
            {
                fprintf(fp,"%d %d ",j%10,way[i][j]);
            }
        }
        else
        {
            if(way[i][j]<10)
            {
                fprintf(fp," %d ",way[i][j]);
            }
            else
            {
                fprintf(fp,"%d ",way[i][j]);
            }
        }
    }
}
fclose(fp);
}
}

```

10. gins.c

```
#include "headfile.h"
```

```

/*****
*****
"GINS.h"
版本:V1.1
更新日期 : 2022.2.9 20:08
*****/

/*****
*****
* 函数名称      Solid_Triangle
* 函数作用      绘制实心三角形

```



```

* 函数输入      x0,y0,x1,y1,x2,y2 三点坐标, color 颜色
* 函数输出      无
*****
*****/

```

```

void Solid_Triangle(int x0,int y0,int x1,int y1,int x2,int y2,int
color)
{
    float m0,m1,m2,b0,b1,b2; //三直线数据
    int start,end;
    int y=0;
    YMAX_Triswap(&x0,&y0,&x1,&y1,&x2,&y2); //排序
    if(y2-y0==0)
    {
        m0=65535;
    }
    else
    {
        m0=(float)(x2-x0)/(float)(y2-y0);
    }
    if(y2-y1==0)
    {
        m1=65535;
    }
    else
    {
        m1=(float)(x2-x1)/(float)(y2-y1);
    }
    if(y1-y0==0)
    {
        m2=65535;
    } //斜率为0 情况排除
    else
    {
        m2=(float)(x1-x0)/(float)(y1-y0);
    }
    b0=x2-m0*y2;
    b1=x2-m1*y2;
    b2=x1-m2*y1;
    for(y=y2;y>y1;y--)
    {
        start=(int)(m0*y+b0);
        end=(int)(m1*y+b1);
    }
}

```

```

        Line_Plus(start,y,end,y,color); //扫描第一段
    }
    for(y=y1;y>y0;y--)
    {
        start=(int)(m0*y+b0);
        end=(int)(m2*y+b2);
        Line_Plus(start,y,end,y,color); //扫描第二段
    }
    Line_Plus(x0,y0,x1,y1,color);
    Line_Plus(x0,y0,x2,y2,color);
    Line_Plus(x1,y1,x2,y2,color);
    //绘制外边框
}

/*****
*****
* 函数名称      Diswap
* 函数作用      两数交换
* 函数输入      a,b 两数地址
* 函数输出      无
*****
*****/

void Diswap(int *a,int *b)
{
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}

/*****
*****
* 函数名称      Long_Diswap
* 函数作用      两Long int 交换
* 函数输入      a,b 两数地址
* 函数输出      无
*****
*****/

void Long_Diswap(long int *a,long int *b)
{
    long int temp;

```

```

    temp=*a;
    *a=*b;
    *b=temp;
}

/*****
*****
* 函数名称      YMAX_Trishwap
* 函数作用      按Y 值从小到大排列点
* 函数输入      x0,y0,x1,y1,x2,y2 三点坐标
* 函数输出      无
*****
*****/

void YMAX_Trishwap(int *x0,int *y0,int *x1,int *y1,int *x2,int *y2
)
{
    if(*y0>*y1)
    {
        Diswap(x0,x1);
        Diswap(y0,y1);
    }
    if(*y1>*y2)
    {
        Diswap(x1,x2);
        Diswap(y1,y2);
    }
    if(*y0>*y1)
    {
        Diswap(x0,x1);
        Diswap(y0,y1);
    }
}

/*****
*****
* 函数名称      Solid_HalfSector_up
* 函数作用      绘制实心上半圆
* 函数输入      x,y 圆心坐标, r 半径, color 颜色
* 函数输出      无
*****
*****/

```

```

void Solid_HalfSector_up(int x,int y,int r,int color)
{
    Solid_QuarterSector_right_up(x,y,r,color);
    Solid_QuarterSector_left_up(x,y,r,color);
}

/*****
*****
* 函数名称      Solid_HalfSector_up
* 函数作用      绘制实心下半圆
* 函数输入      x,y 圆心坐标, r 半径, color 颜色
* 函数输出      无
*****
*****/

void Solid_HalfSector_down(int x,int y,int r,int color)
{
    Solid_QuarterSector_right_down(x,y,r,color);
    Solid_QuarterSector_left_down(x,y,r,color);
}

/*****
*****
* 函数名称      YMAX_Array_Sort
* 函数作用      按Y 值从小到大排列点
* 函数输入      p 点的坐标结构, Length 排列长度
* 函数输出      无
*****
*****/

void YMAX_Array_Sort(struct P_oimt *p,int length)
{
    int i,j;
    struct P_oimt temp;
    for(i=0;i<length;i++)
    {
        for(j=0;j<length-i-1;j++)
        {
            if((p+j)->y>(p+j+1)->y)
            {
                temp=*(p+j);
                *(p+j)=*(p+j+1);
                *(p+j+1)=temp;
            }
        }
    }
}

```

```

    }
}
}

/*****
*****
* 函数名称      Solid_Quadrangle
* 函数作用      绘制实心四边形
* 函数输入      x0,y0,x1,y1,x2,y2,x3,y3 四点坐标, color 颜色
* 函数输出      无
*****
*****/

void Solid_Quadrangle(int x0,int y0,int x1,int y1,int x2,int y2,int x3,int y3,int color)
{
    struct P_point Quad[4];
    float m0,m1,m2,m3,b0,b1,b2,b3;
    int y,start,end;
    Quad[0].x=x0;
    Quad[0].y=y0;
    Quad[1].x=x1;
    Quad[1].y=y1;
    Quad[2].x=x2;
    Quad[2].y=y2;
    Quad[3].x=x3;
    Quad[3].y=y3;
    YMAX_Array_Sort(Quad,4);
    Solid_Triangle(Quad[0].x,Quad[0].y,Quad[1].x,Quad[1].y,Quad[2].x,Quad[2].y,color);
    Solid_Triangle(Quad[1].x,Quad[1].y,Quad[2].x,Quad[2].y,Quad[3].x,Quad[3].y,color);
}

/*****
*****
* 函数名称      Trimax
* 函数作用      求三数最大值
* 函数输入      a,b,c 三数
* 函数输出      三数最大值
*****
*****/

```

```

int Trimax(int a,int b,int c)
{
    int max=a;
    if(b>max)
        max=b;
    if(c>max)
        max=c;
    return max;
}

/*****
*****
* 函数名称      Trimin
* 函数作用      求三数最小值
* 函数输入      a,b,c 三数
* 函数输出      三数最小值
*****
*****/

int Trimin(int a,int b,int c)
{
    int min=a;
    if(b<min)
        min=b;
    if(c<min)
        min=c;
    return min;
}

/*****
*****
* 函数名称      Triangle_Random_Spot
* 函数作用      在三角形区域产生一定随机点
* 函数输入      x0,y0,x1,y1,x2,y2 三点坐标, number 点数, color 颜色
* 函数输出
*****
*****/

void Triangle_Random_Spot(int x0,int y0,int x1,int y1,int x2,int
y2,int number,int color)
{

```

```

float m0,m1,b0,b1;
int i=0,x,y;
m0=(float)(x2-x0)/(float)(y2-y0);
m1=(float)(x2-x1)/(float)(y2-y1);
b0=x2-m0*y2;
b1=x2-m1*y2;
randomize();
for(i=0;i<number;i++)
{
    x=random(x1-x0)+x0;
    y=random(y2-y0)+y0;
    if((x>m0*y+b0)&&(x<m1*y+b1))
    {
        Putpixel64k(x,y,color);
    }
    else
    {
        i--;
        continue;
    }
}
}

/*****
*****
* 函数名称      Circle_Random_Spot
* 函数作用      在域产生一定随机点
* 函数输入      x0,y0 点数, color 颜色
* 函数输出      无
*****
*****/

void Circle_Random_Spot(int x0,int y0,int radius,int number,int color)
{
    int x,y,i;
    //randomize();
    for(i=0;i<number;i++)
    {
        x=random(2*radius)+x0-radius;
        y=random(2*radius)+y0-radius;
        if(((x-x0)*(x-x0)+(y-y0)*(y-y0))<radius*radius)

```

```

        {
            Putpixel64k(x,y,color);
        }
    else
    {
        i--;
        continue;
    }
}
}

/*****
*****
* 函数名称      Solid_Ellipse
* 函数作用      绘制实心椭圆
* 函数输入      x,y 椭圆中心, 赏衷渤ふ幔琤椭圆短轴
* 函数输出      无
*****
*****/

void Solid_Ellipse(double x,double y,double a,double b,int color)
{
    double i,j;
    for(i=x-a;i<=x+a;i++)
    {
        for(j=y-b;j<=y+b;j++)
        {
            if(b*b*(x-i)*(x-i)+a*a*(y-j)*(y-j)<=a*a*b*b)
            {
                Putpixel64k((int)i,(int)j,color);
            }
        }
    }
}

/*****
*****
* 函数名称      Solid_Upper_Ellipse
* 函数作用      绘制实心上半椭圆
* 函数输入      x,y 椭圆中心, a 椭圆长轴, b 椭圆短轴, color 颜色
* 函数输出      无
*****
*****/

```



```

void Solid_Upper_Ellipse(double x,double y,double a,double b,int
color)
{
    double i,j;
    for(i=x-a;i<=x+a;i++)
    {
        for(j=y-b;j<y;j++)
        {
            if(b*b*(x-i)*(x-i)+a*a*(y-j)*(y-j)<=a*a*b*b)
            {
                Putpixel64k((int)i,(int)j,color);
            }
        }
    }
}

```

```

/*****
*****
* 函数名称      Bar_Random_Spot
* 函数作用      在形区域产生一定随机点
* 函数输入      x0,y0,x1,y1 长方形对角线坐标, number 点数, color 颜色
* 函数输出      无
*****
*****/

```

```

void Bar_Random_Spot(int x0,int y0,int x1,int y1,int number,int c
olor)
{
    int x,y,i;
    for(i=0;i<number;i++)
    {
        x=random(x1-x0+1)+x0;
        y=random(y1-y0+1)+y0;
        Putpixel64k(x,y,color);
    }
}

```

```

/*****
*****
* 函数名称      Hollow_Quadrangle
* 函数作用      绘制四边形

```

```

* 函数输入      x0,y0,x1,y1,x2,y2,x3,y3 四点坐标, color 颜色
* 函数输出      无
*****
*****/

```

```

void Hollow_Quadrangle(int x0,int y0,int x1,int y1,int x2,int y2,
int x3,int y3,int color)
{
    Line_Plus(x0,y0,x1,y1,color);
    Line_Plus(x1,y1,x2,y2,color);
    Line_Plus(x2,y2,x3,y3,color);
    Line_Plus(x0,y0,x3,y3,color);
}

```

```

/*****
*****
* 函数名称      Hollow_Ellipse
* 函数作用      绘制椭圆
* 函数输入      x,y 椭圆中心, a 椭圆长轴, b 椭圆短轴, color 颜色
* 函数输出      无
*****
*****/

```

```

void Hollow_Ellipse(double x,double y,double a,double b,int color
)
{
    double i,j;
    for(i=x-a;i<=x+a;i++)
    {
        for(j=y-b;j<=y+b;j++)
        {
            if(fabs(b*b*(x-i)*(x-i)+a*a*(y-j)*(y-j)-
a*a*b*b)<500000)
            {
                Putpixel64k((int)i,(int)j,color);
            }
        }
    }
}

```

```

/*****
*****

```

```

* 函数名称      Ring
* 函数作用      绘制圆环
* 函数输入      x,y 中心,  1 小圆半径, R2 大圆半径
* 函数输出      无
*****
*****/

```

```

void Ring(int x,int y,int R1,int R2,int color)
{
    int i,j;
    for(i=x-R2;i<=x+R2;i++)
    {
        for(j=y-R2;j<=y+R2;j++)
        {
            if((i-x)*(i-x)+(j-y)*(j-y)>=R1*R1&&(i-x)*(i-x)+(j-
y)*(j-y)<=R2*R2)
            {
                Putpixel64k(i,j,color);
            }
        }
    }
}

```

11. gragh.c

```

#include"headfile.h"

/*****
*****

* @author      ytm
* @date        2022-4-4
*****
*****/

/*****
*****

* 函数名称      Line_Plus
* 函数作用      画直线(x0,y0)到(x1,y1)
* 函数输入      x0, y0 与 x1, y1 直线两 endpoint 坐标, color 绘制颜色
* 函数输出      无
*****
*****/

void Line_Plus(int x0,int y0,int x1,int y1,int color)
{
    if(y0==y1)

```

```

    {
        if(x1>x0)
        {
            Horizline(x0,y0,x1-x0+1,color);
        }
        else
        {
            Horizline(x1,y1,x0-x1+1,color);
        }
    }
    else if(x0==x1)
    {
        if(y1>y0)
        {
            Vertiline(x0,y0,y1-y0+1,color);
        }
        else
        {
            Vertiline(x1,y1,y0-y1+1,color);
        }
    }
    else
    {
        Obliqline(x0,y0,x1,y1,color);
    }
}

/*****
*****
* 函数名称      Square
* 函数作用      绘制正方形
* 函数输入      x0, y0 正方形中心坐标, length 正方形边长, color 绘制
颜色
* 函数注意      若 length 不为奇数, 会绘制 length+1 变成的正方形
* 函数输出      无
*****
*****/
void Square(int x0,int y0,int length,int color)
{
    int y,dev;
    if(length%2)
    {
        dev=(length-1)/2;

```

```

    }
    else
    {
        dev=length/2;
    }
    for(y=y0-dev;y<=y0+dev;y++)
    {
        Line_Plus(x0-dev,y,x0+dev,y,color);
    }
}

/*****
*****
* 函数名称      Hollow_Bar
* 函数作用      画对角线为(x0,y0)到(x1,y1)的空心矩形
* 函数输入      x0, y0 与 x1, y1 矩形对角线坐标, color 绘制颜色
* 函数注意      默认厚度为1
* 函数输出      无
*****
*****/
void Hollow_Bar(int x0,int y0,int x1,int y1,int color)
{
    Line_Plus(x0,y0,x0,y1,color);
    Line_Plus(x0,y0,x1,y0,color);
    Line_Plus(x1,y0,x1,y1,color);
    Line_Plus(x0,y1,x1,y1,color);
}

/*****
*****
* 函数名称      Solid_Bar
* 函数作用      画对角线为(x0,y0)到(x1,y1)的实心矩形
* 函数输入      x0, y0 与 x1, y1 矩形对角线坐标, color 绘制颜色
* 函数输出      无
*****
*****/
void Solid_Bar(int x0,int y0,int x1,int y1,int color)
{
    int i;
    if(y0>y1)
    {
        for(i=y1;i<=y0;i++)
        {

```

```

        Line_Plus(x0,i,x1,i,color);
    }
}
else
{
    for(i=y0;i<=y1;i++)
    {
        Line_Plus(x0,i,x1,i,color);
    }
}
}

/*****
*****

* 函数名称      Hollow_Triangle
* 函数作用      画三点坐标为(x0,y0), (x1,y1), (x2,y2)的空心三角形
* 函数输入      x0, y0 与 x1, y1 与 x2, y2 为三角形三点坐标, color 绘
制颜色
* 函数注意      默认厚度为1
* 函数输出      无
*****
*****/
void Hollow_Triangle(int x0,int y0,int x1,int y1,int x2,int y2,int
t color)
{
    Line_Plus(x0,y0,x1,y1,color);
    Line_Plus(x0,y0,x2,y2,color);
    Line_Plus(x1,y1,x2,y2,color);
}

/*****
*****

* 函数名称      Hollow_Circle
* 函数作用      画圆心为(xr,yr), 半径为radius 的空心圆
* 函数输入      (xr,yr)圆心坐标, radius 为半径长度, color 绘制颜色
* 函数注意      默认厚度为1
* 函数输出      无
*****
*****/
void Hollow_Circle(int xc,int yc,int radius,int color)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/

```

```

    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    //    unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*Bresenham 算法所需变量*/
    int x, y, d;
    y = radius;
    d = 3 - radius << 1;

    for (x = 0; x <= y; x++)
    {
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(yc - y) << 10) + xc;
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page - x) = color;
        *(video_buffer + page + x) = color;
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(yc + y) << 10) + xc;
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page - x) = color;
        *(video_buffer + page + x) = color;
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(yc + x) << 10) + xc;
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page - y) = color;
        *(video_buffer + page + y) = color;
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(yc - x) << 10) + xc;
        new_page = page >> 15;
        SelectPage(new_page);
        *(video_buffer + page - y) = color;
        *(video_buffer + page + y) = color;

        if (d < 0)

```

```

        {
            d += x * 4 + 6;
        }
    else
    {
        d += (x - y) * 4 + 10;
        y--;
    }
}
}

/*****
*****
* 函数名称      Solid_Circle
* 函数作用      画圆心为(xr,yr), 半径为radius 的实心圆
* 函数输入      (xr,yr) 圆心坐标, radius 为半径长度, color 绘制颜色
* 函数注意      基于 Bresenham 算法
* 函数输出      无
*****
*****/
void Solid_Circle(int xc, int yc, int radius, int color)
{
    int x = 0,
        y = radius,
        dx = 3,
        dy = 2 - radius - radius,
        d = 1 - radius;

    while (x <= y)
    {
        Line_Plus(xc - x, yc - y, xc + x, yc - y, color);
        Line_Plus(xc - y, yc - x, xc + y, yc - x, color);
        Line_Plus(xc - y, yc + x, xc + y, yc + x, color);
        Line_Plus(xc - x, yc + y, xc + x, yc + y, color);

        if (d < 0)
        {
            d += dx;
            dx += 2;
        }
        else
        {
            d += (dx + dy);

```



```

        dx += 2;
        dy += 2;
        y--;
    }
    x++;
}
}

/*****
*****

* 函数名称      Button_Draw
* 函数作用      画对角线坐标为x0, y0 与 x1, y1 的按钮
* 函数输入      x0, y0 与 x1, y1 矩形对角线坐标, ext_color 外边框颜色, int_color 内部颜色
* 函数注意      默认宽度为2
* 函数输出      无
*****
*****/
void Button_Draw(int x0,int y0,int x1,int y1,int ext_color,int int_color)
{
    Hollow_Bar(x0,y0,x1,y1,ext_color);
    Hollow_Bar(x0+1,y0+1,x1-1,y1-1,ext_color);
    Solid_Bar(x0+2,y0+2,x1-2,y1-2,int_color);
}

/*****
*****

* 函数名称      Button_Edge
* 函数作用      改变对角线坐标为x0, y0 与 x1, y1 的按钮的外圈颜色
* 函数输入      x0, y0 与 x1, y1 矩形对角线坐标, color 绘制颜色
* 函数注意      默认宽度为2
* 函数输出      无
*****
*****/
void Button_Edge(int x0,int y0,int x1,int y1,int color)
{
    Hollow_Bar(x0-1,y0-1,x1+1,y1+1,color);
    Hollow_Bar(x0-2,y0-2,x1+2,y1+2,color);
}

/*****
*****

```

```

* 函数名称      UpperBar_Draw
* 函数作用      画上方状态栏
* 函数输入      thick 厚度, edge 边框宽度, solid_color 内部颜色,
edge_color 边框颜色
* 函数输出      无
*****
*****/
void UpperBar_Draw(int thick,int edge,int solid_color,int edge_color)
{
    Solid_Bar(0,0,1023,thick-1,solid_color);
    Solid_Bar(0,thick,1023,thick+edge-1,edge_color);
}

/*****
*****/
* 函数名称      Boundary_Draw
* 函数作用      画分界条
* 函数输入      x0, y0 分界线左上角坐标, thick 厚度, color 分界线颜色
* 函数输出      无
*****
*****/
void Boundary_Draw(int x0,int y0,int thick,int color)
{
    Solid_Bar(x0,y0,x0+thick-1,767,color);
}

/*****
*****/
* 函数名称      RevertImage_Draw
* 函数作用      画32*32 的返回图标
* 函数输入      x0, y0 左上角坐标, color 绘制颜色
* 函数输出      无
*****
*****/
void RevertImage_Draw(int x0,int y0,int color)
{
    Line_Plus(x0-8,y0,x0+8,y0-16,color);
    Line_Plus(x0-8,y0,x0+8,y0+16,color);
}

/*****
*****/

```

```

*****
* 函数名称      RevertButton_Draw
* 函数作用      画40*40 的返回按钮
* 函数输入      x0, y0 左上角坐标, color 绘制颜色
* 函数输出      无
*****
*****/
void RevertButton_Draw(int x0,int y0,int color)
{
    Solid_Bar(x0-20,y0-20,x0+20,y0+20,color);
}

/*****
*****
* 函数名称      SignEdge_Draw
* 函数作用      改变对角线坐标为x0, y0 与x1, y1 的图例的外圈颜色
* 函数输入      x0, y0 与x1, y1 矩形对角线坐标, color 绘制颜色
* 函数注意      图例大小是24*24, 画这个外圈不被包括在内
*               默认宽度是2
* 函数输出      无
*****
*****/
void SignEdge_Draw(int x0,int y0,int color)
{
    Hollow_Bar(x0-1,y0-1,x0+24,y0+24,color);
    Hollow_Bar(x0-2,y0-2,x0+25,y0+25,color);
}

/*****
*****
* 函数名称      AnimationEdge_Draw
* 函数作用      改变对角线坐标为x0, y0 与x1, y1 的动画的外圈颜色
* 函数输入      x0, y0 与x1, y1 矩形对角线坐标, color 绘制颜色
* 函数注意      动画大小是64*64, 画这个外圈不被包括在内
*               默认宽度是2
* 函数输出      无
*****
*****/
void AnimationEdge_Draw(int x0,int y0,int color)
{
    Hollow_Bar(x0-1,y0-1,x0+64,y0+64,color);
    Hollow_Bar(x0-2,y0-2,x0+65,y0+65,color);
}

```

```

/*****
*****
* 函数名称      TriangleImage
* 函数作用      画33*17的菜单展开图标
* 函数输入      x0, y0 左上角坐标, color 绘制颜色
* 函数输出      无
*****
*****/
void TriangleImage(int x0, int y0, int color)
{
    int x_left=x0;
    int x_right=x0+32;
    int y=y0;
    for(;x_left<=x_right;y++)
    {
        Line_Plus(x_left,y,x_right,y,color);
        x_left++;
        x_right--;
    }
}

```

12. help.c

```

#include"headfile.h"

/*****
*****
* @author      ytm
* @date        2022-4-12
*****
*****/

//driver
#define Revert_x0  5
#define Revert_y0  5

#define Intro_x0    12
#define Intro_y0    70
#define Text_xplus  160
#define Text_yplus  24

#define x_first     429
#define y_first     75
#define y_second    115

```

```

#define y_third      156
#define y_fourth    196
#define y_fifth     237
#define y_sixth     278
#define y_seventh   319
#define y_eighth    360
#define y_ninth     401
#define y_tenth     442
#define y_eleventh  483
#define y_twelfth   524
#define y_thirteenth 565

#define signtitle1_x_first 453 //words2
#define signtitle2_x_first 621 //words2
#define signtitle3_x_first 837 //words4
#define signline1_word1_x 465 //word1:sign
#define signline2_word1_x 633
#define signline2_word2_x 621
#define signline2_word3_x 609
#define signline3_word4_x 837
#define signline3_word6_x 813
#define verbttitle1_x_first 453 //word2
#define verbttitle2_x_first 561 //word2
#define verbttitle3_x_first 681 //word2
#define verbttitle4_x_first 801 //word2
#define verbttitle5_x_first 921 //word2
#define verblines1_word1_x 573 //word1
#define verblines1_word3_x 555 //word3 part16
#define verblines1_word5_x 545 //word5 part16
#define verblines2_word1_x 693 //word1
#define verblines2_word2_x 685 //word2 part16
#define verblines2_word3_x 675 //word3 part16
#define verblines2_word5_x 665 //word5 part16
#define verblines3_word1_x 813 //word1
#define verblines3_word2_x 805 //word2 part16
#define verblines3_word3_x 799 //word3 part16
#define verblines3_word5_x 785 //word5 part16
#define verblines4_word1_x 933 //word1
#define verblines4_word3_x 919 //word3 part16
#define verblines4_word5_x 905 //word5 part16
#define verbattention_x 441 //attention sentence
//verbttitle1_x_first 453 //word2
#define verbweather_x 537 //description

```

```

//follower
#define Revert_x1 (Revert_x0+39)
#define Revert_y1 (Revert_y0+39)

#define Intro_x1 (Intro_x0+383)
#define Intro_y1 (Intro_y0+79)
#define Rule_x0 12
#define Rule_y0 (Intro_y0+85)
#define Rule_x1 (Intro_x0+383)
#define Rule_y1 (Intro_y0+165)
#define Sign_x0 12
#define Sign_y0 (Intro_y0+171)
#define Sign_x1 (Intro_x0+383)
#define Sign_y1 (Intro_y0+251)
#define Verb_x0 12
#define Verb_y0 (Intro_y0+257)
#define Verb_x1 (Intro_x0+383)
#define Verb_y1 (Intro_y0+337)

/*****
*****
* 函数名称      Help
* 函数作用      菜单界面
* 函数输入      page 界面地址
* 函数输出      无
*****
*****/
void Help(int *page)
{
    /*按钮状态变量*/
    int flag,flag_press_last,flag_press,flag_revert,flag_choose_
ign,flag_verb,flag_mouse,flag_mouse_light;
    /*右键菜单变量*/
    int xm1,ym1,xm2,ym2;
Refresh_Help:
    //SetSVGA64k();
    Solid_Bar(0,0,1023,767,BLACK);
    flag=0,flag_press_last=0,flag_press=0,flag_revert=0,flag_choo
se_sign=0,flag_verb=0,flag_mouse=0,flag_mouse_light=0;
    Draw_Help();
    resetMouse(MouseX,MouseY);
    while(1)

```

```

{
    newxy(&MouseX,&MouseY,&press);
    /*刷新界面*/
    if(mouse_press(1,1,1023,767)==3)
    {
        mousehide(MouseX,MouseY);
        if(flag_mouse==0)
        {
            flag_mouse=1;
            /*根据鼠标位置绘制右键菜单*/
            if(MouseX>0&&MouseX<864&&MouseY>0&&MouseY<728)
            {
                xm1=MouseX,ym1=MouseY,xm2=MouseX+160,ym2=Mous
eY+40;
            }
            else if(MouseX>863&&MouseX<1024&&MouseY>0&&MouseY
<728)
            {
                xm1=MouseX-
160,ym1=MouseY,xm2=MouseX,ym2=MouseY+40;
            }
            else if(MouseX>0&&MouseX<864&&MouseY>727&&MouseY<
768)
            {
                xm1=MouseX,ym1=MouseY-
40,xm2=MouseX+160,ym2=MouseY;
            }
            else
            {
                xm1=MouseX-160,ym1=MouseY-
40,xm2=MouseX,ym2=MouseY;
            }
            save_image(xm1,ym1,xm2+3,ym2+3);
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        resetMouse(MouseX,MouseY);
        continue;
    }
    if(flag_mouse==1)
    {
        if(mouse_out_press(xm1,ym1,xm2,ym2)==1)
        {

```

```

        mousehide(MouseX,MouseY);
        flag_mouse=0;
        printf_image(xm1,ym1,xm2+3,ym2+3);
        resetMouse(MouseX,MouseY);
    }
    if(mouse_press(xm1,ym1,xm2,ym2)==2)
    {
        shape=3;
        puthz(xm1+56,ym1+8,24,24,WHITE,"刷新");
        flag_mouse_light=1;
        continue;
    }
    else if(mouse_press(xm1,ym1,xm2,ym2)==1)
    {
        press=0;
        shape=0;
        //refresh page
        delay(100);
        goto Refresh_Help;
    }
    if(flag_mouse_light!=0)
    {
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        flag_mouse_light=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
//button 1-4
if(MouseX>Intro_x0&&MouseX<Intro_x1&&MouseY>Intro_y0&&MouseY<Intro_y1)
{
    if(mouse_press(Intro_x0,Intro_y0,Intro_x1,Intro_y1)==
2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Help(1);
            resetMouse(MouseX,MouseY);

```



```

        flag=1;
    }
    continue;
}
else if(mouse_press(Intro_x0,Intro_y0,Intro_x1,Intro_
y1)==1)
{
    Message_Light_Help(1);
    flag_press_last=flag_press;
    flag_press=1;
    if(flag_press_last!=0)
    {
        Button_Darken_Help(flag_press_last);
    }
    Button_Press_Help(1);
    //hide mouse
    Solid_Bar(Intro_x0,Intro_y1+1,Intro_x1,Rule_y0-
1,BLACK);
    Solid_Bar(Intro_x1+1,Intro_y0,Intro_x1+9,Intro_y1
+15,BLACK);
    Button_Darken_Help(2);
    Button_Darken_Help(3);
    Button_Darken_Help(4);
    //for next newxy will firstly call mousehide whic
h use the bk of light button, must getbk again
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(MouseX>Rule_x0&&MouseX<Rule_x1&&MouseY>Rule_y0&&MouseY
<Rule_y1)
{
    if(mouse_press(Rule_x0,Rule_y0,Rule_x1,Rule_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Help(2);
            resetMouse(MouseX,MouseY);
            flag=2;
        }
    }
}

```

```

        continue;
    }
    else if(mouse_press(Rule_x0,Rule_y0,Rule_x1,Rule_y1)=
=1)
    {
        Message_Light_Help(2);
        flag_press_last=flag_press;
        flag_press=2;
        if(flag_press_last!=0)
        {
            Button_Darken_Help(flag_press_last);
        }
        Button_Press_Help(2);
        //hide mouse
        Solid_Bar(Rule_x0,Rule_y1+1,Rule_x1,Sign_y0-
1,BLACK);
        Solid_Bar(Rule_x1+1,Rule_y0,Rule_x1+9,Rule_y1+15,
BLACK);
        Button_Darken_Help(1);
        Button_Darken_Help(3);
        Button_Darken_Help(4);
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
//button3
if(MouseX>Sign_x0&&MouseX<Sign_x1&&MouseY>Sign_y0&&MouseY
<Sign_y1)
{
    if(mouse_press(Sign_x0,Sign_y0,Sign_x1,Sign_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Help(3);
            resetMouse(MouseX,MouseY);
            flag=3;
        }
        continue;
    }
    else if(mouse_press(Sign_x0,Sign_y0,Sign_x1,Sign_y1)=

```

```

=1)
    {
        flag_press_last=flag_press;
        flag_press=3;
        if(flag_press_last!=0)
        {
            Button_Darken_Help(flag_press_last);
        }
        Button_Darken_Help(1);
        Button_Darken_Help(2);
        Button_Press_Help(3);
        Button_Darken_Help(4);
        //hide mouse
        Solid_Bar(Sign_x1+1,Sign_y0,Sign_x1+9,Sign_y1+15,
BLACK);
        delay(210);
        ChooseButtonSign(Sign_x0,Sign_y0,Sign_x1,Sign_y1,
&flag_choose_sign);
        //refresh button 3-4
        Solid_Bar(Sign_x0-2,Sign_y0-
2,Verb_x1+2,Verb_y1+2,BLACK);
        Button_Draw(Sign_x0,Sign_y0,Sign_x1,Sign_y1,Hollo
w_Color,Solid_Color);
        puthz(Sign_x0+Text_xplus,Sign_y0+Text_yplus,32,32
,BLACK,"图像");
        TriangleImage(Sign_x0+311,Sign_y0+31,DARK_GRAY);
        Button_Draw(Verb_x0,Verb_y0,Verb_x1,Verb_y1,Hollo
w_Color,Solid_Color);
        puthz(Verb_x0+Text_xplus,Verb_y0+Text_yplus,32,32
,BLACK,"速度");
        //display message
        MessageSign_Light_Help(flag_choose_sign);
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(flag_press==3&&flag_choose_sign==4)
{
    //children
    if(mouse_press(verbtitle2_x_first-
8,y_second+10,verbtitle2_x_first+56,y_second+74)==2)
    {

```

```

        shape=3;
        flag_verb=1;
        //lighten button
        AnimationEdge_Draw(verbtitle2_x_first-
8,y_second+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtitle2_x_first-
8,y_second+10,verbtitle2_x_first+56,y_second+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtitle2_x_first-
8,y_second+10,MOVE_LEFT,WALK,CHILD,RED);
        Animate_Help(verbtitle2_x_first-
8,y_second+10,MOVE_UP,WALK,CHILD,RED);
        Animate_Help(verbtitle2_x_first-
8,y_second+10,MOVE_RIGHT,WALK,CHILD,RED);
        resetMouse(MouseX,MouseY);
        continue;
    }
    if(mouse_press(verbtitle2_x_first-
8,y_fourth+10,verbtitle2_x_first+56,y_fourth+74)==2)
    {
        shape=3;
        flag_verb=2;
        //lighten button
        AnimationEdge_Draw(verbtitle2_x_first-
8,y_fourth+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtitle2_x_first-
8,y_fourth+10,verbtitle2_x_first+56,y_fourth+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtitle2_x_first-
8,y_fourth+10,MOVE_LEFT,RUN,CHILD,RED);
        Animate_Help(verbtitle2_x_first-
8,y_fourth+10,MOVE_UP,RUN,CHILD,RED);
        Animate_Help(verbtitle2_x_first-
8,y_fourth+10,MOVE_RIGHT,RUN,CHILD,RED);
        resetMouse(MouseX,MouseY);
    }

```

```

        continue;
    }
    if(mouse_press(verbtittle2_x_first-
8,y_sixth+10,verbtittle2_x_first+56,y_sixth+74)==2)
    {
        shape=3;
        flag_verb=3;
        //lighten button
        AnimationEdge_Draw(verbtittle2_x_first-
8,y_sixth+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtittle2_x_first-
8,y_sixth+10,verbtittle2_x_first+56,y_sixth+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtittle2_x_first-
8,y_sixth+10,MOVE_LEFT,CLIMB,CHILD,RED);
        Animate_Help(verbtittle2_x_first-
8,y_sixth+10,MOVE_UP,CLIMB,CHILD,RED);
        Animate_Help(verbtittle2_x_first-
8,y_sixth+10,MOVE_RIGHT,CLIMB,CHILD,RED);
        resetMouse(MouseX,MouseY);
        continue;
    }
    //young
    if(mouse_press(verbtittle3_x_first-
8,y_second+10,verbtittle3_x_first+56,y_second+74)==2)
    {
        shape=3;
        flag_verb=4;
        //lighten button
        AnimationEdge_Draw(verbtittle3_x_first-
8,y_second+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtittle3_x_first-
8,y_second+10,verbtittle3_x_first+56,y_second+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtittle3_x_first-

```

```

8,y_second+10,MOVE_LEFT,WALK,TEEN,YELLOW);
    Animate_Help(verbtitle3_x_first-
8,y_second+10,MOVE_UP,WALK,TEEN,YELLOW);
    Animate_Help(verbtitle3_x_first-
8,y_second+10,MOVE_RIGHT,WALK,TEEN,YELLOW);
    resetMouse(MouseX,MouseY);
    continue;
}
if(mouse_press(verbtitle3_x_first-
8,y_fourth+10,verbtitle3_x_first+56,y_fourth+74)==2)
{
    shape=3;
    flag_verb=5;
    //lighten button
    AnimationEdge_Draw(verbtitle3_x_first-
8,y_fourth+10,WHITE);
    continue;
}
else if(mouse_press(verbtitle3_x_first-
8,y_fourth+10,verbtitle3_x_first+56,y_fourth+74)==1)
{
    mousehide(MouseX,MouseY);
    //animation
    Animate_Help(verbtitle3_x_first-
8,y_fourth+10,MOVE_LEFT,RUN,TEEN,YELLOW);
    Animate_Help(verbtitle3_x_first-
8,y_fourth+10,MOVE_UP,RUN,TEEN,YELLOW);
    Animate_Help(verbtitle3_x_first-
8,y_fourth+10,MOVE_RIGHT,RUN,TEEN,YELLOW);
    resetMouse(MouseX,MouseY);
    continue;
}
if(mouse_press(verbtitle3_x_first-
8,y_sixth+10,verbtitle3_x_first+56,y_sixth+74)==2)
{
    shape=3;
    flag_verb=6;
    //lighten button
    AnimationEdge_Draw(verbtitle3_x_first-
8,y_sixth+10,WHITE);
    continue;
}
else if(mouse_press(verbtitle3_x_first-

```

```

8,y_sixth+10,verbtittle3_x_first+56,y_sixth+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtittle3_x_first-
8,y_sixth+10,MOVE_LEFT,CLIMB,TEEN,YELLOW);
        Animate_Help(verbtittle3_x_first-
8,y_sixth+10,MOVE_UP,CLIMB,TEEN,YELLOW);
        Animate_Help(verbtittle3_x_first-
8,y_sixth+10,MOVE_RIGHT,CLIMB,TEEN,YELLOW);
        resetMouse(MouseX,MouseY);
        continue;
    }
    //middle-age
    if(mouse_press(verbtittle4_x_first-
8,y_second+10,verbtittle4_x_first+56,y_second+74)==2)
    {
        shape=3;
        flag_verb=7;
        //lighten button
        AnimationEdge_Draw(verbtittle4_x_first-
8,y_second+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtittle4_x_first-
8,y_second+10,verbtittle4_x_first+56,y_second+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtittle4_x_first-
8,y_second+10,MOVE_LEFT,WALK,MID,GREEN);
        Animate_Help(verbtittle4_x_first-
8,y_second+10,MOVE_UP,WALK,MID,GREEN);
        Animate_Help(verbtittle4_x_first-
8,y_second+10,MOVE_RIGHT,WALK,MID,GREEN);
        resetMouse(MouseX,MouseY);
        continue;
    }
    if(mouse_press(verbtittle4_x_first-
8,y_fourth+10,verbtittle4_x_first+56,y_fourth+74)==2)
    {
        shape=3;
        flag_verb=8;

```

```

        //lighten button
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_fourth+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtitle4_x_first-
8,y_fourth+10,verbtitle4_x_first+56,y_fourth+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtitle4_x_first-
8,y_fourth+10,MOVE_LEFT,RUN,MID,GREEN);
        Animate_Help(verbtitle4_x_first-
8,y_fourth+10,MOVE_UP,RUN,MID,GREEN);
        Animate_Help(verbtitle4_x_first-
8,y_fourth+10,MOVE_RIGHT,RUN,MID,GREEN);
        resetMouse(MouseX,MouseY);
        continue;
    }
    if(mouse_press(verbtitle4_x_first-
8,y_sixth+10,verbtitle4_x_first+56,y_sixth+74)==2)
    {
        shape=3;
        flag_verb=9;
        //lighten button
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_sixth+10,WHITE);
        continue;
    }
    else if(mouse_press(verbtitle4_x_first-
8,y_sixth+10,verbtitle4_x_first+56,y_sixth+74)==1)
    {
        mousehide(MouseX,MouseY);
        //animation
        Animate_Help(verbtitle4_x_first-
8,y_sixth+10,MOVE_LEFT,CLIMB,MID,GREEN);
        Animate_Help(verbtitle4_x_first-
8,y_sixth+10,MOVE_UP,CLIMB,MID,GREEN);
        Animate_Help(verbtitle4_x_first-
8,y_sixth+10,MOVE_RIGHT,CLIMB,MID,GREEN);
        resetMouse(MouseX,MouseY);
        continue;
    }
}

```



```

        //old
        if(mouse_press(verbtittle5_x_first-
8,y_second+10,verbtittle5_x_first+56,y_second+74)==2)
        {
            shape=3;
            flag_verb=10;
            //lighten button
            AnimationEdge_Draw(verbtittle5_x_first-
8,y_second+10,WHITE);
            continue;
        }
        else if(mouse_press(verbtittle5_x_first-
8,y_second+10,verbtittle5_x_first+56,y_second+74)==1)
        {
            mousehide(MouseX,MouseY);
            //animation
            Animate_Help(verbtittle5_x_first-
8,y_second+10,MOVE_LEFT,WALK,OLD,SKY_BLUE);
            Animate_Help(verbtittle5_x_first-
8,y_second+10,MOVE_UP,WALK,OLD,SKY_BLUE);
            Animate_Help(verbtittle5_x_first-
8,y_second+10,MOVE_RIGHT,WALK,OLD,SKY_BLUE);
            resetMouse(MouseX,MouseY);
            continue;
        }
        if(mouse_press(verbtittle5_x_first-
8,y_fourth+10,verbtittle5_x_first+56,y_fourth+74)==2)
        {
            shape=3;
            flag_verb=11;
            //lighten button
            AnimationEdge_Draw(verbtittle5_x_first-
8,y_fourth+10,WHITE);
            continue;
        }
        else if(mouse_press(verbtittle5_x_first-
8,y_fourth+10,verbtittle5_x_first+56,y_fourth+74)==1)
        {
            mousehide(MouseX,MouseY);
            //animation
            Animate_Help(verbtittle5_x_first-
8,y_fourth+10,MOVE_LEFT,RUN,OLD,SKY_BLUE);
            Animate_Help(verbtittle5_x_first-

```

```

8,y_fourth+10,MOVE_UP,RUN,OLD,SKY_BLUE);
    Animate_Help(verbtitle5_x_first-
8,y_fourth+10,MOVE_RIGHT,RUN,OLD,SKY_BLUE);
    resetMouse(MouseX,MouseY);
    continue;
}
if(mouse_press(verbtitle5_x_first-
8,y_sixth+10,verbtitle5_x_first+56,y_sixth+74)==2)
{
    shape=3;
    flag_verb=12;
    //lighten button
    AnimationEdge_Draw(verbtitle5_x_first-
8,y_sixth+10,WHITE);
    continue;
}
else if(mouse_press(verbtitle5_x_first-
8,y_sixth+10,verbtitle5_x_first+56,y_sixth+74)==1)
{
    mousehide(MouseX,MouseY);
    //animation
    Animate_Help(verbtitle5_x_first-
8,y_sixth+10,MOVE_LEFT,CLIMB,OLD,SKY_BLUE);
    Animate_Help(verbtitle5_x_first-
8,y_sixth+10,MOVE_UP,CLIMB,OLD,SKY_BLUE);
    Animate_Help(verbtitle5_x_first-
8,y_sixth+10,MOVE_RIGHT,CLIMB,OLD,SKY_BLUE);
    resetMouse(MouseX,MouseY);
    continue;
}
//darken button
if(flag_verb!=0)
{
    mousehide(MouseX,MouseY);
    Button_Darken_Verb(flag_verb);
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    flag_verb=0;
}
}

```

```

        //reset mouse
        if(shape!=0)
        {
            shape=0;
        }
    }
    //button 4
    if(MouseX>Verb_x0&&MouseX<Verb_x1&&MouseY>Verb_y0&&MouseY
&ltVerb_y1)
    {
        if(mouse_press(Verb_x0,Verb_y0,Verb_x1,Verb_y1)==2)
        {
            shape=3;
            if(flag==0);
            {
                mousehide(MouseX,MouseY);
                Button_Light_Help(4);
                resetMouse(MouseX,MouseY);
                flag=4;
            }
            continue;
        }
        else if(mouse_press(Verb_x0,Verb_y0,Verb_x1,Verb_y1)=
=1)
        {
            Message_Light_Help(4);
            flag_press_last=flag_press;
            flag_press=4;
            if(flag_press_last!=0)
            {
                Button_Darken_Help(flag_press_last);
            }
            Button_Press_Help(4);
            //hide mouse
            Solid_Bar(Verb_x0,Verb_y1+1,Verb_x1,Verb_y1+15,BL
ACK);

            Solid_Bar(Verb_x1+1,Verb_y0,Verb_x1+9,Verb_y1+15,
BLACK);

            Button_Darken_Help(1);
            Button_Darken_Help(2);
            Button_Darken_Help(3);
            resetMouse(MouseX,MouseY);
            delay(100);

```

```

        continue;
    }
}
if(flag!=0)
{
    if(flag!=flag_press)
    {
        mousehide(MouseX,MouseY);
        Button_Darken_Help(flag);
        resetMouse(MouseX,MouseY);
    }
    else
    {
        mousehide(MouseX,MouseY);
        Button_Press_Help(flag);
        resetMouse(MouseX,MouseY);
    }
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    flag=0;
}
//button revert
if(MouseX>Revert_x0&&MouseX<Revert_x1&&MouseY>Revert_y0&&
MouseY<Revert_y1)
{
    if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Revert_y
1)==2)
    {
        shape=3;
        if(flag_revert==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Help(6);
            resetMouse(MouseX,MouseY);
            flag_revert=6;
        }
        continue;
    }
    else if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Rev
ert_y1)==1)

```

```

        {
            *page=0;
            //go to menu page
            delay(100);
            break;
        }
    }
    if(flag_revert!=0)
    {
        mousehide(MouseX,MouseY);
        Button_Darken_Help(flag_revert);
        if(flag_mouse==1)
        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        resetMouse(MouseX,MouseY);
        flag_revert=0;
    }
    //reset mouse
    if(shape!=0)
    {
        shape=0;
    }
}

/*****
*****
* 函数名称      Draw_Help
* 函数作用      绘制关于界面
* 函数输入      无
* 函数输出      无
*****
*****/

void Draw_Help()
{
    //upperBar: 1024*50 DARK_GRAY + LIGHT_GRAY
    UpperBar_Draw(50,4,LIGHT_GRAY,DARK_GRAY);

    //BoundaryBar: 410-413 DARK_GRAY
    Boundary_Draw(410,53,4,DARK_GRAY);

```

```

        //button 1-4: 384*86 DARK_GRAY + LIGHT_GRAY
        Button_Draw(Intro_x0,Intro_y0,Intro_x1,Intro_y1,Hollow_Color,
Solid_Color);
        puthz(Intro_x0+Text_xplus,Intro_y0+Text_yplus,32,32,BLACK,"介
绍");

        Button_Draw(Rule_x0,Rule_y0,Rule_x1,Rule_y1,Hollow_Color,Soli
d_Color);
        puthz(Rule_x0+Text_xplus,Rule_y0+Text_yplus,32,32,BLACK,"规则
");

        Button_Draw(Sign_x0,Sign_y0,Sign_x1,Sign_y1,Hollow_Color,Soli
d_Color);
        puthz(Sign_x0+Text_xplus,Sign_y0+Text_yplus,32,32,BLACK,"图像
");
        TriangleImage(Sign_x0+311,Sign_y0+31,DARK_GRAY);

        Button_Draw(Verb_x0,Verb_y0,Verb_x1,Verb_y1,Hollow_Color,Soli
d_Color);
        puthz(Verb_x0+Text_xplus,Verb_y0+Text_yplus,32,32,BLACK,"速度
");

        //buttons 5: 40*40 BLACK + LIGHT_GRAY
        RevertImage_Draw(25,25,BLACK);
        puthz(50,9,32,32,BLACK,"帮助");
    }

/*****
*****
* 函数名称      Button_Light_Help
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Light_Help(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Intro_x0,Intro_y0,Intro_x1,Intro_y1,W
HITE);

```

```

Solid_Bar(Intro_x0,Intro_y0,Intro_x1,Intro_y1,YEL
LOW);
    puthz(Intro_x0+Text_xplus,Intro_y0+Text_yplus,32,
32,GRAY,"介绍");
    break;
}
case 2:
{
    Button_Edge(Rule_x0,Rule_y0,Rule_x1,Rule_y1,WHITE
);
    Solid_Bar(Rule_x0,Rule_y0,Rule_x1,Rule_y1,YELLOW)
;
    puthz(Rule_x0+Text_xplus,Rule_y0+Text_yplus,32,32
,GRAY,"规则");
    break;
}
case 3:
{
    Button_Edge(Sign_x0,Sign_y0,Sign_x1,Sign_y1,WHITE
);
    Solid_Bar(Sign_x0,Sign_y0,Sign_x1,Sign_y1,YELLOW)
;
    puthz(Sign_x0+Text_xplus,Sign_y0+Text_yplus,32,32
,GRAY,"图像");
    TriangleImage(Sign_x0+311,Sign_y0+31,DARK_GRAY);
    break;
}
case 4:
{
    Button_Edge(Verb_x0,Verb_y0,Verb_x1,Verb_y1,WHITE
);
    Solid_Bar(Verb_x0,Verb_y0,Verb_x1,Verb_y1,YELLOW)
;
    puthz(Verb_x0+Text_xplus,Verb_y0+Text_yplus,32,32
,GRAY,"速度");
    break;
}
case 6:
{
    RevertButton_Draw(25,25,DARK_GRAY);
    RevertImage_Draw(25,25,WHITE);
    //revert button on
    break;
}

```

```

    }
}

/*****
*****
* 函数名称      Button_Darken_Help
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Darken_Help(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Intro_x0,Intro_y0,Intro_x1,Intro_y1,BLACK);
            Button_Draw(Intro_x0,Intro_y0,Intro_x1,Intro_y1,Hollow_Color,Solid_Color);
            puthz(Intro_x0+Text_xplus,Intro_y0+Text_yplus,32,32,BLACK,"介绍");
            break;
        }
        case 2:
        {
            Button_Edge(Rule_x0,Rule_y0,Rule_x1,Rule_y1,BLACK);
            Button_Draw(Rule_x0,Rule_y0,Rule_x1,Rule_y1,Hollow_Color,Solid_Color);
            puthz(Rule_x0+Text_xplus,Rule_y0+Text_yplus,32,32,BLACK,"规则");
            break;
        }
        case 3:
        {
            Button_Edge(Sign_x0,Sign_y0,Sign_x1,Sign_y1,BLACK);
            Button_Draw(Sign_x0,Sign_y0,Sign_x1,Sign_y1,Hollow_Color,Solid_Color);
            puthz(Sign_x0+Text_xplus,Sign_y0+Text_yplus,32,32

```



```

, BLACK, "图像");
        TriangleImage(Sign_x0+311, Sign_y0+31, DARK_GRAY);
        break;
    }
    case 4:
    {
        Button_Edge(Verb_x0, Verb_y0, Verb_x1, Verb_y1, BLACK
);
        Button_Draw(Verb_x0, Verb_y0, Verb_x1, Verb_y1, Hollow_Color, Solid_Color);
        puthz(Verb_x0+Text_xplus, Verb_y0+Text_yplus, 32, 32
, BLACK, "速度");
        break;
    }
    case 6:
    {
        RevertButton_Draw(25, 25, LIGHT_GRAY);
        RevertImage_Draw(25, 25, BLACK);
        //revert button off
        break;
    }
}
}
}

```

```

/*****
*****
* 函数名称      Button_Press_Help
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数注意      上方状态栏右边文字中心x 值: 719, 故上方状态栏右边文字位置: (687,9)(2 个字), (639,9)(5 个字)
* 函数输出      无
*****
*****/

```

```

void Button_Press_Help(int flag)
{
    Solid_Bar(639, 9, 799, 41, LIGHT_GRAY);
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Intro_x0, Intro_y0, Intro_x1, Intro_y1, BLACK);

```

```

        Button_Draw(Intro_x0,Intro_y0,Intro_x1,Intro_y1,LIGHT_GRAY,GRAY);
        puthz(Intro_x0+Text_xplus,Intro_y0+Text_yplus,32,32,BLACK,"介绍");
        puthz(687,9,32,32,BLACK,"介绍");
        break;
    }
    case 2:
    {
        Button_Edge(Rule_x0,Rule_y0,Rule_x1,Rule_y1,BLACK);
        Button_Draw(Rule_x0,Rule_y0,Rule_x1,Rule_y1,LIGHT_GRAY,GRAY);
        puthz(Rule_x0+Text_xplus,Rule_y0+Text_yplus,32,32,BLACK,"规则");
        puthz(687,9,32,32,BLACK,"规则");
        break;
    }
    case 3:
    {
        Button_Edge(Sign_x0,Sign_y0,Sign_x1,Sign_y1,BLACK);
        Button_Draw(Sign_x0,Sign_y0,Sign_x1,Sign_y1,LIGHT_GRAY,GRAY);
        puthz(Sign_x0+Text_xplus,Sign_y0+Text_yplus,32,32,BLACK,"图像");
        TriangleImage(Sign_x0+311,Sign_y0+31,DARK_GRAY);
        puthz(687,9,32,32,BLACK,"图像");
        break;
    }
    case 4:
    {
        Button_Edge(Verb_x0,Verb_y0,Verb_x1,Verb_y1,BLACK);
        Button_Draw(Verb_x0,Verb_y0,Verb_x1,Verb_y1,LIGHT_GRAY,GRAY);
        puthz(Verb_x0+Text_xplus,Verb_y0+Text_yplus,32,32,BLACK,"速度");
        puthz(687,9,32,32,BLACK,"速度");
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      Message_Light_Help
* 函数作用      显示帮助信息
* 函数输入      flag 显示信息序号
* 函数输出      无
*****
*****/
void Message_Light_Help(int flag)
{
    //message box refresh
    Solid_Bar(414,54,1023,767,BLACK);

    switch(flag)
    {
        case 1:
        {
            //message box1: 610*716 Introduction
            puthz(x_first+46,y_first,24,23,WHITE,"定向越野是一种借助地图、指北针或其他导航工具，");
            puthz(x_first,y_second,24,24,WHITE,"在一个设定的范围内，通过途中的各种障碍，快速到");
            puthz(x_first,y_third,24,24,WHITE,"达各个目标点位，并且完成各个点位任务，最后到达");
            puthz(x_first,y_fourth,24,24,WHITE,"终点点位的运动。");

            break;
        }
        case 2:
        {
            //message box2: 610*716 Rule
            puthz(x_first+48,y_first,24,24,WHITE,"第一、仅能携带标准地图与指北针。");
            puthz(x_first+48,y_second,24,24,WHITE,"第二、不得游泳通过水域。");
            puthz(x_first+48,y_third,24,24,WHITE,"第三、必须依照顺序寻找各个检查点。");
            puthz(x_first+48,y_fourth,24,24,WHITE,"第四、按照完成时间长短来确定名次。");

            break;
        }
        case 4:

```

```

        {
            //graph1
            puthz(verbtitle1_x_first,y_first,24,24,WHITE,"数
值");

            //draw people
            puthz(verbtitle2_x_first+15,y_first,24,24,WHITE,"
少年");

            SignEdge_Draw(verbtitle2_x_first-
15,y_first,GRAY);
            //draw people:24*24 left_up:(verbtitle2_x_first-
15,y_first)
            Draw_Child_Icon(verbtitle2_x_first-
15,y_first,YELLOW);

            puthz(verbtitle3_x_first+15,y_first,24,24,WHITE,"
青年");

            SignEdge_Draw(verbtitle3_x_first-
15,y_first,GRAY);
            //draw people:24*24 left_up:(verbtitle3_x_first-
15,y_first)
            Draw_Teen_Icon(verbtitle3_x_first-
15,y_first,RED);

            puthz(verbtitle4_x_first+15,y_first,24,24,WHITE,"
中年");

            SignEdge_Draw(verbtitle4_x_first-
15,y_first,GRAY);
            //draw people:24*24 left_up:(verbtitle4_x_first-
15,y_first)
            Draw_Mid_Icon(verbtitle4_x_first-
15,y_first,GREEN);

            puthz(verbtitle5_x_first+15,y_first,24,24,WHITE,"
老年");

            SignEdge_Draw(verbtitle5_x_first-
15,y_first,GRAY);
            //draw people:24*24 left_up:(verbtitle5_x_first-
15,y_first)
            Draw_Old_Icon(verbtitle5_x_first-
15,y_first,SKY_BLUE);

            puthz(verbtitle1_x_first,y_second,24,24,WHITE,"体
力");

```

```

30");
60");
50");
20");
复");

;
;

路");

;
;

");
;

步");

);
山");

);

);

);

//line & noticification

```

```

Solid_Bar(verbattention_x,y_seventh+11,993,y_seve
nth+13,DARK_GRAY);
puthz(verbattention_x,y_seventh+41,24,24,WHITE,"
数值含义: 速度表示一分钟内能走过的方块数, 跑步");
puthz(verbattention_x,y_eighth+41,24,24,WHITE,"一
格方块消耗对应体力, 走路一格方块恢复对应体力");
Solid_Bar(verbattention_x,y_tenth+11,993,y_tenth+
13,DARK_GRAY);
//gragh2
puthz(verbttitle1_x_first,y_ninth+82,24,24,WHITE,"
天气");
//draw weather
puthz(verbttitle1_x_first+15,y_tenth+82,24,24,WHIT
E,"晴天");
SignEdge_Draw(verbttitle1_x_first-
15,y_tenth+82,GRAY);
//draw weather:24*24 left_up:(verbttitle1_x_first-
15,y_tenth+82)
Draw_Sun(verbttitle1_x_first-
15,y_tenth+82,YELLOW);

puthz(verbweather_x,y_tenth+82,24,22,WHITE,"走路
速度减少五分之一, 跑步、爬山速度减半");
puthz(verbttitle1_x_first+15,y_eleventh+82,24,24,W
HITE,"多云");
SignEdge_Draw(verbttitle1_x_first-
15,y_eleventh+82,GRAY);
//draw weather:24*24 left_up:(verbttitle1_x_first-
15,y_eleventh+82)
Draw_Cloud(verbttitle1_x_first-
15,y_eleventh+82,WHITE);

puthz(verbweather_x,y_eleventh+82,24,24,WHITE,"走
路、跑步、爬山速度不受影响");
puthz(verbttitle1_x_first+15,y_twelfth+82,24,24,WH
ITE,"雨天");
SignEdge_Draw(verbttitle1_x_first-
15,y_twelfth+82,GRAY);
//draw weather:24*24 left_up:(verbttitle1_x_first-
15,y_twelfth+82)
Draw_Rainy(verbttitle1_x_first-
15,y_twelfth+82,WHITE);

```

```

        puthz(verbweather_x,y_twelfth+82,24,24,WHITE,"走路、跑步、爬山速度减少二分之一");
        break;
    }
}
}

/*****
*****
* 函数名称      MessageSign_Light_Help
* 函数作用      显示帮助信息
* 函数输入      flag 显示信息序号
* 函数注意      左上角坐标是(414,54)，最多25个字
* 函数输出      无
*****
*****/
void MessageSign_Light_Help(int flag)
{
    //message box refresh
    Solid_Bar(414,54,1023,767,BLACK);
    //draw title
    if(flag!=4)
    {
        puthz(signtitle1_x_first,y_first,24,24,WHITE,"图例");
        puthz(signtitle2_x_first,y_first,24,24,WHITE,"名称");
        puthz(signtitle3_x_first,y_first,24,24,WHITE,"可通过性");
    }

    //draw message box3
    switch(flag)
    {
        case 1:
        {
            SignEdge_Draw(signline1_word1_x,y_second,GRAY);
            //draw sign
            Draw_Urban_House_Icon(signline1_word1_x,y_second)
;
            puthz(signline2_word3_x,y_second,24,24,WHITE,"建筑物");
            puthz(signline3_word4_x,y_second,24,24,WHITE,"无法通过");
            SignEdge_Draw(signline1_word1_x,y_third,GRAY);
            //draw sign

```

```

        Draw_Urban_Flower(signline1_word1_x,y_third);
        puthz(signline2_word2_x,y_third,24,24,WHITE,"树木
");
        puthz(signline3_word4_x,y_third,24,24,WHITE,"可以
通过");
        SignEdge_Draw(signline1_word1_x,y_fourth,GRAY);
        //draw sign
        Draw_V_Narrow_Highway_Part(signline1_word1_x,y_fo
urth);
        puthz(signline2_word2_x,y_fourth,24,24,WHITE,"公
路");
        puthz(signline3_word4_x,y_fourth,24,24,WHITE,"可
以通过");
        SignEdge_Draw(signline1_word1_x,y_fifth,GRAY);
        //draw sign
        Draw_River_Icon(signline1_word1_x,y_fifth);
        puthz(signline2_word2_x,y_fifth,24,24,WHITE,"河流
");
        puthz(signline3_word4_x,y_fifth,24,24,WHITE,"无法
通过");
        SignEdge_Draw(signline1_word1_x,y_sixth,GRAY);
        //draw sign
        Draw_Urban_Lake(signline1_word1_x,y_sixth);
        puthz(signline2_word2_x,y_sixth,24,24,WHITE,"湖泊
");
        puthz(signline3_word4_x,y_sixth,24,24,WHITE,"无法
通过");
        SignEdge_Draw(signline1_word1_x,y_seventh,GRAY);
        //draw sign
        Draw_Urban_Bridge_Icon(signline1_word1_x,y_sevent
h);
        puthz(signline2_word1_x,y_seventh,24,24,WHITE,"桥
");
        puthz(signline3_word4_x,y_seventh,24,24,WHITE,"可
以通过");
        break;
    }
    case 2:
    {
        SignEdge_Draw(signline1_word1_x,y_second,GRAY);
        //draw sign
        Draw_Suburban_Bush(signline1_word1_x,y_second);
        puthz(signline2_word2_x,y_second,24,24,WHITE,"树

```



```

木");
    puthz(signline3_word4_x,y_second,24,24,WHITE,"可
以通过");
    SignEdge_Draw(signline1_word1_x,y_third,GRAY);
    //draw sign
    Draw_Suburban_Grass(signline1_word1_x,y_third);
    puthz(signline2_word2_x,y_third,24,24,WHITE,"草地
");
    puthz(signline3_word4_x,y_third,24,24,WHITE,"可以
通过");
    SignEdge_Draw(signline1_word1_x,y_fourth,GRAY);
    //draw sign
    Draw_Suburban_House(signline1_word1_x,y_fourth);
    puthz(signline2_word3_x,y_fourth,24,24,WHITE,"建
筑物");
    puthz(signline3_word4_x,y_fourth,24,24,WHITE,"无
法通过");
    SignEdge_Draw(signline1_word1_x,y_fifth,GRAY);
    //draw sign
    Draw_Suburban_Field(signline1_word1_x,y_fifth);
    puthz(signline2_word2_x,y_fifth,24,24,WHITE,"田地
");
    puthz(signline3_word4_x,y_fifth,24,24,WHITE,"可以
通过");
    SignEdge_Draw(signline1_word1_x,y_sixth,GRAY);
    //draw sign
    Draw_Suburban_Path_Part(signline1_word1_x,y_sixth
,1);
    puthz(signline2_word2_x,y_sixth,24,24,WHITE,"小径
");
    puthz(signline3_word4_x,y_sixth,24,24,WHITE,"可以
通过");
    SignEdge_Draw(signline1_word1_x,y_seventh,GRAY);
    //draw sign
    Draw_River_Icon(signline1_word1_x,y_seventh);
    puthz(signline2_word2_x,y_seventh,24,24,WHITE,"河
流");
    puthz(signline3_word4_x,y_seventh,24,24,WHITE,"无
法通过");
    SignEdge_Draw(signline1_word1_x,y_eighth,GRAY);
    //draw sign
    Draw_Narrow_Bridge_Connection(signline1_word1_x,y
_eighth,V_ERTICAL);

```

```

        puthz(signline2_word1_x,y_eighth,24,24,WHITE,"桥
");
        puthz(signline3_word4_x,y_eighth,24,24,WHITE,"可
以通过");
        SignEdge_Draw(signline1_word1_x,y_ninth,GRAY);
        //draw sign
        Draw_V_Narrow_Highway_Part(signline1_word1_x,y_ni
nth);
        puthz(signline2_word2_x,y_ninth,24,24,WHITE,"公路
");
        puthz(signline3_word4_x,y_ninth,24,24,WHITE,"可以
通过");
        break;
    }
    case 3:
    {
        SignEdge_Draw(signline1_word1_x,y_second,GRAY);
        //draw sign
        Draw_Mountain(signline1_word1_x,y_second);
        puthz(signline2_word1_x,y_second,24,24,WHITE,"山
");
        puthz(signline3_word4_x,y_second,24,24,WHITE,"可
以通过");
        SignEdge_Draw(signline1_word1_x,y_third,GRAY);
        //draw sign
        Draw_Mountainous_Bush(signline1_word1_x,y_third);
        puthz(signline2_word2_x,y_third,24,24,WHITE,"树木
");
        puthz(signline3_word4_x,y_third,24,24,WHITE,"可以
通过");
        SignEdge_Draw(signline1_word1_x,y_fourth,GRAY);
        //draw sign
        Draw_Mountainous_Grass(signline1_word1_x,y_fourth
);
        puthz(signline2_word2_x,y_fourth,24,24,WHITE,"草
地");
        puthz(signline3_word4_x,y_fourth,24,24,WHITE,"可
以通过");
        SignEdge_Draw(signline1_word1_x,y_fifth,GRAY);
        //draw sign
        Draw_River_Icon(signline1_word1_x,y_fifth);
        puthz(signline2_word2_x,y_fifth,24,24,WHITE,"河流
");

```

```

        puthz(signline3_word4_x,y_fifth,24,24,WHITE,"无法
通过");

        SignEdge_Draw(signline1_word1_x,y_sixth,GRAY);
        //draw sign
        Draw_Mountainous_Lake(signline1_word1_x,y_sixth);
        puthz(signline2_word2_x,y_sixth,24,24,WHITE,"湖泊
");

        puthz(signline3_word4_x,y_sixth,24,24,WHITE,"无法
通过");

        SignEdge_Draw(signline1_word1_x,y_seventh,GRAY);
        //draw sign
        Draw_Mountainous_Swamp(signline1_word1_x,y_sevent
h);

        puthz(signline2_word3_x,y_seventh,24,24,WHITE,"沼
泽地");

        puthz(signline3_word6_x,y_seventh,24,24,WHITE,"雨
天无法通过");

        SignEdge_Draw(signline1_word1_x,y_eighth,GRAY);
        //draw sign
        Draw_Narrow_Bridge_Connection(signline1_word1_x,y
_eighth,V_ERTICAL);
        puthz(signline2_word1_x,y_eighth,24,24,WHITE,"桥
");

        puthz(signline3_word4_x,y_eighth,24,24,WHITE,"可
以通过");

        SignEdge_Draw(signline1_word1_x,y_ninth,GRAY);
        //draw sign
        Draw_Mountainous_House(signline1_word1_x,y_ninth)
;

        puthz(signline2_word2_x,y_ninth,24,24,WHITE,"小屋
");

        puthz(signline3_word4_x,y_ninth,24,24,WHITE,"无法
通过");

        break;
    }
    case 4:
    {
        puthz(verbtitle1_x_first,y_first,24,24,WHITE,"动
画");

        puthz(verbtitle1_x_first,y_second+30,24,24,WHITE,
"走路");

        puthz(verbtitle1_x_first,y_fourth+30,24,24,WHITE,
"跑步");

```

```

        puthz(verbtitle1_x_first,y_sixth+30,24,24,WHITE,"
爬山");
        puthz(verbtitle2_x_first,y_first,24,24,WHITE,"少
年");
        AnimationEdge_Draw(verbtitle2_x_first-
8,y_second+10,GRAY);
        AnimationEdge_Draw(verbtitle2_x_first-
8,y_fourth+10,GRAY);
        AnimationEdge_Draw(verbtitle2_x_first-
8,y_sixth+10,GRAY);
        puthz(verbtitle3_x_first,y_first,24,24,WHITE,"青
年");
        AnimationEdge_Draw(verbtitle3_x_first-
8,y_second+10,GRAY);
        AnimationEdge_Draw(verbtitle3_x_first-
8,y_fourth+10,GRAY);
        AnimationEdge_Draw(verbtitle3_x_first-
8,y_sixth+10,GRAY);
        puthz(verbtitle4_x_first,y_first,24,24,WHITE,"中
年");
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_second+10,GRAY);
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_fourth+10,GRAY);
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_sixth+10,GRAY);
        puthz(verbtitle5_x_first,y_first,24,24,WHITE,"老
年");
        AnimationEdge_Draw(verbtitle5_x_first-
8,y_second+10,GRAY);
        AnimationEdge_Draw(verbtitle5_x_first-
8,y_fourth+10,GRAY);
        AnimationEdge_Draw(verbtitle5_x_first-
8,y_sixth+10,GRAY);
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      Button_Darken_Verb
* 函数作用      熄灭按钮

```

```

* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Darken_Verb(flag)
{
    switch(flag)
    {
        case 1:
        {
            AnimationEdge_Draw(verbttitle2_x_first-
8,y_second+10,GRAY);
            break;
        }
        case 2:
        {
            AnimationEdge_Draw(verbttitle2_x_first-
8,y_fourth+10,GRAY);
            break;
        }
        case 3:
        {
            AnimationEdge_Draw(verbttitle2_x_first-
8,y_sixth+10,GRAY);
            break;
        }
        case 4:
        {
            AnimationEdge_Draw(verbttitle3_x_first-
8,y_second+10,GRAY);
            break;
        }
        case 5:
        {
            AnimationEdge_Draw(verbttitle3_x_first-
8,y_fourth+10,GRAY);
            break;
        }
        case 6:
        {
            AnimationEdge_Draw(verbttitle3_x_first-
8,y_sixth+10,GRAY);
            break;
        }
    }
}

```

```

        }
    case 7:
    {
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_second+10,GRAY);
        break;
    }
    case 8:
    {
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_fourth+10,GRAY);
        break;
    }
    case 9:
    {
        AnimationEdge_Draw(verbtitle4_x_first-
8,y_sixth+10,GRAY);
        break;
    }
    case 10:
    {
        AnimationEdge_Draw(verbtitle5_x_first-
8,y_second+10,GRAY);
        break;
    }
    case 11:
    {
        AnimationEdge_Draw(verbtitle5_x_first-
8,y_fourth+10,GRAY);
        break;
    }
    case 12:
    {
        AnimationEdge_Draw(verbtitle5_x_first-
8,y_sixth+10,GRAY);
        break;
    }
}

```

13. hz. c

```
#include"headfile.h"
```

```

/*****

```

```

*****
* @author          unkwown
* @edictor         ytm
* @date            2022-1-15
* @discription     2022-1-24 新增 56,64 大小汉字的支持
* @notice          此版本为 SVGA 64k(24 位) 显示版本( 专门为 SVGA 优
出
*                  若汉字库的文件不存在, 将在按键后等待 6s 后退
*****
*****/

/*****
*****
/*****
*****
* 函数名称          puthz
* 函数作用          输出汉字
* 函数输入          x 输出位置横坐标, y 输出位置纵坐标, flag 输出大小,
part 输出偏移, color 输出颜色, s 输出内容
* 函数输出          无
*****
*****/
void puthz(int x, int y,int flag,int part,int color,char *s)
{
    FILE *hzk_p=NULL;                                // 定
义汉字库文件指针
    unsigned char quma,weima;                        // 定义汉字的区码和位
码
    unsigned long offset;                            // 定义汉字在字
库中的偏移量
    unsigned char mask[] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x
01}; // 功能数组, 用于显示汉字点阵中的亮点
    int i,j,pos;
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L; // 显存指针常量, 指向显存首地址, 指针本身不允许修改
    unsigned char new_page = 0;                      // 要切换的页面号
    unsigned long int page;                          // 对应显存地址偏移量

    switch(flag) // 不同的 flag 对应不同的汉字库, 实现了汉字的大小可
根据需要改变
    {
        case 16 :
            {
                char mat[32]; //16*16 的汉字需要 32 个字节的数

```

组来存储

```
int y0=y;
int x0=x;
hzk_p = fopen(".\\HZK\\HZ16","rb");    //路
径自行修改!

if(hzk_p==NULL)
{
    getch();
    delay(6000);
    exit(1);
}
while(*s!=NULL)
{
    while (x<1024-flag && (*s!=NULL))
    {
        y=y0;
        quma=s[0]-
0xa0;    // 求出区码
        weima=s[1]-
0xa0;    // 求出位码
        offset=(94*(quma-1)+(weima-
1))*32L;    // 求出要显示的汉字在字库文件中的偏移
        fseek(hzk_p,offset,SEEK_SET);

        // 重定位文件指针

        fread (mat,32,1,hzk_p);    //
        读出该汉字的具体点阵数据,1 为要读入的项数

        for(i=0;i<16;i++)
        {
            /* 计算显存地址偏移量和对应的页面号,
            做换页操作*/

            page = ((unsigned long int)y << 1
0) + x;

            new_page = page >> 15;
            SelectPage(new_page);
            pos=2*i;    //16*16 矩阵中有每
            一行有两外字节

            for(j=0;j<16;j++)    // 一行一行地
            扫描, 将位上为了1 的点显示出来

            {
                if((mask[j%8]&mat[pos+j/8])!=
NULL)    //j%8 只能在0-8 之间循环, j/8 在0, 1 之间循环
                {
```



```

                                *(video_buffer + page + j
) = color;

                                }
                                }
                                y++;
                                }
                                /*=====
=====
                                以上是一个汉字显示完
=====
=====*/

                                x+=part;          // 给x 一个偏移量part
                                s+=2;             // 汉字里存放的是内码, 2
个字节, 所以要加2

                                }
                                x=x0;y0+=flag+10; // 一行汉字显示完后, 重新从
左侧开始输出汉字, 给y 一个偏移量, 10 为行间距
                                }
                                break;
                                }
case 24 :
{
    char mat[72];    //24*24 矩阵要 72 个字节来存储
    int y0=y;
    int x0=x;
    hzk_p = fopen(".\\HZK\\Hk24k", "rb");
    if (hzk_p==NULL)
    if(hzk_p==NULL)
    {
        getch();
        delay(6000);
        exit(1);
    }
    while(*s!=NULL)
    {
        while(x<1024-flag && (*s!=NULL))
        {
            y=y0;
            quma=s[0]-
0xa0;          // 求出区码
            weima=s[1]-
0xa0;          // 求出位码
            offset=(94*(quma-1)+(weima-1))*72L;

```

```

fseek(hzk_p,offset,SEEK_SET);
fread (mat,72,1,hzk_p);
for (i=0;i<24;i++)
{
    /* 计算显存地址偏移量和对应的页面号,
做换页操作*/

    page = ((unsigned long int)y << 1
0) + x;

    new_page = page >> 15;
    SelectPage(new_page);
    pos=3*i;    // 矩阵中每一行有三个字节
    for (j=0;j<24;j++)    // 每一行有
24 位

    {
        if ((mask[j%8]&mat[pos+j/8])!
=NULL)

        {
            *(video_buffer + page + j
) = color;

        }
        y++;
    }
    x+=part;
    s+=2;
}
x=x0;y0+=flag+10;
}
break;
}
case 32 :
{
    char mat[128];    //32*32 的汉字需要 128 个字节
的数组来存储

    int y0=y;
    int x0=x;
    hzk_p = fopen(".\\HZK\\HZK32S","rb");
    if(hzk_p==NULL)
    {
        getch();
        delay(6000);
        exit(1);
    }
}

```

```

while(*s!=NULL)
{
    while (x<1024-flag && (*s!=NULL))
    {
        y=y0;
        quma=s[0]-
0xa0;           // 求出区码
        weima=s[1]-
0xa0;           // 求出位码
        offset=(94*(quma-1)+(weima-1))*128L;
        fseek(hzk_p,offset,SEEK_SET);
        fread (mat,128,1,hzk_p);
        for(i=0;i<32;i++)
        {
            /* 计算显存地址偏移量和对应的页面号,
做换页操作*/

            page = ((unsigned long int)y << 1
0) + x;

            new_page = page >> 15;
            SelectPage(new_page);
            pos=4*i;           //32*32 矩阵中有每
一行有两外字节

            for(j=0;j<32;j++)
            {
                if((mask[j%8]&mat[pos+j/8])!=
NULL)

                {
                    *(video_buffer + page + j
) = color;

                }
            }
            y++;
        }
        // 以上是一个汉字显示完
        x+=part;           // 给 x 一个偏移量 part
        s+=2;               // 汉字里存放的是内码, 2
个字节, 所以要加 2

    }
    x=x0;y0+=flag+10;     // 一行汉字显示完后, 给
y 一个偏移量, 10 为行间距
}
break;
}

```

```

case 48:
{
    char mat[288];    //48*48 的汉字需要 288 个字节的
数组来存储

    int y0=y;
    int x0=x;
    hzk_p = fopen(".\\HZK\\Hzk48k","rb");
    if(hzk_p==NULL)
    {
        getch();
        delay(6000);
        exit(1);
    }
    while(*s!=NULL)
    {
        while (x<1024-flag && (*s!=NULL))
        {
            y=y0;
            quma=s[0]-
0xa0;    // 求出区码
            weima=s[1]-
0xa0;    // 求出位码
            offset=(94*(quma-1)+(weima-
1))*288L;    // 求出要显示的汉字在字库文件中的偏移
            fseek(hzk_p,offset,SEEK_SET);

            // 重定位文件指针

            fread (mat,288,1,hzk_p);    /

            // 读出该汉字的具体点阵数据,1 为要读入的项数

            for(i=0;i<48;i++)
            {
                /* 计算显存地址偏移量 and 对应的页面号,
做换页操作*/

                page = ((unsigned long int)y << 1
0) + x;

                new_page = page >> 15;
                SelectPage(new_page);
                pos=6*i;
                for(j=0;j<48;j++)    // 一行一行地
扫描, 将位上为了 1 的点显示出来

                {
                    if((mask[j%8]&mat[pos+j/8])!=
NULL)    //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环

```

```

        {
            *(video_buffer + page + j
) = color;
        }
    }
    y++;
}
// 以上是一个汉字显示完
x+=part;    // 给 x 一个偏移量 part
s+=2;       // 汉字里存放的是内码, 2
个字节, 所以要加 2
}
x=x0;y0+=flag+10;    // 一行汉字显示完后, 给
y 一个偏移量, 10 为行间距
}
break;
}
case 56:
{
    char mat[392];    // 56*56 的汉字需要 392 个字节的数组
来存储

    int y0=y;
    int x0=x;
    hzk_p = fopen(".\\HZK\\HZK56", "rb");
    if(hzk_p==NULL)
    {
        getch();
        delay(6000);
        exit(1);
    }
    while(*s!=NULL)
    {
        while (x<1024-flag && (*s!=NULL))
        {
            y=y0;
            quma=s[0]-0xa0;                // 求
出区码

            weima=s[1]-0xa0;                // 求
出位码

            offset=(94*(quma-1)+(weima-1))*392L;    //
求出要显示的汉字在字库文件中的偏移

            fseek(hzk_p,offset,SEEK_SET);            //
重定位文件指针

```

```

        fread (mat,392,1,hzk_p);           // 读
出该汉字的具体点阵数据,1 为要读入的项数

        for(i=0;i<56;i++)
        {
            /* 计算显存地址偏移量和对应的页面号, 做换
页操作*/

            page = ((unsigned long int)y << 10) +
x;

            new_page = page >> 15;
            SelectPage(new_page);
            pos=7*i;
            for(j=0;j<56;j++)    // 一行一行地扫
描, 将位上为了1 的点显示出来
            {
                if((mask[j%8]&mat[pos+j/8])!=NULL
)    //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环
                {
                    *(video_buffer + page + j) =
color;

                }
            }
            y++;
        }
        // 以上是一个汉字显示完
        x+=part;    // 给 x 一个偏移量 part
        s+=2;        // 汉字里存放的是内码, 2 个字
节, 所以要加 2
    }
    x=x0;y0+=flag+10;    // 一行汉字显示完后, 给 y 一
个偏移量, 10 为行间距
    }
    break;
}
case 64:
{
    char mat[512];    //64*64 的汉字需要 512 个字节的数组
来存储

    int y0=y;
    int x0=x;
    hzk_p = fopen(".\\HZK\\HZK64","rb");
    if(hzk_p==NULL)
    {

```

```

        getch();
        delay(6000);
        exit(1);
    }
    while(*s!=NULL)
    {
        while (x<1024-flag && (*s!=NULL))
        {
            y=y0;
            quma=s[0]-0xa0;                // 求
出区码
            weima=s[1]-0xa0;                // 求
出位码
            offset=(94*(quma-1)+(weima-1))*512L;    //
求出要显示的汉字在字库文件中的偏移
            fseek(hzk_p,offset,SEEK_SET);            //
重定位文件指针
            fread (mat,512,1,hzk_p);                // 读
出该汉字的具体点阵数据,1 为要读入的项数

            for(i=0;i<60;i++)
            {
                /* 计算显存地址偏移量和对应的页面号, 做换
页操作*/
                page = ((unsigned long int)y << 10) +
x;

                new_page = page >> 15;
                SelectPage(new_page);
                pos=8*i;
                for(j=0;j<60;j++)    // 一行一行地扫
描, 将位上为了1 的点显示出来
                {
                    if((mask[j%8]&mat[pos+j/8])!=NULL
)    //j%8 只能在 0-8 之间循环, j/8 在 0, 1 之间循环
                    {
                        *(video_buffer + page + j) =
color;
                    }
                }
                y++;
            }
            // 以上是一个汉字显示完
            x+=part;    // 给 x 一个偏移量 part

```

```

                                s+=2;                // 汉字里存放的是内码, 2 个字
节, 所以要加2
                                }
                                x=x0;y0+=flag+10;    // 一行汉字显示完后, 给y 一
个偏移量, 10 为行间距
                                }
                                break;
                                }
                                default:
                                {
                                    break;
                                }
                                }
                                fclose(hzk_p);
}

```

14. icon.c

```

#include "headfile.h"

/*****
*****

* 函数名称      Draw_Teen
* 函数作用      绘制青年
* 函数输入      x0,y0 坐标, color 衣服颜色
* 函数输出      无
*****
*****/

void Draw_Teen(int x0,int y0,int color)
{
    Solid_Circle(x0+11,y0+4,5,BODYCOLOR);//head
    Solid_Bar(x0+7,y0+1,x0+15,y0+2,RED);
    Putpixel64k(x0+6,y0+2,RED);
    Putpixel64k(x0+16,y0+2,RED);
    Solid_Bar(x0+2,y0+11,x0+4,y0+22,color);//left upper arm
    Solid_Bar(x0+2,y0+24,x0+4,y0+31,BODYCOLOR);//left lower arm
    Solid_Bar(x0+6,y0+11,x0+17,y0+27,color);//body
    Solid_Bar(x0+19,y0+11,x0+21,y0+22,color);//right upper arm
    Solid_Bar(x0+19,y0+24,x0+21,y0+31,BODYCOLOR);//right lower ar
m
    Solid_Bar(x0+6,y0+29,x0+10,y0+38,color);//left leg
    Solid_Bar(x0+13,y0+29,x0+17,y0+38,color);//right leg
    Solid_Bar(x0+6,y0+40,x0+10,y0+47,BODYCOLOR);//left foot
    Solid_Bar(x0+13,y0+40,x0+17,y0+47,BODYCOLOR);//right foot

```



```

}

/*****
*****
* 函数名称      Draw_Old
* 函数作用      绘制老年
* 函数输入      x0,y0 坐标, color 衣服颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Old(int x0,int y0,int color)
{
    Solid_HalfSector_up(x0+11,y0+4,5,GRAY);//hair
    Solid_HalfSector_down(x0+11,y0+4,5,BODYCOLOR);//head
    Solid_Bar(x0+2,y0+11,x0+4,y0+22,color);//Left upper arm
    Solid_Bar(x0+2,y0+24,x0+4,y0+31,BODYCOLOR);//Left Lower arm
    Solid_Bar(x0+6,y0+11,x0+17,y0+27,color);//body
    Solid_Bar(x0+19,y0+11,x0+21,y0+22,color);//right upper arm
    Solid_Bar(x0+19,y0+24,x0+21,y0+31,BODYCOLOR);//right Lower ar
m
    Solid_Bar(x0+6,y0+29,x0+10,y0+38,color);//Left Leg
    Solid_Bar(x0+13,y0+29,x0+17,y0+38,color);//right Leg
    Solid_Bar(x0+6,y0+40,x0+10,y0+47,BODYCOLOR);//Left foot
    Solid_Bar(x0+13,y0+40,x0+17,y0+47,BODYCOLOR);//right foot
}

```

```

/*****
*****
* 函数名称      Draw_Mid
* 函数作用      绘制中年
* 函数输入      x0,y0 坐标, color 衣服颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Mid(int x0,int y0,int color)
{
    Solid_Circle(x0+11,y0+4,5,BODYCOLOR);//head
    /*
    Putpixel64k(x0+9,y0+7,WHITE);
    Putpixel64k(x0+10,y0+7,WHITE);
    Putpixel64k(x0+10,y0+6,WHITE);

```

```

    Putpixel64k(x0+13,y0+6,WHITE);
    Putpixel64k(x0+13,y0+7,WHITE);
    Putpixel64k(x0+14,y0+7,WHITE);
    */
    Solid_Bar(x0+2,y0+11,x0+4,y0+22,color);//left upper arm
    Solid_Bar(x0+2,y0+24,x0+4,y0+31,BODYCOLOR);//left lower arm
    Solid_Bar(x0+6,y0+11,x0+17,y0+27,color);//body
    Solid_Bar(x0+19,y0+11,x0+21,y0+22,color);//right upper arm
    Solid_Bar(x0+19,y0+24,x0+21,y0+31,BODYCOLOR);//right lower ar
m
    Solid_Bar(x0+6,y0+29,x0+10,y0+38,color);//left leg
    Solid_Bar(x0+13,y0+29,x0+17,y0+38,color);//right leg
    Solid_Bar(x0+6,y0+40,x0+10,y0+47,BODYCOLOR);//left foot
    Solid_Bar(x0+13,y0+40,x0+17,y0+47,BODYCOLOR);//right foot
}

```

```

/*****
*****
* 函数名称      Draw_Child
* 函数作用      绘制少年
* 函数输入      x0,y0 坐标, color 衣服颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Child(int x0,int y0,int color)
{
    Solid_Circle(x0+11,y0+11,5,BODYCOLOR);//head
    Solid_Bar(x0+4,y0+18,x0+5,y0+27,color);//left upper arm
    Solid_Bar(x0+4,y0+29,x0+5,y0+35,BODYCOLOR);//left lower arm
    Solid_Bar(x0+7,y0+18,x0+16,y0+31,color);//body
    Solid_Bar(x0+18,y0+18,x0+19,y0+27,color);//right upper arm
    Solid_Bar(x0+18,y0+29,x0+19,y0+35,BODYCOLOR);//right lower ar
m
    Solid_Bar(x0+7,y0+33,x0+10,y0+39,color);//left leg
    Solid_Bar(x0+13,y0+33,x0+16,y0+39,color);//right leg
    Solid_Bar(x0+7,y0+41,x0+10,y0+47,BODYCOLOR);//left foot
    Solid_Bar(x0+13,y0+41,x0+16,y0+47,BODYCOLOR);//right foot
}

```

```

/*****
*****
* 函数名称      Draw_Cloud

```

```

* 函数作用      绘制云朵
* 函数输入      x0,y0 坐标, color 云朵颜色
* 函数输出      无
*****
*****/

void Draw_Cloud(int x0,int y0,int color)
{
    Solid_Circle(x0+8,y0+11,7,color);
    Solid_Circle(x0+18,y0+13,5,color);
    Solid_Bar(x0+8,y0+14,x0+18,y0+18,color);
}

/*****
*****
* 函数名称      Draw_Sun
* 函数作用      绘制太阳
* 函数输入      x0,y0 坐标, color 太阳颜色
* 函数输出      无
*****
*****/

void Draw_Sun(int x0,int y0,int color)
{
    Solid_Circle(x0+12,y0+12,5,color);
    Line_Plus(x0+12,y0+2,x0+12,y0+5,color);
    Line_Plus(x0+12,y0+19,x0+12,y0+22,color);
    Line_Plus(x0+2,y0+12,x0+5,y0+12,color);
    Line_Plus(x0+19,y0+12,x0+22,y0+12,color);
    Line_Plus(x0+4,y0+4,x0+7,y0+7,color);
    Line_Plus(x0+17,y0+17,x0+20,y0+20,color);
    Line_Plus(x0+17,y0+7,x0+20,y0+4,color);
    Line_Plus(x0+4,y0+20,x0+7,y0+17,color);
}

/*****
*****
* 函数名称      Draw_RainDrop
* 函数作用      绘制雨点
* 函数输入      x0,y0 坐标, color 雨滴颜色
* 函数输出      无
*****
*****/

```

```

void Draw_RainDrop(int x0,int y0,int color)
{
    Solid_Bar(x0+4,y0,x0+6,y0+5,color);
    Solid_Bar(x0+10,y0,x0+12,y0+5,color);
    Solid_Bar(x0+16,y0,x0+18,y0+5,color);
}

/*****
*****

* 函数名称      Draw_Rainy
* 函数作用      绘制雨天图标
* 函数输入      x0,y0 坐标, color 图标颜色
* 函数输出      无
*****
*****/

void Draw_Rainy(int x0,int y0,int color)
{
    //Hollow_Bar(x0,y0,x0+23,y0+23,RED);
    Draw_Cloud(x0,y0-3,color);
    Draw_RainDrop(x0+1,y0+17,AQUA_BLUE);
}

/*****
*****

* 函数名称      Draw_Compass
* 函数作用      绘制指南针图标
* 函数输入      x0,y0 坐标
* 函数输出      无
*****
*****/

void Draw_Compass(int x0,int y0)
{
    Solid_Circle(x0+12,y0+12,12,GOLDEN);
    Solid_Circle(x0+12,y0+12,10,BLACK);
    //Hollow_Circle(x0+12,y0+12,11,GOLDEN);
    Solid_Triangle(x0+12,y0+1,x0+10,y0+11,x0+15,y0+11,RED);
    Solid_Triangle(x0+13,y0+22,x0+10,y0+12,x0+15,y0+12,WHITE);
}

/*****
*****

```

```

*****
* 函数名称      Draw_Mountainous_Tree
* 函数作用      绘制山区树木
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Mountainous_Tree(int x,int y)
{
    y+=24;
    randomize();
    Solid_Quadrangle(x+4,y-2,x+19,y-2,x+7,y-16,x+16,y-16,BROWN);
    Solid_Triangle(x,y-17,x+23,y-17,x+11,y-36,DARK_GREEN);
    Solid_Triangle(x+3,y-31,x+19,y-31,x+11,y-48,DARK_GREEN);
    Solid_Triangle(x+5,y-44,x+17,y-44,x+10,y-55,DARK_GREEN);
    //Triangle_Random_Spot(x+5,y-44,x+17,y-44,x+10,y-
55,15,FOREST_GREEN);
}

```

```

/*****
*****
* 函数名称      Draw_Wave
* 函数作用      绘制波浪
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Wave(int x,int y)
{
    Putpixel64k(x,y,SKY_BLUE);
    Putpixel64k(x+1,y-1,SKY_BLUE);
    Putpixel64k(x+2,y-2,SKY_BLUE);
    Putpixel64k(x+3,y-1,SKY_BLUE);
    Putpixel64k(x+4,y,SKY_BLUE);
    Putpixel64k(x+5,y-1,SKY_BLUE);
    Putpixel64k(x+6,y-2,SKY_BLUE);
    Putpixel64k(x+7,y-1,SKY_BLUE);
    Putpixel64k(x+8,y,SKY_BLUE);
}

/*****

```

```

*****
* 函数名称          Draw_Urban_Lake
* 函数作用          绘制城区湖泊
* 函数输入          x,y 坐标
* 函数输出          无
*****
*****/

void Draw_Urban_Lake(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,AZURE);//or use circle
    Draw_Wave(x+4,y+8);
    Draw_Wave(x+12,y+13);
    Draw_Wave(x+6,y+18);
}

/*****
*****
* 函数名称          Draw_Mountainous_Lake
* 函数作用          绘制山区湖泊
* 函数输入          x,y 坐标
* 函数输出          无
*****
*****/

void Draw_Mountainous_Lake(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,BLUE);
    //Solid_Circle(x+11,y+11,12,STEEL_BLUE);
    Draw_Wave(x+4,y+8);
    Draw_Wave(x+12,y+13);
    Draw_Wave(x+6,y+18);
}

/*****
*****
* 函数名称          Draw_Swamp_Wave
* 函数作用          绘制沼泽波浪
* 函数输入          x,y 坐标
* 函数输出          无
*****
*****/

```

```

void Draw_Swamp_Wave(int x,int y)
{
    Putpixel64k(x,y,DARK_GREEN);
    Putpixel64k(x+1,y-1,DARK_GREEN);
    Putpixel64k(x+2,y-2,DARK_GREEN);
    Putpixel64k(x+3,y-1,DARK_GREEN);
    Putpixel64k(x+4,y,DARK_GREEN);
    Putpixel64k(x+5,y-1,DARK_GREEN);
    Putpixel64k(x+6,y-2,DARK_GREEN);
    Putpixel64k(x+7,y-1,DARK_GREEN);
    Putpixel64k(x+8,y,DARK_GREEN);
}

/*****
*****
* 函数名称      Draw_Mountainous_Swamp
* 函数作用      绘制山区沼泽
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Mountainous_Swamp(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,OLIVE_DRAB);
    Bar_Random_Spot(x,y,x+23,y+23,20,BROWN);
    //Solid_Circle(x+11,y+11,12,OLIVE_DRAB);
    //Circle_Random_Spot(x+11,y+11,12,20,BROWN);
    Draw_Swamp_Wave(x+4,y+8);
    Draw_Swamp_Wave(x+12,y+13);
    Draw_Swamp_Wave(x+6,y+18);
    //Circle_Random_Spot(x+11,y+11,12,10,BROWN);
}

/*****
*****
* 函数名称      Draw_River
* 函数作用      绘制河流
* 函数输入      x0,y0,x1,y1 两端点坐标
* 函数输出      无
*****
*****/

```

```

void Draw_River(int x0,int y0,int x1,int y1)
{
    long int xd0,yd0,xd1,yd1,stepx,stepy;
    float k,k0,b1,b0,kc,bc;
    if(x0>x1)
    {
        Long_Diswap(&x0,&x1);
        Long_Diswap(&y0,&y1);
    }
    Solid_Circle(x0+11,y0+11,12,STEEL_BLUE);
    Solid_Circle(x1+11,y1+11,12,STEEL_BLUE);
    if(x0!=x1&&y0!=y1)
    {
        k=-(float)(x1-x0)/(float)(y1-y0);
        kc=(float)(y1-y0)/(float)(x1-x0);
        b0=(float)(y0+11)-k*(x0+11);
        b1=(float)(y1+11)-k*(x1+11);
        bc=(float)(y0+11)-kc*(x0+11);
        xd0=x0+11;
        xd1=x1+11;
        yd0=(long int)(k*xd0+b0);
        yd1=(long int)(k*xd1+b1);
        Line_Plus(xd0,yd0,xd1,yd1,STEEL_BLUE);
        for(xd0=x0+11,xd1=x1+11;((xd0-x0-11)*(xd0-x0-11)+(yd0-y0-
11)*(yd0-y0-11))<=144;xd0--,xd1--)
        {
            yd0=(long int)(k*xd0+b0);
            yd1=(long int)(k*xd1+b1);
        }
        Solid_Quadrangle(xd0,yd0,xd1,yd1,x0+11,y0+11,x1+11,y1+11,
STEEL_BLUE);
        for(xd0=x0+11,xd1=x1+11;((xd0-x0-11)*(xd0-x0-11)+(yd0-y0-
11)*(yd0-y0-11))<=144;xd0++,xd1++)
        {
            yd0=(int)(k*xd0+b0);
            yd1=(int)(k*xd1+b1);
        }
        Solid_Quadrangle(xd0,yd0,xd1,yd1,x0+11,y0+11,x1+11,y1+11,
STEEL_BLUE);
        if(kc>0)
        {
            for(stepx=x0+11,stepy=y0+11;((stepx-x1-11)*(stepx-x1-
11)+(stepy-y1-11)*(stepy-y1-11))>24*24&&abs(stepx-x1-

```



```

11)>6&&abs(stepy-y1-11)>6;stepx+=12,stepy=kc*stepx+bc)
    {
        Draw_Wave(stepx-3,stepy);
    }
}
else
{
    for(stepx=x0+11,stepy=y0+11;((stepx-x1-11)*(stepx-x1-
11)+(stepy-y1-11)*(stepy-y1-11))>24*24&&abs(stepx-x1-
11)>6&&abs(stepy-y1-11)>6;stepx-=12,stepy=kc*stepx+bc)
    {
        Draw_Wave(stepx-3,stepy);
    }
}
if(x0==x1)
{
    if(y0>y1)
    {
        Long_Diswap(&x0,&x1);
        Long_Diswap(&y0,&y1);
    }
    Solid_Bar(x0-1,y0+11,x1+24,y1+11,STEEL_BLUE);
    for(stepx=x0+11,stepy=y0+11;y1-stepy>-8;stepy+=24)
    {
        Draw_Wave(stepx-3,stepy);
    }
}
if(y0==y1)
{
    Solid_Bar(x0+11,y0-1,x1+11,y1+24,STEEL_BLUE);
    for(stepx=x0+11,stepy=y0+11;x1-stepx>-24;stepx+=24)
    {
        Draw_Wave(stepx-3,stepy);
    }
}
}

```

```

/*****
*****

```

```

* 函数名称      Draw_Grass
* 函数作用      绘制草地
* 函数输入      x,y 坐标

```

```

* 函数输出      无
*****
*****/

```

```

void Draw_Grass(int x,int y)
{
    int color=DARK_GREEN;
    Line_Plus(x,y,x+6,y+6,color);
    Line_Plus(x+1,y,x+6,y+5,color);
    Line_Plus(x,y+1,x+5,y+6,color);
    Line_Plus(x+6,y,x,y+6,color);
    Line_Plus(x+5,y,x,y+5,color);
    Line_Plus(x+6,y+1,x+1,y+6,color);
}

```

```

/*****
*****
* 函数名称      Draw_Suburban_Grass
* 函数作用      绘制郊区草地
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Suburban_Grass(int x,int y)
{
    int ran_x,ran_y;
    Solid_Bar(x,y,x+23,y+23,LIME);
    ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;
    y+=ran_y;
    //IVY_GREEN

    Draw_Grass(x+3,y+2);
        ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;
    y+=ran_y;
    Draw_Grass(x+14,y+6);
        ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;

```

```

        y+=ran_y;
        Draw_Grass(x+6,y+14);
    }

/*****
*****
* 函数名称      Draw_Leaf_1
* 函数作用      绘制树叶1
* 函数输入      x,y 坐标, radius 半径, color 颜色
* 函数输出      无
*****
*****/

void Draw_Leaf_1(int x0,int y0,int radius,int color)
{
    int i,j;
    int x,y;
    for(x=x0;x<x0+radius;x++)
    {
        for(y=y0;y<y0+radius;y++)
        {
            if((x-(x0+radius))*(x-(x0+radius))+(y-y0)*(y-
y0)<=radius*radius&&(y-(y0+radius))*(y-(y0+radius))+(x-x0)*(x-
x0)<=radius*radius)
            {
                Putpixel64k(x,y,color);
            }
        }
    }
    //Line_Plus(x0,y0,x0+radius,y0+radius,DARK_GREEN);
}

/*****
*****
* 函数名称      Draw_Leaf_2
* 函数作用      绘制树叶2
* 函数输入      x,y 坐标, radius 半径, color 颜色
* 函数输出      无
*****
*****/

void Draw_Leaf_2(int x0,int y0,int radius,int color)
{

```

```

    int i,j;
    int x,y;
    for(x=x0;x<x0+radius;x++)
    {
        for(y=y0;y<y0+radius;y++)
        {
            if((x-(x0+radius))*(x-(x0+radius))+(y-
(y0+radius))*(y-(y0+radius))<=radius*radius&&(y-y0)*(y-y0)+(x-
x0)*(x-x0)<=radius*radius)
            {
                Putpixel64k(x,y,color);
            }
        }
    }
    //Line_Plus(x0,y0,x0+radius,y0+radius,DARK_GREEN);
}

/*****
*****
* 函数名称      Draw_Mountainous_Grass
* 函数作用      绘制山区草地
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Mountainous_Grass(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,LIME);
    Solid_Bar(x+4,y+4,x+5,y+6,GRAY);
    Putpixel64k(x+3,y+5,GRAY);
    Putpixel64k(x+6,y+5,GRAY);//1
    Solid_Bar(x+18,y+18,x+20,y+19,YELLOW_GREEN);
    Putpixel64k(x+19,y+17,GRAY);//2
    Solid_Bar(x+13,y+9,x+19,y+12,GREEN);
    Solid_Bar(x+17,y+7,x+19,y+8,GREEN);
    Solid_Bar(x+17,y+4,x+18,y+6,GREEN);
    Solid_Bar(x+20,y+11,x+21,y+12,GREEN);
    Putpixel64k(x+12,y+12,GREEN);
    Putpixel64k(x+14,y+7,GREEN);
    Putpixel64k(x+14,y+8,GREEN);
    Putpixel64k(x+17,y+8,GREEN);//3
    Solid_Bar(x+3,y+17,x+9,y+19,GREEN);
}

```

```

    Putpixel64k(x+2,y+18,GREEN);
    Putpixel64k(x+2,y+19,GREEN);
    Putpixel64k(x+10,y+19,GREEN);
    Line_Plus(x+3,y+16,x+6,y+16,GREEN);
    Putpixel64k(x+3,y+14,GREEN);
    Putpixel64k(x+3,y+15,GREEN);
    Solid_Bar(x+5,y+13,x+6,y+15,GREEN);
    Putpixel64k(x+8,y+15,GREEN);
    Putpixel64k(x+8,y+16,GREEN);
}

/*****
*****
* 函数名称      Draw_Suburban_Field
* 函数作用      绘制郊区田地
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Suburban_Field(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,BROWN);
    Solid_Bar(x+3,y+3,x+20,y+6,MAROON);
    Solid_Bar(x+3,y+10,x+20,y+13,MAROON);
    Solid_Bar(x+3,y+17,x+20,y+20,MAROON);
    Putpixel64k(x+2,y+4,MAROON);
    Putpixel64k(x+2,y+5,MAROON);
    Putpixel64k(x+2,y+11,MAROON);
    Putpixel64k(x+2,y+12,MAROON);
    Putpixel64k(x+2,y+18,MAROON);
    Putpixel64k(x+2,y+19,MAROON);//
    Putpixel64k(x+21,y+4,MAROON);
    Putpixel64k(x+21,y+5,MAROON);
    Putpixel64k(x+21,y+11,MAROON);
    Putpixel64k(x+21,y+12,MAROON);
    Putpixel64k(x+21,y+18,MAROON);
    Putpixel64k(x+21,y+19,MAROON);
}

/*****
*****
* 函数名称      Draw_Urban_H_Wide_Highway_Part

```

```

* 函数作用      绘制城区水平高速公路块
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```
void Draw_Urban_H_Wide_Highway_Part(int x,int y)
```

```

{
    Solid_Bar(x,y,x+23,y+47,GRAY);
    Solid_Bar(x,y+4,x+23,y+7,WHITE);
    Solid_Bar(x,y+22,x+23,y+25,YELLOW);
    Solid_Bar(x,y+40,x+23,y+43,WHITE);
    Solid_Bar(x,y+14,x+3,y+15,WHITE);
    Solid_Bar(x+8,y+14,x+15,y+15,WHITE);
    Solid_Bar(x+20,y+14,x+23,y+15,WHITE);
    Solid_Bar(x,y+32,x+3,y+33,WHITE);
    Solid_Bar(x+8,y+32,x+15,y+33,WHITE);
    Solid_Bar(x+20,y+32,x+23,y+33,WHITE);
}

```

```

/*****
*****

```

```

* 函数名称      Draw_Urban_V_Wide_Highway_Part
* 函数作用      绘制城区垂直高速公路块
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```
void Draw_Urban_V_Wide_Highway_Part(int x,int y)
```

```

{
    Solid_Bar(x,y,x+47,y+23,GRAY);
    Solid_Bar(x+4,y,x+7,y+23,WHITE);
    Solid_Bar(x+22,y,x+25,y+23,YELLOW);
    Solid_Bar(x+40,y,x+43,y+23,WHITE);
    Solid_Bar(x+14,y,x+15,y+3,WHITE);
    Solid_Bar(x+14,y+8,x+15,y+15,WHITE);
    Solid_Bar(x+14,y+20,x+15,y+23,WHITE);
    Solid_Bar(x+32,y,x+33,y+3,WHITE);
    Solid_Bar(x+32,y+8,x+33,y+15,WHITE);
    Solid_Bar(x+32,y+20,x+33,y+23,WHITE);
}

```

```

/*****
*****
* 函数名称      Draw_H_Narrow_Highway_Part
* 函数作用      绘制城区水平公路块
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_H_Narrow_Highway_Part(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,GRAY);
    Solid_Bar(x,y+2,x+23,y+3,WHITE);
    Solid_Bar(x,y+20,x+23,y+21,WHITE);
    Solid_Bar(x,y+11,x+3,y+12,YELLOW);
    Solid_Bar(x+8,y+11,x+15,y+12,YELLOW);
    Solid_Bar(x+20,y+11,x+23,y+12,YELLOW);
}

```

```

/*****
*****
* 函数名称      Draw_V_Narrow_Highway_Part
* 函数作用      绘制城区垂直公路块
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_V_Narrow_Highway_Part(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,GRAY);
    Solid_Bar(x+2,y,x+3,y+23,WHITE);
    Solid_Bar(x+20,y,x+21,y+23,WHITE);
    Solid_Bar(x+11,y,x+12,y+3,YELLOW);
    Solid_Bar(x+11,y+8,x+12,y+15,YELLOW);
    Solid_Bar(x+11,y+20,x+12,y+23,YELLOW);
}

```

```

/*****
*****
* 函数名称      Draw_H_Narrow_Highway_Part
* 函数作用      绘制城区水平公路块
* 函数输入      x,y 坐标, Length 长度

```

```

* 函数输出      无
*****
*****/

```

```

void Draw_Urban_H_Wide_Highway(int x,int y,int length)
{
    int i;
    for(i=0;i<length;i++)
    {
        Draw_Urban_H_Wide_Highway_Part(x,y);
        x+=24;
    }
}

```

```

/*****
*****
* 函数名称      Draw_Urban_V_Wide_Highway
* 函数作用      绘制城区垂直高速公路
* 函数输入      x,y 坐标, Length 长度
* 函数输出      无
*****
*****/

```

```

void Draw_Urban_V_Wide_Highway(int x,int y,int length)
{
    int i;
    for(i=0;i<length;i++)
    {
        Draw_Urban_V_Wide_Highway_Part(x,y);
        y+=24;
    }
}

```

```

/*****
*****
* 函数名称      Draw_H_Narrow_Highway
* 函数作用      绘制城区水平公路
* 函数输入      x,y 坐标, Length 长度
* 函数输出      无
*****
*****/

```

```

void Draw_H_Narrow_Highway(int x,int y,int length)

```



```

{
    int i;
    for(i=0;i<length;i++)
    {
        Draw_H_Narrow_Highway_Part(x,y);
        x+=24;
    }
}

/*****
*****
* 函数名称      Draw_V_Narrow_Highway
* 函数作用      绘制城区垂直公路
* 函数输入      x,y 坐标, length 长度
* 函数输出      无
*****
*****/

void Draw_V_Narrow_Highway(int x,int y,int length)
{
    int i;
    for(i=0;i<length;i++)
    {
        Draw_V_Narrow_Highway_Part(x,y);
        y+=24;
    }
}

/*****
*****
* 函数名称      Draw_Suburban_Path_Part
* 函数作用      绘制郊区小路块
* 函数输入      x,y 坐标, mode 模式
* 函数输出      无
*****
*****/

void Draw_Suburban_Path_Part(int x,int y,int mode)
{
    Solid_Bar(x,y,x+23,y+23,MARIGOLD);
    //Solid_Ellipse(x+8,y+7,5,3,BROWN);
    //Solid_Ellipse(x+15,y+16,3,2,DARK_RED);
    if(mode==1)

```

```

    {
        Bar_Random_Spot(x,y,x+23,y+4,25,BRIGHT_GREEN);
        Bar_Random_Spot(x,y+19,x+23,y+23,25,BRIGHT_GREEN);
    }
    if(mode==2)
    {
        Bar_Random_Spot(x,y,x+4,y+23,25,BRIGHT_GREEN);
        Bar_Random_Spot(x+19,y,x+23,y+23,25,BRIGHT_GREEN);
    }
    if(mode==3)
    {
        Bar_Random_Spot(x,y,x+23,y+4,25,BRIGHT_GREEN);
        Bar_Random_Spot(x,y+19,x+23,y+23,25,BRIGHT_GREEN);
        Bar_Random_Spot(x,y,x+4,y+23,25,BRIGHT_GREEN);
        Bar_Random_Spot(x+19,y,x+23,y+23,25,BRIGHT_GREEN);
    }
}

/*****
*****
* 函数名称      Draw_Suburban_H_Path
* 函数作用      绘制郊区水平小径
* 函数输入      x,y 坐标, Length 长度
* 函数输出      无
*****
*****/

void Draw_Suburban_H_Path(int x,int y,int length)
{
    int i;
    for(i=0;i<length;i++)
    {
        Draw_Suburban_Path_Part(x,y,1);
        x+=24;
    }
}

/*****
*****
* 函数名称      Draw_Suburban_V_Path
* 函数作用      绘制郊区垂直小径
* 函数输入      x,y 坐标, Length 长度
* 函数输出      无
*****/

```

```

*****
*****/

```

```

void Draw_Suburban_V_Path(int x,int y,int length)
{
    int i;
    for(i=0;i<length;i++)
    {
        Draw_Suburban_Path_Part(x,y,2);
        y+=24;
    }
}

```

```

/*****
*****

```

```

* 函数名称      Draw_Narrow_Bridge_End
* 函数作用      绘制小桥端点
* 函数输入      x,y 坐标, forward 方向
* 函数输出      无

```

```

*****
*****/

```

```

void Draw_Narrow_Bridge_End(int x,int y,int forward)
{
    if(forward==L_EFT)
    {
        Solid_Bar(x+23,y+3,x+17,y+23,MARIGOLD);
        Solid_Ellipse(x+20,y+2,3,2,BROWN);
        Solid_Ellipse(x+20,y+14,3,2,BROWN);
        Solid_Quadrangle(x+17,y+6,x,y,x+17,y+23,x,y+17,MARIGOLD);
        Hollow_Bar(x+23,y+3,x+17,y+23,BROWN);
        Hollow_Quadrangle(x+17,y+6,x+17,y+23,x,y+17,x,y,BROWN);
        Solid_Bar(x+15,y+7,x+12,y+19,BROWN);
        Solid_Bar(x+10,y+5,x+7,y+17,BROWN);
        Solid_Bar(x+5,y+3,x+2,y+15,BROWN);
    }
    if(forward==R_IGHT)
    {
        Solid_Bar(x,y+3,x+6,y+23,MARIGOLD);
        Solid_Ellipse(x+3,y+2,3,2,BROWN);
        Solid_Ellipse(x+3,y+14,3,2,BROWN);
        Solid_Quadrangle(x+6,y+6,x+23,y,x+6,y+23,x+23,y+17,MARIGOLD);
    }
}

```

```

        Hollow_Bar(x,y+3,x+6,y+23,BROWN);
        Hollow_Quadrangle(x+6,y+6,x+6,y+23,x+23,y+17,x+23,y,BROWN
    );

        Solid_Bar(x+8,y+7,x+11,y+19,BROWN);
        Solid_Bar(x+13,y+5,x+16,y+17,BROWN);
        Solid_Bar(x+18,y+3,x+21,y+15,BROWN);
    }
    if(forward==U_P)
    {
        Solid_Bar(x,y,x+23,y+23,MARIGOLD);
        Solid_Ellipse(x+3,y+13,3,2,BROWN);
        Solid_Ellipse(x+20,y+13,3,2,BROWN);
        Hollow_Bar(x,y+13,x+5,y+23,BROWN);
        Hollow_Bar(x+18,y+13,x+23,y+23,BROWN);
        Solid_Bar(x+2,y+2,x+21,y+5,BROWN);
        Solid_Bar(x+2,y+8,x+21,y+11,BROWN);
        Solid_Bar(x+7,y+14,x+16,y+17,BROWN);
        Solid_Bar(x+7,y+20,x+16,y+23,BROWN);
        Line_Plus(x,y,x,y+23,BROWN);
        Line_Plus(x+23,y,x+23,y+23,BROWN);
    }
    if(forward==D_DOWN)
    {
        Solid_Bar(x,y,x+23,y+23,MARIGOLD);
        Solid_Ellipse(x+3,y+10,3,2,BROWN);
        Solid_Ellipse(x+20,y+10,3,2,BROWN);
        Hollow_Bar(x,y+10,x+5,y,BROWN);
        Hollow_Bar(x+18,y+10,x+23,y,BROWN);
        Solid_Bar(x+2,y+18,x+21,y+21,BROWN);
        Solid_Bar(x+2,y+12,x+21,y+15,BROWN);
        Solid_Bar(x+7,y+6,x+16,y+9,BROWN);
        Solid_Bar(x+7,y,x+16,y+3,BROWN);
        Solid_Bar(x,y,x,y+23,BROWN);
        Line_Plus(x+23,y,x+23,y+23,BROWN);
    }
}

```

```

/*****
*****

```

```

* 函数名称      Draw_Narrow_Bridge_Connection
* 函数作用      绘制小桥中间
* 函数输入      x,y 坐标, forward 方向
* 函数输出      无

```

```

*****
*****/

```

```

void Draw_Narrow_Bridge_Connection(int x,int y,int forward)
{
    if(forward==H_ORIZONTAL)
    {
        Solid_Bar(x,y,x+23,y+17,MARIGOLD);
        Solid_Bar(x,y,x+23,y+1,BROWN);
        Solid_Bar(x,y+16,x+23,y+17,BROWN);
        Solid_Bar(x+1,y+3,x+4,y+14,BROWN);
        Solid_Bar(x+7,y+3,x+10,y+14,BROWN);
        Solid_Bar(x+13,y+3,x+16,y+14,BROWN);
        Solid_Bar(x+19,y+3,x+22,y+14,BROWN);
    }
    if(forward==V_ERTICAL)
    {
        Solid_Bar(x,y,x+23,y+23,MARIGOLD);
        Solid_Bar(x,y,x+1,y+23,BROWN);
        Solid_Bar(x+22,y,x+23,y+23,BROWN);
        Solid_Bar(x+3,y+1,x+20,y+4,BROWN);
        Solid_Bar(x+3,y+7,x+20,y+10,BROWN);
        Solid_Bar(x+3,y+13,x+20,y+16,BROWN);
        Solid_Bar(x+3,y+19,x+20,y+22,BROWN);
    }
    if(forward==C_ENTER)
    {
        Solid_Bar(x,y,x+23,y+23,MARIGOLD);
        Solid_Circle(x+11,y+11,5,BROWN);
    }
}

```

```

/*****
*****

```

```

* 函数名称      Draw_Suburban_H_Narrow_Bridge
* 函数作用      绘制郊区水平公路
* 函数输入      x,y 坐标, length 长度
* 函数输出      无

```

```

*****
*****/

```

```

void Draw_Suburban_H_Narrow_Bridge(int x,int y,int length)
{

```

```

    int i;
    Draw_Narrow_Bridge_End(x,y,R_IGHT);
    Draw_Narrow_Bridge_End(x+length*24-24,y,L_EFT);
    for(i=1;i<length-1;i++)
    {
        Draw_Narrow_Bridge_Connection(x+24*i,y,H_ORIZONTAL);
    }
}

/*****
*****
* 函数名称      Draw_Suburban_V_Narrow_Bridge
* 函数作用      绘制郊区垂直公路
* 函数输入      x,y 坐标, length 长度
* 函数输出      无
*****
*****/

void Draw_Suburban_V_Narrow_Bridge(int x,int y,int length)
{
    int i;
    Draw_Narrow_Bridge_End(x,y,D_OWN);
    Draw_Narrow_Bridge_End(x,y+length*24-24,U_P);
    for(i=1;i<length-1;i++)
    {
        Draw_Narrow_Bridge_Connection(x,y+24*i,V_ERTICAL);
    }
}

/*****
*****
* 函数名称      Draw_Wide_Bridge_End
* 函数作用      绘制高速公路桥端点
* 函数输入      x,y 坐标, forward 方向
* 函数输出      无
*****
*****/

void Draw_Wide_Bridge_End(int x,int y,int forward)
{
    if(forward==L_EFT)
    {
        Solid_Quadrangle(x,y+47,x,y+8,x+23,y,x+23,y+39,GRAY);
    }
}

```

```

        Solid_Quadrangle(x,y+26,x,y+29,x+23,y+18,x+23,y+21,YELLOW
    );
        Solid_Quadrangle(x,y+16,x,y+17,x+23,y+9,x+23,y+8,WHITE);
        Solid_Quadrangle(x,y+38,x,y+39,x+23,y+30,x+23,y+31,WHITE)
    ;

        Solid_Bar(x+10,y+9,x+13,y+16,GRAY);
        Solid_Bar(x+10,y+30,x+13,y+38,GRAY);
        Hollow_Quadrangle(x,y+47,x,y+8,x+23,y,x+23,y+39,DIMGRAY);
    }
    if(forward==R_IGHT)
    {
        Solid_Quadrangle(x+23,y+47,x+23,y+8,x,y,x,y+39,GRAY);
        Solid_Quadrangle(x+23,y+26,x+23,y+29,x,y+18,x,y+21,YELLOW
    );
        Solid_Quadrangle(x+23,y+16,x+23,y+17,x,y+9,x,y+8,WHITE);
        Solid_Quadrangle(x+23,y+38,x+23,y+39,x,y+30,x,y+31,WHITE)
    ;

        Solid_Bar(x+10,y+9,x+13,y+16,GRAY);
        Solid_Bar(x+10,y+30,x+13,y+38,GRAY);
        Hollow_Quadrangle(x+23,y+47,x+23,y+8,x,y,x,y+39,DIMGRAY);
    }
    if(forward==D_OWN)
    {
        Solid_Quadrangle(x+7,y,x+41,y,x,y+23,x+47,y+23,GRAY);
        Hollow_Quadrangle(x+40,y,x+6,y,x,y+23,x+47,y+23,DIMGRAY);
        Solid_Bar(x+22,y,x+25,y+23,YELLOW);
        Solid_Bar(x+13,y+3,x+14,y+9,WHITE);
        Solid_Bar(x+12,y+14,x+13,y+20,WHITE);
        Solid_Bar(x+33,y+3,x+34,y+9,WHITE);
        Solid_Bar(x+34,y+14,x+35,y+20,WHITE);
    }
    if(forward==U_P)
    {
        Solid_Quadrangle(x+7,y+23,x+41,y+23,x,y,x+47,y,GRAY);
        Hollow_Quadrangle(x+40,y+23,x+6,y+23,x,y,x+47,y,DIMGRAY);
        Solid_Bar(x+22,y+23,x+25,y,YELLOW);
        Solid_Bar(x+12,y+9,x+13,y+3,WHITE);
        Solid_Bar(x+13,y+20,x+14,y+14,WHITE);
        Solid_Bar(x+34,y+9,x+35,y+3,WHITE);
        Solid_Bar(x+33,y+20,x+34,y+14,WHITE);
    }
}

```

```

/*****
*****
* 函数名称      Draw_Wide_Bridge_Part
* 函数作用      绘制高速公路桥块
* 函数输入      x,y 坐标, forward 方向
* 函数输出      无
*****
*****/

```

```

void Draw_Wide_Bridge_Part(int x,int y,int forward)
{
    if(forward==H_ORIZONTAL)
    {
        Solid_Bar(x,y+2,x+23,y+37,GRAY);
        Solid_Bar(x,y,x+23,y+1,DIMGRAY);
        Solid_Bar(x,y+38,x+23,y+39,DIMGRAY);
        Solid_Bar(x,y+18,x+23,y+21,YELLOW);
        Solid_Bar(x+2,y+8,x+9,y+9,WHITE);
        Solid_Bar(x+14,y+8,x+21,y+9,WHITE);
        Solid_Bar(x+2,y+30,x+9,y+31,WHITE);
        Solid_Bar(x+14,y+30,x+21,y+31,WHITE);
        Solid_Triangle(x+6,y+40,x+9,y+40,x+9,y+43,GRAY);
        Solid_Triangle(x+17,y+40,x+14,y+40,x+14,y+43,GRAY);
        Solid_Bar(x+10,y+40,x+13,y+47,GRAY);
        Solid_Bar(x+11,y+43,x+12,y+47,DIMGRAY);
        Line_Plus(x+9,y+41,x+14,y+41,DIMGRAY);
        Line_Plus(x+10,y+42,x+13,y+42,DIMGRAY);
    }
    if(forward==V_ERTICAL)
    {
        Solid_Bar(x+6,y,x+41,y+23,GRAY);
        Solid_Bar(x+6,y,x+7,y+23,DIMGRAY);
        Solid_Bar(x+40,y,x+41,y+23,DIMGRAY);
        Solid_Bar(x+22,y,x+25,y+23,YELLOW);
        Solid_Bar(x+13,y+2,x+14,y+9,WHITE);
        Solid_Bar(x+13,y+14,x+14,y+21,WHITE);
        Solid_Bar(x+33,y+2,x+34,y+9,WHITE);
        Solid_Bar(x+33,y+14,x+34,y+21,WHITE);
        Line_Plus(x+5,y+8,x+5,y+15,GRAY);
        Line_Plus(x+4,y+9,x+4,y+14,GRAY);
        Solid_Bar(x,y+10,x+3,y+13,GRAY);
        Solid_Bar(x,y+11,x+4,y+12,DIMGRAY);
        Line_Plus(x+42,y+8,x+42,y+15,GRAY);
    }
}

```



```

        Line_Plus(x+43,y+9,x+43,y+14,GRAY);
        Solid_Bar(x+47,y+10,x+44,y+13,GRAY);
        Solid_Bar(x+47,y+11,x+43,y+12,DIMGRAY);
    }
}

/*****
*****
* 函数名称      Draw_Urban_H_Wide_Bridge
* 函数作用      绘制城区水平宽桥
* 函数输入      x,y 坐标, Length 长度
* 函数输出      无
*****
*****/

void Draw_Urban_H_Wide_Bridge(int x,int y,int length)
{
    int i;
    Draw_Wide_Bridge_End(x,y,L_EFT);
    Draw_Wide_Bridge_End(x+length*24-24,y,R_IGHT);
    for(i=1;i<length-1;i++)
    {
        Draw_Wide_Bridge_Part(x+24*i,y,H_ORIZONTAL);
    }
}

/*****
*****
* 函数名称      Draw_Urban_V_Wide_Bridge
* 函数作用      绘制城区垂直宽桥
* 函数输入      x,y 坐标, Length 长度
* 函数输出      无
*****
*****/

void Draw_Urban_V_Wide_Bridge(int x,int y,int length)
{
    int i;
    Draw_Wide_Bridge_End(x,y,U_P);
    Draw_Wide_Bridge_End(x,y+length*24-24,D_OWN);
    for(i=1;i<length-1;i++)
    {
        Draw_Wide_Bridge_Part(x,y+24*i,V_ERTICAL);
    }
}

```

```

    }
}

/*****
*****
* 函数名称      Draw_Window
* 函数作用      绘制窗户
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Window(int x,int y)
{
    Solid_Bar(x,y,x+2,y+2,SKY_BLUE);
    Solid_Bar(x+4,y,x+6,y+2,SKY_BLUE);
    Solid_Bar(x,y+4,x+2,y+6,SKY_BLUE);
    Solid_Bar(x+4,y+4,x+6,y+6,SKY_BLUE);
}

/*****
*****
* 函数名称      Draw_Mountainous_House
* 函数作用      绘制山区小屋
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Mountainous_House(int x,int y)
{
    Solid_Bar(x+17,y+2,x+18,y+7,DIMGRAY);
    Solid_Triangle(x+11,y,x+1,y+10,x+22,y+10,BROWN);
    Solid_Bar(x+4,y+11,x+19,y+23,MARIGOLD);
    Draw_Window(x+6,y+12);
    Solid_Bar(x+14,y+17,x+17,y+23,BROWN);
    Putpixel64k(x+16,y+20,MARIGOLD);
}

/*****
*****
* 函数名称      Draw_Suburban_House
* 函数作用      绘制郊区小屋

```

```

* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Suburban_House(int x,int y)
{
    Solid_Bar(x+11,y+1,x+12,y+9,DIMGRAY);
    Solid_Triangle(x+5,y+7,x+1,y+11,x+10,y+11,BROWN);
    Solid_Triangle(x+15,y+1,x+9,y+11,x+22,y+11,BROWN);
    Solid_Bar(x+3,y+12,x+20,y+23,MARIGOLD);
    Draw_Window(x+4,y+13);
    Solid_Bar(x+13,y+16,x+18,y+23,BROWN);
    Line_Plus(x+14,y+15,x+17,y+15,BROWN);
    Line_Plus(x+15,y+14,x+16,y+14,BROWN);
    Putpixel64k(x+17,y+19,MARIGOLD);
}

```

```

/*****
*****
* 函数名称      Draw_Random_Window
* 函数作用      绘制随机窗户
* 函数输入      x,y 坐标, length 长度, heigh 宽度, block 间隔
* 函数输出      无
*****
*****/

```

```

void Draw_Random_Window(int x,int y,int length,int heigh,int block)
{
    int i,color;
    for(i=0;i<6;i++)
    {
        if(random(2)==0)
        {
            color=SKY_BLUE;
        }
        else
        {
            color=DIMGRAY;
        }
        Solid_Bar(x,y+(heigh+block)*i,x+length,y+heigh-1+(heigh+block)*i,color);
    }
}

```

```

    }
}

/*****
*****
* 函数名称      Draw_Urban_House
* 函数作用      绘制城区小楼
* 函数输入      x,y 坐标, length 长度, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Urban_House(int x,int y,int color)
{
    y-=24;
    Solid_Bar(x+2,y+5,x+21,y+47,color);
    Draw_Random_Window(x+4,y+7,3,3,3);
    Draw_Random_Window(x+10,y+7,4,3,3);
    Draw_Random_Window(x+17,y+7,3,3,3);
    Solid_Bar(x+8,y+1,x+15,y+4,STEEL_BLUE);
    Bar_Random_Spot(x+8,y+1,x+15,y+4,5,WHITE);
    Solid_Bar(x+8,y+42,x+15,y+47,DIMGRAY);
    Solid_Bar(x+11,y+42,x+12,y+47,SILVER);
}

```

```

/*****
*****
* 函数名称      Draw_Random_Single_Window
* 函数作用      绘制随机单窗户
* 函数输入      x,y 坐标, sizex 宽度, sizey 长度
* 函数输出      无
*****
*****/

```

```

void Draw_Random_Single_Window(int x,int y,int sizex,int sizey)
{
    int color,ran=3;
    if(random(ran)<2)
    {
        color=YELLOW;
    }
    else
    {

```

```

        color=DIMGRAY;
    }
    Solid_Bar(x,y,x+sizeX-1,y+sizeY-1,color);
}

/*****
*****
* 函数名称      Draw_Urban_Hospital
* 函数作用      绘制城区医院
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Urban_Hospital(int x,int y)
{
    int i;
    Solid_Bar(x,y+22,x+12,y+47,GRAY);
    Solid_Bar(x+13,y+12,x+34,y+47,STEEL_BLUE);
    Solid_Bar(x+35,y+16,x+47,y+47,GRAY);
    Solid_Bar(x+19,y+7,x+28,y+10,RED);
    Solid_Bar(x+22,y+4,x+25,y+13,RED);
    Putpixel64k(x+23,y+3,RED);
    Putpixel64k(x+24,y+3,RED);
    Putpixel64k(x+18,y+8,RED);
    Putpixel64k(x+18,y+9,RED);
    Putpixel64k(x+23,y+14,RED);
    Putpixel64k(x+24,y+14,RED);
    Putpixel64k(x+29,y+8,RED);
    Putpixel64k(x+29,y+9,RED);
    Solid_Bar(x+16,y+40,x+31,y+47,GRAY);
    Solid_Bar(x+20,y+40,x+21,y+47,WHITE);
    Solid_Bar(x+26,y+40,x+27,y+47,WHITE);
    Draw_Random_Single_Window(x+2,y+24,9,5);
    Draw_Random_Single_Window(x+2,y+31,9,5);
    Draw_Random_Single_Window(x+2,y+38,3,3);
    Draw_Random_Single_Window(x+8,y+38,3,3);
    for(i=0;i<4;i++)
    {
        Draw_Random_Single_Window(x+16,y+16+6*i,3,3);
        Draw_Random_Single_Window(x+22,y+16+6*i,4,3);
        Draw_Random_Single_Window(x+29,y+16+6*i,3,3);
    }
}

```

```

        Draw_Random_Single_Window(x+37,y+18,9,5);
        Draw_Random_Single_Window(x+37,y+31,9,5);
        Draw_Random_Single_Window(x+37,y+25,3,3);
        Draw_Random_Single_Window(x+43,y+25,3,3);
        Draw_Random_Single_Window(x+37,y+39,3,3);
        Draw_Random_Single_Window(x+43,y+39,3,3);
    }

    /*****
    *****/
    * 函数名称      Draw_Suburban_Bush
    * 函数作用      绘制郊区灌木
    * 函数输入      x,y 坐标
    * 函数输出      无
    *****/
    *****/

void Draw_Suburban_Bush(int x,int y)
{
    Solid_Ellipse(x+13,y+8,9,5,GREEN);
    Solid_Ellipse(x+5,y+14,5,3,GREEN);
    Solid_Bar(x+12,y+13,x+15,y+14,BROWN);
    Solid_Bar(x+14,y+14,x+16,y+23,BROWN);
    Line_Plus(x+7,y+17,x+10,y+17,BROWN);
    Line_Plus(x+9,y+18,x+11,y+18,BROWN);
    Line_Plus(x+11,y+19,x+13,y+19,BROWN);
    Line_Plus(x+12,y+20,x+13,y+20,BROWN);
}

    /*****
    *****/
    * 函数名称      Draw_Bush_Redball
    * 函数作用      绘制灌木果子
    * 函数输入      x,y 坐标
    * 函数输出      无
    *****/
    *****/

void Draw_Bush_Redball(int x,int y)
{
    Solid_Bar(x,y,x+1,y+1,RED);
}

```

```

/*****
*****
* 函数名称      Draw_Bush_Orangeball
* 函数作用      绘制灌木橙色果
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Bush_Orangeball(int x,int y)
{
    Solid_Bar(x,y,x+1,y+1,ORANGE);
}

```

```

/*****
*****
* 函数名称      Draw_Bush_Greenball
* 函数作用      绘制灌木绿色果
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Bush_Greenball(int x,int y)
{
    Solid_Bar(x,y,x+1,y+1,BRIGHT_GREEN);
}

```

```

/*****
*****
* 函数名称      Draw_Mountainous_Bush
* 函数作用      绘制山区灌木
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Mountainous_Bush(int x,int y)
{
    Solid_Ellipse(x+11,y+7,7,5,GREEN);
    Solid_Ellipse(x+11,y+12,11,6,GREEN);
    Solid_Quadrangle(x+10,y+19,x+13,y+19,x+8,y+23,x+15,y+23,BROWN
);
}

```

```

        Draw_Bush_Redball(x+10,y+5);
        Draw_Bush_Redball(x+12,y+9);
        Draw_Bush_Redball(x+5,y+10);
        Draw_Bush_Redball(x+18,y+12);
        Draw_Bush_Redball(x+10,y+14);
    }

    /*****
    *****/
    * 函数名称      Draw_Store_Window
    * 函数作用      绘制商店窗户
    * 函数输入      x,y 坐标, length 长度, forward 绘制方向
    * 函数输出      无
    *****/
    *****/

void Draw_Store_Window(int x,int y,int length,int forward)
{
    int i,color,ran=3;
    for(i=0;i<length;i++)
    {
        if(random(ran)<2)
        {
            color=YELLOW;
        }
        else
        {
            color=STEEL_BLUE;
        }
        if(forward==L_EFT)
        {
            Solid_Bar(x+6*i,y+i,x+2+6*i,y+2+i,color);
        }
        if(forward==R_IGHT)
        {
            Solid_Bar(x-6*i,y+i,x+2-6*i,y+2+i,color);
        }
    }
}

    /*****
    *****/
    * 函数名称      Draw_Urban_Store

```



```

* 函数作用      绘制城区商店
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Urban_Store(int x,int y,int color)
{
    int i,j,ex0,ey0;
    Solid_Bar(x+6,y+38,x+65,y+71,color);
    Solid_Triangle(x+7,y+19,x+35,y+4,x+64,y+19,color);
    Solid_Ellipse(x+35,y+29,36,19,ORANGE);
    Solid_Bar(x+22,y+12,x+49,y+13,RED);
    Solid_Bar(x+11,y+16,x+60,y+17,YELLOW);
    Solid_Bar(x+6,y+20,x+66,y+21,GREEN);
    Solid_Bar(x+5,y+38,x+66,y+39,AQUA_BLUE);
    Solid_Bar(x+11,y+42,x+60,y+43,BLUE);
    Solid_Bar(x+22,y+46,x+49,y+47,PURPLE);
    //STORE
    Solid_Bar(x+17,y+25,x+22,y+26,RED);
    Solid_Bar(x+17,y+27,x+18,y+30,RED);
    Solid_Bar(x+19,y+29,x+22,y+30,RED);
    Solid_Bar(x+21,y+31,x+22,y+34,RED);
    Solid_Bar(x+17,y+33,x+22,y+34,RED);//S
    Solid_Bar(x+25,y+25,x+30,y+26,RED);
    Solid_Bar(x+27,y+27,x+28,y+34,RED);//T
    Solid_Bar(x+33,y+25,x+38,y+34,RED);
    Solid_Bar(x+35,y+27,x+36,y+32,ORANGE);//O
    Solid_Bar(x+41,y+25,x+46,y+34,RED);
    Draw_Bush_Orangeball(x+43,y+27);
    Draw_Bush_Orangeball(x+45,y+31);
    Draw_Bush_Orangeball(x+43,y+33);//R
    ex0=x+49;
    ey0=y+25;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            Draw_Bush_Redball(ex0+2*i,ey0+4*j);
        }
    }
    Draw_Bush_Redball(x+49,y+27);
    Draw_Bush_Redball(x+49,y+31);//E

```

```

Solid_Upper_Ellipse(x+35,y+71,18,9,SKY_BLUE);
Solid_Bar(x+35,y+63,x+36,y+71,DIMGRAY);
Line_Plus(x+24,y+69,x+26,y+67,WHITE);
Line_Plus(x+29,y+68,x+31,y+66,WHITE);
Line_Plus(x+40,y+66,x+42,y+68,WHITE);
Line_Plus(x+45,y+67,x+47,y+69,WHITE);
for(i=4;i>0;i--)
{
    Draw_Store_Window(x+9,y+49+6*(4-i),i,L_EFT);
}
for(i=4;i>0;i--)
{
    Draw_Store_Window(x+60,y+49+6*(4-i),i,R_IGHT);
}
Solid_Bar(x+27,y+58,x+44,y+60,RED);
Solid_Bar(x+35,y,x+36,y+3,YELLOW);
Line_Plus(x+34,y+2,x+37,y+2,YELLOW);
}

/*****
*****
* 函数名称      Draw_Suburban_Appletree
* 函数作用      绘制郊区苹果树
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Suburban_Appletree(int x,int y)
{
    int i;
    y-=24;
    Solid_Bar(x+8,y+21,x+15,y+37,BROWN);
    Solid_Bar(x+7,y+38,x+16,y+43,BROWN);
    Solid_Bar(x+6,y+43,x+17,y+44,BROWN);
    for(i=0;i<3;i++)
    {
        Line_Plus(x+5-i,y+45+i,x+18+i,y+45+i,BROWN);
    }
    Solid_Circle(x+17,y+22,7,GREEN);
    Solid_Circle(x+6,y+22,7,GREEN);
    Solid_Circle(x+11,y+12,8,GREEN);
    Draw_Bush_Redball(x+3,y+20);
}

```

```

        Draw_Bush_Redball(x+7,y+14);
        Draw_Bush_Redball(x+9,y+9);
        Draw_Bush_Redball(x+11,y+20);
        Draw_Bush_Redball(x+13,y+15);
        Draw_Bush_Redball(x+16,y+10);
        Draw_Bush_Redball(x+18,y+23);
    }

    /*****
    *****/
    * 函数名称      Draw_Suburban_Sakura
    * 函数作用      绘制郊区樱花树
    * 函数输入      x,y 坐标
    * 函数输出      无
    *****/
    *****/

void Draw_Suburban_Sakura(int x,int y)
{
    int i;
    y-=24;
    Solid_Bar(x+9,y+24,x+11,y+44,BROWN);
    for(i=0;i<3;i++)
    {
        Line_Plus(x+9-i,y+45+i,x+11+i,y+45+i,BROWN);
    }
    Solid_Circle(x+11,y+12,12,HOT_PINK);
    Solid_Circle(x+6,y+36,4,HOT_PINK);
    Solid_Circle(x+17,y+30,4,HOT_PINK);
}

    /*****
    *****/
    * 函数名称      Draw_Flower
    * 函数作用      绘制花朵
    * 函数输入      x,y 坐标
    * 函数输出      无
    *****/
    *****/

void Draw_Flower(int x,int y,int color)
{
    Putpixel64k(x,y,color);
}

```

```

        Putpixel64k(x+1,y,color);
        Putpixel64k(x,y+1,color);
        Putpixel64k(x-1,y,color);
        Putpixel64k(x,y-1,color);
    }

/*****
*****
* 函数名称      Draw_Urban_Flower
* 函数作用      绘制城区花朵
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Urban_Flower(int x,int y)
{
    int i;
    for(i=0;i<3;i++)
    {
        Line_Plus(x+10-i,y+21+i,x+13+i,y+21+i,BROWN);
    }
    Solid_Ellipse(x+11,y+12,12,9,BRIGHT_GREEN);
    Draw_Flower(x+5,y+13,YELLOW);
    Draw_Flower(x+12,y+8,YELLOW);
    Draw_Flower(x+16,y+14,YELLOW);
}

/*****
*****
* 函数名称      Draw_Urban_Tree
* 函数作用      绘制城区树
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Urban_Tree(int x,int y)
{
    int i;
    y-=24;
    Solid_Bar(x+9,y+30,x+14,y+41,BROWN);
    Solid_Bar(x+8,y+42,x+15,y+44,BROWN);
}

```

```

    for(i=0;i<3;i++)
    {
        Line_Plus(x+7-i,y+45+i,x+16+i,y+45+i,BROWN);
    }
    Solid_Ellipse(x+11,y+12,6,11,GREEN);
    Solid_Ellipse(x+11,y+29,12,6,GREEN);
    Solid_Circle(x+11,y+21,8,GREEN);
    Draw_Bush_Greenball(x+5,y+29);
    Draw_Bush_Greenball(x+11,y+23);
    Draw_Bush_Greenball(x+14,y+31);
    Draw_Bush_Greenball(x+18,y+28);
    Draw_Bush_Greenball(x+9,y+17);
    Draw_Bush_Greenball(x+12,y+9);
    Draw_Bush_Greenball(x+8,y+7);
}

/*****
*****
* 函数名称      Draw_Urban_House_Icon
* 函数作用      绘制城区屋子图标
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Urban_House_Icon(int x,int y)
{
    Solid_Bar(x+2,y+5,x+21,y+23,GRAY);
    Draw_Random_Window_Icon(x+4,y+7,3,3,3);
    Draw_Random_Window_Icon(x+10,y+7,4,3,3);
    Draw_Random_Window_Icon(x+17,y+7,3,3,3);
    Solid_Bar(x+8,y+1,x+15,y+4,STEEL_BLUE);
    Bar_Random_Spot(x+8,y+1,x+15,y+4,5,WHITE);
}

/*****
*****
* 函数名称      Draw_Random_Window_Icon
* 函数作用      绘制随机窗户图标
* 函数输入      x,y 坐标, length 长度, heigh 高度, block 间隔
* 函数输出      无
*****
*****/

```

```

void Draw_Random_Window_Icon(int x,int y,int length,int heigh,int
block)
{
    int i,color,ran=2;
    for(i=0;i<3;i++)
    {
        if(random(ran)==0)
        {
            color=SKY_BLUE;
        }
        else
        {
            color=DIMGRAY;
        }
        Solid_Bar(x,y+(heigh+block)*i,x+length,y+heigh-
1+(heigh+block)*i,color);
    }
}

```

```

/*****
*****

```

```

* 函数名称      Draw_River_Icon
* 函数作用      绘制河图标
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_River_Icon(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,STEEL_BLUE);
    Draw_Wave(x+7,y+11);
}

```

```

/*****
*****

```

```

* 函数名称      Draw_Urban_Bridge_Icon
* 函数作用      绘制城区桥图标
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

```

```

void Draw_Urban_Bridge_Icon(int x,int y)
{
    y-=24;
        Solid_Bar(x,y+24,x+23,y+37,GRAY);
        Solid_Bar(x,y+38,x+23,y+39,DIMGRAY);
        Solid_Bar(x+2,y+30,x+9,y+31,WHITE);
        Solid_Bar(x+14,y+30,x+21,y+31,WHITE);
        Solid_Triangle(x+6,y+40,x+9,y+40,x+9,y+43,GRAY);
        Solid_Triangle(x+17,y+40,x+14,y+40,x+14,y+43,GRAY);
        Solid_Bar(x+10,y+40,x+13,y+47,GRAY);
        Solid_Bar(x+11,y+43,x+12,y+47,DIMGRAY);
        Line_Plus(x+9,y+41,x+14,y+41,DIMGRAY);
        Line_Plus(x+10,y+42,x+13,y+42,DIMGRAY);
}

/*****
*****
* 函数名称      Draw_Mountain
* 函数作用      绘制山区块
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Mountain(int x,int y)
{
    int i;
    Solid_Bar(x,y,x+23,y+23,BRIGHT_GREEN);
    Solid_Bar(x+4,y+6,x+11,y+8,GRAY);
    for(i=0;i<3;i++)
    {
        Line_Plus(x+6-i,y+3+i,x+9+i,y+3+i,GRAY);
    }
    Solid_Bar(x+3,y+17,x+10,y+19,BROWN);
    for(i=0;i<3;i++)
    {
        Line_Plus(x+5-i,y+15+i,x+8+i,y+15+i,BROWN);
    }
    for(i=0;i<2;i++)
    {
        Line_Plus(x+18-i,y+3+i,x+19+i,y+3+i,MARIGOLD);
    }
}

```

```

        Solid_Bar(x+15,y+14,x+20,y+15,DIMGRAY);
        Line_Plus(x+16,y+13,x+19,y+13,DIMGRAY);
    }

/*****
*****
* 函数名称      Draw_Urban_Midhouse
* 函数作用      绘制城区中等屋子
* 函数输入      x,y 坐标, color 颜色1, color2 颜色2
* 函数输出      无
*****
*****/

void Draw_Urban_Midhouse(int x,int y,int color1,int color2)
{
    int i;
    Solid_Bar(x,y+22,x+12,y+47,color2);
    Solid_Bar(x+13,y+12,x+34,y+47,color1);
    Solid_Bar(x+35,y+16,x+47,y+47,color2);
    Solid_Bar(x+16,y+40,x+31,y+47,GRAY);
    Solid_Bar(x+20,y+40,x+21,y+47,WHITE);
    Solid_Bar(x+26,y+40,x+27,y+47,WHITE);
    Draw_Random_Single_Window(x+2,y+24,9,5);
    Draw_Random_Single_Window(x+2,y+31,9,5);
    Draw_Random_Single_Window(x+2,y+38,3,3);
    Draw_Random_Single_Window(x+8,y+38,3,3);
    for(i=0;i<4;i++)
    {
        Draw_Random_Single_Window(x+16,y+16+6*i,3,3);
        Draw_Random_Single_Window(x+22,y+16+6*i,4,3);
        Draw_Random_Single_Window(x+29,y+16+6*i,3,3);
    }
    Draw_Random_Single_Window(x+37,y+18,9,5);
    Draw_Random_Single_Window(x+37,y+31,9,5);
    Draw_Random_Single_Window(x+37,y+25,3,3);
    Draw_Random_Single_Window(x+43,y+25,3,3);
    Draw_Random_Single_Window(x+37,y+39,3,3);
    Draw_Random_Single_Window(x+43,y+39,3,3);
}

/*****
*****
* 函数名称      Draw_Mountainous_On_Grass

```



```

* 函数作用      绘制山上草地
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Mountainous_On_Grass(int x,int y)
{
    Draw_Leaf_1(x,y,12,GREEN);
    Draw_Leaf_1(x+12,y+12,12,GREEN);
    Draw_Leaf_2(x+12,y,12,GREEN);
    Draw_Leaf_2(x,y+12,12,GREEN);
    Solid_Circle(x+12,y+12,2,DARK_GREEN);
}

/*****
*****
* 函数名称      Draw_Main_Crossing
* 函数作用      绘制主路十字路
* 函数输入      x,y 坐标
* 函数输出      无
*****
*****/

void Draw_Main_Crossing(int x,int y)
{
    Solid_Bar(x,y,x+47,y+47,GRAY);
    Solid_Circle(x+23,y+23,8,YELLOW);
    Solid_QuarterSector_left_up(x+47,y+47,16,WHITE);
    Solid_QuarterSector_left_up(x+47,y+47,14,GRAY);
    Solid_QuarterSector_right_up(x,y+47,16,WHITE);
    Solid_QuarterSector_right_up(x,y+47,14,GRAY);
    Solid_QuarterSector_left_down(x+47,y,16,WHITE);
    Solid_QuarterSector_left_down(x+47,y,14,GRAY);
    Solid_QuarterSector_right_down(x,y,16,WHITE);
    Solid_QuarterSector_right_down(x,y,14,GRAY);
}

/*****
*****
* 函数名称      Draw_Small_Crossing
* 函数作用      绘制小路十字路
* 函数输入      x,y 坐标

```

```

* 函数输出      无
*****
*****/

void Draw_Small_Crossing(int x,int y)
{
    Solid_Bar(x,y,x+23,y+23,GRAY);
    Solid_Circle(x+11,y+11,5,YELLOW);
}

/*****
*****

* 函数名称      Draw_Choose
* 函数作用      绘制选择框
* 函数输入      x,y 坐标, done 选择模式
* 函数输出      无
*****
*****/

void Draw_Choose(int x,int y,int done)
{
    Ring(x+11,y+11,11,12,RED);
    if(done==1)
    {
        Solid_Circle(x+11,y+11,4,RED);
    }
}

/*****
*****

* 函数名称      InfoImage_Draw
* 函数作用      绘制信息图片
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

void InfoImage_Draw(int x,int y,int color)
{
    Ring(x+19,y+19,13,17,color);
    Solid_Circle(x+19,y+11,2,color);
    Solid_Bar(x+16,y+16,x+21,y+18,color);
    Solid_Bar(x+19,y+19,x+21,y+26,color);
}

```

```

        Solid_Bar(x+16,y+27,x+24,y+29,color);
    }

/*****
*****
* 函数名称      Draw_On_Mountainous_Grass
* 函数作用      绘制山区上草地
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_On_Mountainous_Grass(int x,int y)
{
    int ran_x,ran_y;
    ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;
    y+=ran_y;
    Solid_Bar(x+4,y+4,x+5,y+6,GRAY);
    Putpixel64k(x+3,y+5,GRAY);
    Putpixel64k(x+6,y+5,GRAY);//1
    ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;
    y+=ran_y;
    Solid_Bar(x+18,y+18,x+20,y+19,YELLOW_GREEN);
    Putpixel64k(x+19,y+17,GRAY);//2
    ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;
    y+=ran_y;
    Solid_Bar(x+13,y+9,x+19,y+12,GREEN);
    Solid_Bar(x+17,y+7,x+19,y+8,GREEN);
    Solid_Bar(x+17,y+4,x+18,y+6,GREEN);
    Solid_Bar(x+20,y+11,x+21,y+12,GREEN);
    Putpixel64k(x+12,y+12,GREEN);
    Putpixel64k(x+14,y+7,GREEN);
    Putpixel64k(x+14,y+8,GREEN);
    Putpixel64k(x+17,y+8,GREEN);//3
    ran_x=random(5)-2;
    ran_y=random(5)-2;
    x+=ran_x;

```

```

        y+=ran_y;
        Solid_Bar(x+3,y+17,x+9,y+19,GREEN);
        Putpixel64k(x+2,y+18,GREEN);
        Putpixel64k(x+2,y+19,GREEN);
        Putpixel64k(x+10,y+19,GREEN);
        Line_Plus(x+3,y+16,x+6,y+16,GREEN);
        Putpixel64k(x+3,y+14,GREEN);
        Putpixel64k(x+3,y+15,GREEN);
        Solid_Bar(x+5,y+13,x+6,y+15,GREEN);
        Putpixel64k(x+8,y+15,GREEN);
        Putpixel64k(x+8,y+16,GREEN);
    }

/*****
*****
* 函数名称      Draw_Teen_Icon
* 函数作用      绘制青年图标
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

void Draw_Teen_Icon(int x0,int y0,int color)
{
    Solid_Circle(x0+11,y0+5,5,BODYCOLOR); //head
    Solid_Bar(x0+7,y0+2,x0+15,y0+3,RED);
    Putpixel64k(x0+6,y0+3,RED);
    Putpixel64k(x0+16,y0+3,RED);
    Solid_Bar(x0+6,y0+12,x0+17,y0+23,color);
    Solid_Bar(x0+2,y0+12,x0+4,y0+23,color);
    Solid_Bar(x0+19,y0+12,x0+21,y0+23,color);
}

/*****
*****
* 函数名称      Draw_Child_Icon
* 函数作用      绘制少年图标
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

void Draw_Child_Icon(int x0,int y0,int color)

```

```

{
    Solid_Circle(x0+11,y0+11,5,BODYCOLOR);//head
    Solid_Bar(x0+7,y0+18,x0+16,y0+23,color);
    Solid_Bar(x0+4,y0+18,x0+5,y0+23,color);
    Solid_Bar(x0+18,y0+18,x0+19,y0+23,color);
}

/*****
*****
* 函数名称      Draw_Mid_Icon
* 函数作用      绘制中年图标
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

void Draw_Mid_Icon(int x0,int y0,int color)
{
    Solid_Circle(x0+11,y0+5,5,BODYCOLOR);//head
    Solid_Bar(x0+6,y0+12,x0+17,y0+23,color);
    Solid_Bar(x0+2,y0+12,x0+4,y0+23,color);
    Solid_Bar(x0+19,y0+12,x0+21,y0+23,color);
}

/*****
*****
* 函数名称      Draw_Old_Icon
* 函数作用      绘制老年图标
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

void Draw_Old_Icon(int x0,int y0,int color)
{
    Solid_HalfSector_up(x0+11,y0+4,5,GRAY);//hair
    Solid_HalfSector_down(x0+11,y0+4,5,BODYCOLOR);//head
    Solid_Bar(x0+6,y0+11,x0+17,y0+23,color);
    Solid_Bar(x0+2,y0+11,x0+4,y0+23,color);
    Solid_Bar(x0+19,y0+11,x0+21,y0+23,color);
}

/*****
*****/

```

```

*****
* 函数名称      LoginImage_Draw
* 函数作用      绘制登录图标
* 函数输入      x,y 坐标, color 颜色
* 函数输出      无
*****
*****/

```

```

void LoginImage_Draw(int x,int y,int color)
{
    int i,j;
    int xo=x+19,yo=y+43,xc=x+19,yc=y+19;
    Ring(x+19,y+19,14,18,color);
    Solid_Circle(x+19,y+15,6,color);
    for(i=x;i<x+40;i++)
    {
        for(j=y+20;j<y+39;j++)
        {
            if(((i-xo)*(i-xo)+(j-yo)*(j-yo)<=20*20)&&((i-xc)*(i-
xc)+(j-yc)*(j-yc)<=14*14))
            {
                Putpixel64k(i,j,color);
            }
        }
    }
}

```

```

/*****
*****
* 函数名称      Draw_Arrow
* 函数作用      绘制箭头
* 函数输入      x,y 坐标, direction 方向, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Arrow(int x,int y,int diretion,int color)
{
    switch(diretion)
    {
        case 1:
            Solid_Bar(x+8,y+12,x+15,y+20,color);

```

```

        Solid_Triangle(x+11,y+3,x+3,y+11,x+20,y+11,color);

        break;
    case 2:
        Solid_Bar(x+8,y+11,x+15,y+3,color);
        Solid_Triangle(x+11,y+20,x+3,y+12,x+20,y+12,color);
        break;
    case 3:
        Solid_Bar(x+12,y+8,x+20,y+15,color);
        Solid_Triangle(x+3,y+11,x+11,y+3,x+11,y+20,color);

        break;
    case 4:
        Solid_Bar(x+11,y+8,x+3,y+15,color);
        Solid_Triangle(x+20,y+11,x+12,y+3,x+12,y+20,color);

        break;
    case 5:
        Solid_Triangle(x+3,y+3,x+3,y+15,x+15,y+3,color);
        Solid_Quadrangle(x+7,y+11,x+11,y+7,x+14,y+17,x+17,y+1
4,color);
        break;

    case 6:
        Solid_Triangle(x+3,y+20,x+3,y+8,x+15,y+20,color);
        Solid_Quadrangle(x+7,y+12,x+11,y+16,x+14,y+6,x+17,y+9
,color);
        break;
    case 7:
        Solid_Triangle(x+20,y+3,x+20,y+15,x+8,y+3,color);
        Solid_Quadrangle(x+16,y+11,x+12,y+7,x+9,y+17,x+6,y+14
,color);
        break;
    case 8:
        Solid_Triangle(x+20,y+20,x+20,y+8,x+8,y+20,color);
        Solid_Quadrangle(x+16,y+12, x+12,y+16,x+9,y+6,x+6,y+9,
color);
        break;
    }
}

/*****
*****/

```

```

* 函数名称      Draw_Flag
* 函数作用      绘制旗帜
* 函数输入      x,y 坐标, direction 方向, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Flag(int x,int y)
{
    Solid_Ellipse(x+8,y+15,6,3,BROWN);
    Solid_Bar(x+8,y+1,x+9,y+15,WHITE);
    Solid_Triangle(x+10,y+1,x+10,y+9,x+17,y+5,RED);
}

```

```

/*****
*****
* 函数名称      Draw_Clock
* 函数作用      绘制时钟
* 函数输入      x,y 坐标, direction 方向, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Clock(int x,int y)
{
    Solid_Circle(x+9,y+9,7,WHITE);
    Ring(x+9,y+9,6,7,GOLDEN);
    Line_Plus(x+6,y+7,x+9,y+10,RED);
    Line_Plus(x+9,y+10,x+14,y+5,RED);
}

```

```

/*****
*****
* 函数名称      Draw_Power
* 函数作用      绘制能量图标
* 函数输入      x,y 坐标, direction 方向, color 颜色
* 函数输出      无
*****
*****/

```

```

void Draw_Power(int x,int y)
{
    Line_Plus(x+5,y+1,x+2,y+4,YELLOW);
}

```



```

        Line_Plus(x+5,y+2,x+2,y+5,YELLOW);
        Solid_Bar(x+3,y+5,x+8,y+6,YELLOW);
        Line_Plus(x+9,y+6,x+6,y+9,YELLOW);
        Line_Plus(x+9,y+7,x+6,y+10,YELLOW);
    }
15. info.c
#include "headfile.h"

/*****
*****

* 函数名称      Info_Point
* 函数作用      绘制点位数据
* 函数输入      x,y 坐标, finish 完成点位数, point 总点位数
* 函数输出      无
*****
*****/

void Info_Point(int x,int y,int finish,int point)
{
    int i;
    Draw_Flag(x,y);
    for(i=0;i<point;i++)
    {
        Hollow_Bar(x+20+10*i,y+6,x+27+10*i,y+13,RED);
    }
    for(i=0;i<finish&& i<point;i++)
    {
        Solid_Bar(x+21+10*i,y+7,x+26+10*i,y+12,RED);
    }
}

/*****
*****

* 函数名称      Info_Time
* 函数作用      绘制时间数据
* 函数输入      x,y 坐标, hour 小时数, minute 分钟数
* 函数输出      无
*****
*****/

void Info_Time(int x,int y,int hour,int minute)
{
    if(minute!=-1)

```

```

    {
        Draw_Clock(x,y);
        Solid_Bar(x+20,y,x+59,y+19,BLACK);
        putsz(x+20,y+3,16,0,WHITE,hour);
        putsz(x+44-7,y+3,16,8,WHITE,minute);
        Solid_Bar(x+39-2,y+6,x+40-2,y+7,YELLOW);
        Solid_Bar(x+39-2,y+12,x+40-2,y+13,YELLOW);
    }
}

/*****
*****
* 函数名称      Info_Power
* 函数作用      绘制路程可视化
* 函数输入      x,y 坐标, hour 小时数, minute 分钟数
* 函数输出      无
*****
*****/

void Info_Power(int x,int y,int power)
{
    Draw_Power(x,y);
    Hollow_Bar(x+13,y+1,x+58,y+10,WHITE);
    if(power<=43)
    {
        Solid_Bar(x+15+power,y+2,x+57,y+9,BLACK);
        Solid_Bar(x+14,y+2,x+14+power,y+9,WHITE);
    }
}

/*****
*****
* 函数名称      Info_People
* 函数作用      绘制人物信息
* 函数输入      x,y 坐标, finish 完成点位数, point 总点位数, hour 小
时数, minute 分钟数, power 路程完成度, age 年龄, color 衣服颜色
* 函数输出      无
*****
*****/

void Info_People(int x,int y,int finish,int point,int hour,int mi
nute,int power,int age,int color)
{

```

```

switch(age)
{
    case CHILD:
        Draw_Child(x,y,color);
        break;
    case TEEN:
        Draw_Teen(x,y,color);
        break;
    case MID:
        Draw_Mid(x,y,color);
        break;
    case OLD:
        Draw_Old(x,y,color);
        break;
}
Info_Power(x+0,y+48,power);
if(finish==0)
{
    Draw_Clock(x,y+60);
    puthz(x+20,y+63,16,16,WHITE,"准备");
}
if(finish<point)
{
    Info_Time(x+0,y+60,hour,minute);
}
Info_Point(x+0,y+80,finish,point);
}

/*****
*****
* 函数名称      Info
* 函数作用      绘制信息
* 函数输入      finish 完成点位数, point 总点位数, hour 小时数,
minute 分钟数, power 路程完成度, age 年龄, color 衣服颜色, number 人数
* 函数输出      无
*****
*****/

void Info(int finish[],int point,int hour[],int minute[],int power[],int age,int color[],int number)
{
    int i;
    int x=0,y=54;

```

```

        for(i=0;i<number;i++)
        {
            Info_People(x,y+118*i,finish[i],point,hour[i],minute[i],power[i],age,color[i]);
            putsz(x+24,y+118*i,32,32,WHITE,i+1);
        }
    }
}

```

```

/*****
*****/

```

```

* 函数名称      Time_Clac
* 函数作用      时间计算
* 函数输入      hour 小时地址, minute 分钟地址, person_competition
竞赛信息, clock_hour 当前小时, clock_minute 当前分钟
* 函数输出      无

```

```

*****/

```

```

void Time_Clac(int *hour,int *minute,people_competition person_competition,int clock_hour,int clock_minute)
{
    int min_temp;
    if(Clock_Compare(clock_hour,clock_minute,person_competition.hour_point[0],person_competition.minute_point[0])==-1)
    {
        min_temp=Clock_Minus(clock_hour,clock_minute,person_competition.hour_point[0],person_competition.minute_point[0]);
        *hour=min_temp/60;
        *minute=min_temp%60;
    }
    else
    {
        *hour=-1;
        *minute=-1;
    }
}

```

```

/*****
*****/

```

```

* 函数名称      Distance_Clac
* 函数作用      路程计算
* 函数输入      map 路程数值, direction 方向数组, map_now 当前路程数

```

值, *people* 人数

* 函数输出 无

```
*****
*****/
```

```
void Distance_Clac(int map[],int **direction,int map_now[],int people)
```

```
{
    int i,j,distance=0;
    for(i=0;i<people;i++)
    {
        for(j=0;direction[i][j]!=0;j++);
        map[i]=(int)((float)map_now[i]/(float)j*44);
    }
}
```

```
/*
*****
*****
```

```
* 函数名称            Info_Player
* 函数作用            玩家高亮显示
* 函数输入            your_number 玩家序号
* 函数输出            无
```

```
*****
*****/
```

```
void Info_Player(int your_number)
```

```
{
    int x=0,y=54;
    static char frame=0;
    if(frame==0)
    {
        Hollow_Bar(x,y+118*your_number,x+59,y+118*your_number+99,
GOLDEN);
        frame++;
    }
    else
    {
        Hollow_Bar(x,y+118*your_number,x+59,y+118*your_number+99,
BLACK);
        frame--;
    }
}
```

16. loding.c

```

#include"headfile.h"

/*****
*****
* @author          ytm
* @date            2022-4-19b
*****
*****/

/*****
*****
* 函数名称          Loading_Welcome
* 函数作用          欢迎界面的加载动画
* 函数输入          x0, y0 加载动画圆圈旋转中心坐标, time_ms 加载动画时
长
* 函数输出          无
*****
*****/
void Loading_Welcome(int x0,int y0,int time_ms)
{
    int num = time_ms / 100;
    //caculate numbers of circulation
    int i,x[16],y[16];
    int head=7,back=0;
    int remainder = 324 % num;
    int every_length = (324 - remainder) / num;
    int temp = x0-162;
    //locate circle x
    x[0]=x0;
    x[1]=x0-3;
    x[2]=x0-6;
    x[3]=x0-9;
    x[4]=x0-9;
    x[5]=x0-9;
    x[6]=x0-6;
    x[7]=x0-3;
    x[8]=x0;
    x[9]=x0+3;
    x[10]=x0+6;
    x[11]=x0+9;
    x[12]=x0+9;
    x[13]=x0+9;
    x[14]=x0+6;

```

```

x[15]=x0+3;
//locate circle y
y[0]=y0-9;
y[1]=y0-9;
y[2]=y0-6;
y[3]=y0-3;
y[4]=y0;
y[5]=y0+3;
y[6]=y0+6;
y[7]=y0+9;
y[8]=y0+9;
y[9]=y0+9;
y[10]=y0+6;
y[11]=y0+3;
y[12]=y0;
y[13]=y0-3;
y[14]=y0-6;
y[15]=y0-9;
//init
for(i=0;i<=7;i++)
{
    Square(x[i],y[i],3,LIGHT_GRAY);
}
Hollow_Bar(x0-164,y0+71,x0+163,y0+82,LIGHT_GRAY);
//animation
for(i=0;i<remainder;i++)
{
    if (head==15)
        head=0;
    else
        head=head+1;
    if (back==15)
        back=0;
    else
        back=back+1;
    Square(x[head],y[head],3,BLACK);
    Square(x[back],y[back],3,LIGHT_GRAY);

    delay(100);
    Solid_Bar(temp,y0+73,temp+every_length,y0+80,LIGHT_GRAY);
    temp=temp+1+every_length;
}
for(;i<num;i++)

```

```

    {
        if (head==15)
            head=0;
        else
            head=head+1;
        if (back==15)
            back=0;
        else
            back=back+1;
        Square(x[head],y[head],3,BLACK);
        Square(x[back],y[back],3,LIGHT_GRAY);
        delay(100);

        Solid_Bar(temp,y0+73,temp+every_length-
1,y0+80,LIGHT_GRAY);
        temp=temp+every_length;
    }
}

/*****
*****
* 函数名称      Loading_Play
* 函数作用      游戏界面的加载动画
* 函数输入      x0 加载动画文字中心 x 值, y0 加载动画文字左上角 y 值,
people 人数, age 人物年龄, zone 区域类型, weather 天气类型
*               point 点位数量, point_x 点位 x 值地址, point_y 点位
y 值地址, clock_hour 小时地址, clock_minute 分钟地址
*               clock_hour_start, clock_minute_start 比赛开始时间
地址
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数注意      x0, y0 加载动画文字左上角坐标, x0 为 512, y0 为 372
* 函数输出      无
*****
*****/
void Loading_Play(int x0,int y0,int people,int age,int zone,int w
eather,int point,int* point_x,int* point_y,int* clock_hour,int* c
lock_minute,int *clock_hour_start,int *clock_minute_start,people_
basic *person_basic,people_competition *person_competition)
{
    int i,every_length = 20,temp = x0-162;
    int flag_gui = FALSE,flag_debug = FALSE;
    //init

```



```

Hollow_Bar(x0-164,y0+71,x0+163,y0+82,LIGHT_GRAY);
//animation and random operation
Solid_Bar(x0-96,y0,x0+96,y0+24,BLACK);
puthz(x0-96,y0,24,24,WHITE,"正在计算随机时间");
Clock_Random(clock_hour,clock_minute,clock_hour_start,clock_m
inute_start);
    for(i=0;i<4;i++)
    {
        delay(100);
        Solid_Bar(temp,y0+73,temp+every_length,y0+80,LIGHT_GRAY);
        temp=temp+1+every_length;
    }
    Solid_Bar(x0-96,y0,x0+96,y0+24,BLACK);
    puthz(x0-96,y0,24,24,WHITE,"正在生成随机地图");
    Map_Random(zone,point,point_x,point_y,weather,flag_gui,flag_d
ebug);
    for(;i<8;i++)
    {
        delay(90);
        Solid_Bar(temp,y0+73,temp+every_length-
1,y0+80,LIGHT_GRAY);
        temp=temp+every_length;
    }
    Solid_Bar(x0-96,y0,x0+96,y0+24,BLACK);
    puthz(x0-96,y0,24,24,WHITE,"正在部署随机选手");
    People_Random(people,age,weather,(*clock_hour),(*clock_minute
),person_basic,person_competition,flag_debug);
    for(;i<12;i++)
    {
        delay(100);
        Solid_Bar(temp,y0+73,temp+every_length-
1,y0+80,LIGHT_GRAY);
        temp=temp+every_length;
    }
    Solid_Bar(x0-96,y0,x0+96,y0+24,BLACK);
    puthz(x0-96,y0,24,24,WHITE,"正在分析随机地图");
}

```

/*****
 *****/

* 函数名称	<i>Loading_Search</i>
* 函数作用	搜索界面的加载动画
* 函数输入	<i>x0, y0</i> 加载动画圆圈旋转中心坐标, <i>people</i> 人数, <i>age</i> 人

物年龄, *zone* 区域类型, *weather* 天气类型, *point* 点位数量

* *clock_hour_start*, *clock_minute_start* 比赛结束时间

地址

* *person_basic* 比赛成员基本信息结构体地址,

person_competition 比赛成员参赛信息结构体地址

* 函数输出 无

*****/

```

void Loading_Search(int x0,int y0,int people,int age,int zone,int
    weather,int point,int *clock_hour_end,int *clock_minute_end,peop
    le_basic *person_basic,people_competition *person_competition)
{
    int i;
    int every_length = 20;
    int temp = x0-162;
    //init
    Hollow_Bar(x0-164,y0+71,x0+163,y0+82,LIGHT_GRAY);
    //animation and anlysis operation
    Solid_Bar(x0-96,y0,x0+108,y0+24,BLACK);
    puthz(x0-96,y0,24,24,WHITE,"正在处理比赛信息");
    ClockEnd_Get(people,point,clock_hour_end,clock_minute_end,per
    son_basic,person_competition);
    for(i=0;i<4;i++)
    {
        delay(100);
        Solid_Bar(temp,y0+73,temp+every_length,y0+80,LIGHT_GRAY);
        temp=temp+1+every_length;
    }
    Solid_Bar(x0-96,y0,x0+108,y0+24,BLACK);
    puthz(x0-108,y0,24,24,WHITE,"正在处理参赛者数据");
    People_Interval(people,point,person_basic,person_competition)
;
    for(;i<8;i++)
    {
        delay(100);
        Solid_Bar(temp,y0+73,temp+every_length-
1,y0+80,LIGHT_GRAY);
        temp=temp+every_length;
    }
    Solid_Bar(x0-96,y0,x0+108,y0+24,BLACK);
    puthz(x0-108,y0,24,24,WHITE,"正在处理参赛者成绩");
    People_Score(people,point,person_basic,person_competition);
    for(;i<12;i++)

```

```

    {
        delay(100);
        Solid_Bar(temp,y0+73,temp+every_length-
1,y0+80,LIGHT_GRAY);
        temp=temp+every_length;
    }
    Solid_Bar(x0-96,y0,x0+108,y0+24,BLACK);
    puthz(x0-108,y0,24,24,WHITE,"正在处理参赛者名次");
    People_Rank(people,person_basic,person_competition);
    for(;i<16;i++)
    {
        delay(100);
        Solid_Bar(temp,y0+73,temp+every_length-
1,y0+80,LIGHT_GRAY);
        temp=temp+every_length;
    }
}

```

17. login.c

```

#include "headfile.h"

#define GUEST 0
#define USER 1
#define ADMIN 2

#define Buttom_x_left 256
#define Buttom_x_right 767
#define Buttom_y_up 192
#define Buttom_y_down 575
#define Message_x_left (Buttom_x_left+8)
#define Message_x_right (Buttom_x_right-8)
#define Message_y_up (Buttom_y_up+16+64+16)
#define Message_y_down Message_y_up+8+32+8+32+8+32+8+32+16

#define login_x0 Buttom_x_left+8
#define login_y0 Message_y_down+8
#define login_x1 512-4
#define login_y1 Buttom_y_down-8
#define out_x0 512+4
#define out_y0 Message_y_down+8
#define out_x1 Buttom_x_right-8
#define out_y1 Buttom_y_down-8
#define loginword_x0 Buttom_x_left+100
#define loginword_y0 Message_y_down+38

```

```

#define outword_x0 608
#define outword_y0 Message_y_down+8+30
#define input1_x0 Message_x_left+8+8
#define input1_y0 Message_y_up+8+32+8
#define input1_x1 Message_x_right-8-8
#define input1_y1 Message_y_up+8+32+8+32
#define input2_x0 Message_x_left+8+8
#define input2_y0 Message_y_up+8+32+8+32+8+32+8
#define input2_x1 Message_x_right-8-8
#define input2_y1 Message_y_up+8+32+8+32+8+32+8+32

#define register_x0 Buttom_x_right-8-100
#define register_y0 Buttom_y_up+8
#define register_x1 Buttom_x_right-8
#define register_y1 Message_y_up-8

#define regiword_x0 Buttom_x_right-8-100+50-32
#define regiword_y0 Buttom_y_up+32

#define msg_x0 312
#define msg_y0 309
#define msg_x1 712
#define msg_y1 459

/*****
*****
* 函数名称      Login
* 函数作用      登录界面
* 函数输入      page 显存页, your_name 用户名
* 函数输出      无
*****
*****/

void Login(int *page,char *your_name)
{
    int flag=0;
    char user_buffer[11]={0},pass_buffer[11]={0};
    Draw_Login();
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        if(MouseX>login_x0&&MouseX<login_x1&&MouseY>login_y0&&Mou

```

```

seY<login_y1)
    {
        if(mouse_press(login_x0,login_y0,login_x1,login_y1)==
2)
        {
            shape=3;
            if(flag==0);
            {
                mousehide(MouseX,MouseY);
                Button_Light_Login(1);
                resetMouse(MouseX,MouseY);
                flag=1;
            }
            continue;
        }
        else if(mouse_press(login_x0,login_y0,login_x1,login_
y1)==1)
        {
            if(Scan_Login(user_buffer,pass_buffer))
            {
                Uni_Msgbox_7();
                Login_State(1);
                strcpy(your_name,user_buffer);
                *page=0;
                delay(100);
                //go to menu page
                break;
            }
            else
            {
                Uni_Msgbox_8();
                Draw_Login();
            }
            delay(100);
        }
    }
    if(MouseX>out_x0&&MouseX<out_x1&&MouseY>out_y0&&MouseY<ou
t_y1)
    {
        if(mouse_press(out_x0,out_y0,out_x1,out_y1)==2)
        {
            shape=3;
            if(flag==0);

```

```

        {
            mousehide(MouseX,MouseY);
            Button_Light_Login(2);
            resetMouse(MouseX,MouseY);
            flag=2;
        }
        continue;
    }
    else if(mouse_press(out_x0,out_y0,out_x1,out_y1)==1)
    {
        Darken_Login();
        *page=0;
        //go to menu page
        break;
    }
}
if(MouseX>input1_x0&&MouseX<input1_x1&&MouseY>input1_y0&&
MouseY<input1_y1)
{
    if(mouse_press(input1_x0,input1_y0,input1_x1,input1_y
1)==2)
    {
        shape=2;
        continue;
    }
    else if(mouse_press(input1_x0,input1_y0,input1_x1,inp
ut1_y1)==1)
    {
        Button_Draw(Message_x_left+8+8,Message_y_up+8+32+
8,Message_x_right-8-8,Message_y_up+8+32+8+32,DARK_GRAY,GRAY);
        Input_Username(user_buffer);
        resetMouse(MouseX,MouseY);
        delay(100);
    }
}
if(MouseX>input2_x0&&MouseX<input2_x1&&MouseY>input2_y0&&
MouseY<input2_y1)
{
    if(mouse_press(input2_x0,input2_y0,input2_x1,input2_y
1)==2)
    {
        shape=2;
        continue;
    }
}

```

```

    }
    else if(mouse_press(input2_x0,input2_y0,input2_x1,input2_y1)==1)
    {
        Button_Draw(input2_x0,input2_y0,input2_x1,input2_y1,DARK_GRAY,GRAY);
        Input_Password(pass_buffer);
        resetMouse(MouseX,MouseY);
        delay(100);
    }
}
if(MouseX>register_x0&&MouseX<register_x1&&MouseY>register_y0&&MouseY<register_y1)
{
    if(mouse_press(register_x0,register_y0,register_x1,register_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Login(3);
            resetMouse(MouseX,MouseY);
            flag=3;
        }
        continue;
    }
    else if(mouse_press(register_x0,register_y0,register_x1,register_y1)==1)
    {
        Button_Draw(register_x0,register_y0,register_x1,register_y1,DARK_GRAY,GRAY);
        *page=11;
        delay(100);
        //go to register page
        break;
    }
}
if(flag!=0)
{
    mousehide(MouseX,MouseY);
    Button_Darken_Login(flag);
    resetMouse(MouseX,MouseY);
}

```

```

        flag=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
}

```

```

/*****
*****
* 函数名称      Login_Check
* 函数作用      登录检测
* 函数输入      page 显存页
* 函数输出      无
*****
*****/

```

```

void Login_Check(int *page)
{
//    puthz(32,767-40,32,32,WHITE,"调试模式: 按返回进入主界面");
//    Login(&page);
//    Input_Username();
    Solid_Bar(0,0,1023,767,BLACK);
}

```

```

/*****
*****
* 函数名称      Scan_Login
* 函数作用      登录信息扫描
* 函数输入      user_buffer 用户名缓存, pass_buffer 密码缓存
* 函数输出      无
*****
*****/

```

```

int Scan_Login(char* user_buffer,char* pass_buffer)
{
    FILE* fp;
    int usernum=0;
    int i=0;
    int* p=&usernum;
    char user_lib[11]={0},pass_lib[11]={0};

```



```

    if((fp=fopen(".\\DATA\\ALL.DAT", "rb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    fseek(fp,2L,SEEK_SET);
    fread(p,sizeof(int),1,fp);
    fclose(fp);
    if((fp=fopen(".\\DATA\\USER.DAT", "rb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    for(i=0;i<usernum;i++)
    {
        fseek(fp,(long)(22*i),SEEK_SET);
        fread(user_lib,11,1,fp);
        fseek(fp,(long)(22*i+11),SEEK_SET);
        fread(pass_lib,11,1,fp);
        if(strcmp(user_buffer,user_lib)==0&&strcmp(pass_buffer,pa
ss_lib)==0)
        {
            fclose(fp);
            return 1;
        }
    }
    fclose(fp);
    return 0;
}

```

```

/*****
*****
* 函数名称      Login_State
* 函数作用      登陆状态检测
* 函数输入      登陆修改
* 函数输出      登录状态
*****
*****/

```

```

char Login_State(char i)
{
    static char state=0;
    if(EXP) state=1;
    if(i==1)
    {

```

```

        state=1;
    }
    return state;
}

/*****
*****
* 函数名称      Create_Admin
* 函数作用      管理员创建函数
* 函数输入      无
* 函数输出      无
*****
*****/

void Create_Admin()
{
    FILE* fp;
    char user_buffer[11]="admin",pass_buffer[11]="admin",u_buffer[11]="user",p_buffer[11]="user";
    if((fp=fopen(".\\DATA\\user.data","wb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    fwrite(user_buffer,11,1,fp);
    fseek(fp,11,SEEK_SET);
    fwrite(pass_buffer,11,1,fp);
    fseek(fp,22,SEEK_SET);
    fwrite(u_buffer,11,1,fp);
    fseek(fp,33,SEEK_SET);
    fwrite(p_buffer,11,1,fp);
    fclose(fp);
}

```

18. main.c

```

#include"headfile.h"

/*****
*****
* @author      ytm
* @date        2022-4-12
*****
*****/

/*****

```

```

*****
* 函数名称      Main
* 函数作用      主函数
* 函数输入      无
* 函数输出      无
*****
*****/

void main()
{
    int page=5;
    int people=1,age=0,zone=0,weather=0,point=2;
    int clock_hour,clock_minute,clock_hour_start=0,clock_minute_s
tart=0,clock_hour_end=0,clock_minute_end=0;
    int    point_x[4],point_y[4];
    int your_number;
    char your_name[11]={0};
    people_basic person_basic[6];
    people_competition person_competition[6];
    SetSVGA64k();
    if(!EXP) Welcome();
    mouseInit(&MouseX,&MouseY,&press);
    press=2;
    Login_State(0);
    while(1)
    {
        switch (page)
        {
            case 0:
                Menu(&page);
                //menu page
                delay(100);
                break;
            case 1:
                Help(&page);
                //help page
                delay(100);
                break;
            case 2:
                Setting(&page,&people,&age,&zone,&weather,&point,poin
t_x,point_y,&clock_hour,&clock_minute,&clock_hour_start,&clock_mi
nute_start,person_basic,person_competition);
                //play_setting page
                delay(100);

```

```

        break;
        case 3:
            Play(&page,&people,&age,&zone,&weather,&point,point_x
,point_y,&clock_hour,&clock_minute,&clock_hour_end,&clock_minute_
end,person_basic,person_competition,&your_number);
            //play_simulating page
            delay(100);
            break;
        case 4:
            Search(&page,&people,&age,&zone,&weather,&point,clock
_hour_start,clock_minute_start,clock_hour_end,clock_minute_end,pe
rson_basic,person_competition,&your_number,your_name);
            //outcome and searching page
            delay(100);
            break;
        case 5:
            SetSVGA64k();
            page=0;
            //init graph after init mouse
            break;
        case 8:
            About(&page);
            //about page
            delay(100);
            break;
        case 9:
            Quit(&page);
            //quit page
            delay(100);
            break;
        case 10:
            Login(&page,your_name);
            delay(100);
            break;
        case 11:
            Register(&page);
            delay(100);
            break;
    }
}
}

```

19. map.c

```

#include"headfile.h"

/*****
*****
* @author          ytm
* @date            2022-4-19
*****
*****/

int map_display[40][28];
int map_process[40][28];
//0: accessible, 1: inaccessible

/*****
*****
* 函数名称          Map_Random_Zone1
* 函数作用          随机生成城区地图
* 函数输入          x0, y0 主路变量地址
*                  flag_gui 过程可视化开关
* 函数注意          flag_gui 开启过程可视化
* 函数输出          无
*****
*****/

void Map_Random_Zone1(int* x0,int* y0,int flag_gui)
{
    int x,y;          //main road
    int x1,y1,x2,y2,x3,y3,x4,y4;  //lrud road
    int lr,ud,xr,yr;
    double k,b;        //river & bridge
    int lu=0,ld=0,ru=0,rd=0,xl,y1,lv,tf;  //lake
    int ts[40][28];    //tree & structure
    int xy1,xy2;       //make sure avenue available
    int lr_p1;         //point
    int i,j;           //temp use
River_illegal:
    //init
    randomize();
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Main Road");
    }
    //main vertical road

```

```

x=random(20)+10;
Main_Road_Convert(x,V_ERTICAL,2);
for(j=0;j<28;j++)
{
    map_display[x][j]=3;
    map_display[x+1][j]=3;
}
(*x0)=x;
//main level road
y=random(6)+16;
Main_Road_Convert(y,H_ORIZONTAL,2);
for(i=0;i<40;i++)
{
    map_display[i][y]=3;
    map_display[i][y+1]=3;
}
(*y0)=y;
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Other Road");
}
//left road: 4-23
y1=random(16)+(y+3);
if(y1>23)
{
    y1-=20;
}
for(i=0;i<x;i++)
{
    map_display[i][y1]=3;
}
Narrow_Road_Convert(y1,L_EFT,2);
if(y1<y)
{
    lu++;
}
else
{
    ld++;
}
//right road
y2=random(16)+(y+3);

```

```

if(y2>23)
{
    y2-=20;
}
//while(y2>=y1-1&& y2<=y1+1)
for(i=x+2;i<40;i++)
{
    map_display[i][y2]=3;
}
Narrow_Road_Convert(y2,R_IGHT,2);
if(y1<y)
{
    ru++;
}
else
{
    rd++;
}
//up road
x1=random(28)+(x+3);
if(x1>35)
{
    x1-=32;
}
for(j=0;j<y;j++)
{
    map_display[x1][j]=3;
}
Narrow_Road_Convert(x1,U_P,2);
if(x1<x)
{
    lu++;
}
else
{
    ru++;
}
//down Load
x2=random(28)+(x+3);
if(x2>35)
{
    x2-=32;
}

```

```

//while(x2>=x1-1&& x2<=x1+1)
for(j=y+2;j<28;j++)
{
    map_display[x2][j]=3;
}
Narrow_Road_Convert(x2,D_OWN,2);
if(x2<x)
{
    ld++;
}
else
{
    rd++;
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"River");
}
//river
lr=random(2);
//ld || ru
if(lr) //ru (xr,0) (39,yr)
{
    //y=k*x+b
    xr=random(30)+6;
    yr=random(21)+4;
    k=(yr)/(39.0-xr);
    b=-k*(xr);
    i=39;
    j=yr;
    for(;i>=0;i--)
    {
        j=k*i+b;
        j=j-1;
        if(j>=0&&j<=27)
        {
            if(map_display[i][j]==3)
            {
                map_display[i][j]=6;
            }
            else
            {

```



```

        map_display[i][j]=4;
    }
}
j=j+1;
if(j>=0&&j<=27)
{
    if(map_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
    else
    {
        map_display[i][j]=4;
    }
}
j=j+1;
if(j>=0&&j<=27)
{
    if(map_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
    else
    {
        map_display[i][j]=4;
    }
}
//river width is 3
}
}
else //ld (xr,27) (0,yr)
{
    //y=k*x+b
    xr=random(30)+6;
    yr=random(21)+4;
    k=(27.0-yr)/(xr);
    b=yr;
    i=0;
    j=yr;
    for(;i<40;i++)
    {
        j=k*i+b;
        if(j>=0&&j<=27)

```

```

    {
        if(map_display[i][j]==3)
        {
            map_display[i][j]=6;
        }
        else
        {
            map_display[i][j]=4;
        }
    }
    j=j+1;
    if(j>=0&&j<=27)
    {
        if(map_display[i][j]==3)
        {
            map_display[i][j]=6;
        }
        else
        {
            map_display[i][j]=4;
        }
    }
    j=j+1;
    if(j>=0&&j<=27)
    {
        if(map_display[i][j]==3)
        {
            map_display[i][j]=6;
        }
        else
        {
            map_display[i][j]=4;
        }
    }
    //river width is 3
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Lake");
}
//Lake

```

```

lv=random(2);
if(lv)
{
    if(lr)
    {
        //level down
        x1=random(x-2);
        if((lr==0)&&(x1>xr-3)&&(x1<xr+3))
        {
            x1=random(x-2);
        }
        for(i=0;i<x;i++)
        {
            for(j=25;j<28;j++)
            {
                if(((i-x1)*(i-x1)+(27-j)*(27-j))<=4)
                {
                    if(map_display[i][j]==3)
                    {
                        map_display[i][j]=6;
                    }
                    else if(map_display[i][j]==0)
                    {
                        map_display[i][j]=5;
                    }
                }
            }
        }
    }
    else
    {
        //level up
        x1=random(35-x)+x+4;
        if((lr)&&(x1>xr-3)&&(x1<xr+3))
        {
            x1=random(35-x)+x+4;
        }
        for(i=x+2;i<40;i++)
        {
            for(j=0;j<3;j++)
            {
                if(((i-x1)*(i-x1)+j*j)<=4)
                {

```

```

        if(map_display[i][j]==3)
        {
            map_display[i][j]=6;
        }
        else if(map_display[i][j]==0)
        {
            map_display[i][j]=5;
        }
    }
}
}
else
{
    if(lr)
    {
        //verticle left
        yl=random(24-y)+y+4;
        if((lr==0)&&(yl>yr-3)&&(xl<yr+3))
        {
            yl=random(24-y)+y+4;
        }
        for(i=0;i<3;i++)
        {
            for(j=y+2;j<28;j++)
            {
                if((i*i+(j-yl)*(j-yl))<=4)
                {
                    if(map_display[i][j]==3)
                    {
                        map_display[i][j]=6;
                    }
                    else if(map_display[i][j]==0)
                    {
                        map_display[i][j]=5;
                    }
                }
            }
        }
    }
}
else
{

```

```

        //verticle right
        yl=random(y-2);
        if((lr)&&(yl>yr-3)&&(xl<yr+3))
        {
            yl=random(y-2);
        }
        for(i=25;i<28;i++)
        {
            for(j=0;j<y;j++)
            {
                if(((i-39)*(i-39)+(j-yl)*(j-yl))<=4)
                {
                    if(map_display[i][j]==3)
                    {
                        map_display[i][j]=6;
                    }
                    else if(map_display[i][j]==0)
                    {
                        map_display[i][j]=5;
                    }
                }
            }
        }
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Bridge");
    }
    //bridge
    for(i=0;i<40;i++)
    {
        if(map_display[i][y]==6&&map_display[i][y+1]==3)
        {
            map_display[i][y+1]=6;
        }
        if(map_display[i][y]==3&&map_display[i][y+1]==6)
        {
            map_display[i][y]=6;
        }
    }
    for(j=0;j<28;j++)

```

```

{
    if(map_display[x][j]==6&&map_display[x+1][j]==3)
    {
        map_display[x+1][j]=6;
    }
    if(map_display[x][j]==3&&map_display[x+1][j]==6)
    {
        map_display[x][j]=6;
    }
}
//check river
if(map_display[x][y]==6||map_display[x+1][y]==6||map_display[
x][y+1]==6||map_display[x+1][y+1]==6)
{
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            map_display[i][j]=0;
        }
    }
    goto River_illegal;
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Tree Stru");
}
//tree & structure
for(i=0;i<40;i++)
{
    for(j=0;j<28;j++)
    {
        ts[i][j]=random(3);
    }
}
for(i=0;i<40;i++)
{
    for(j=0;j<28;j++)
    {
        if(ts[i][j])    //structure more
        {
            if(map_display[i][j]==0)

```

```

        {
            map_display[i][j]=1;
        }
    }
    else
    {
        if(map_display[i][j]==0)
        {
            map_display[i][j]=2;
        }
    }
}
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Add Road");
}
//make sure avenue available: clear starting point and ending
point, create 2 additional road
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        if(map_display[i][j]==1)
        {
            map_display[i][j]=2;
        }
        else if(map_display[i][j]==4)
        {
            map_display[i][j]=6;
        }
    }
}
}
for(i=37;i<40;i++)
{
    for(j=25;j<28;j++)
    {
        if(map_display[i][j]==1)
        {
            map_display[i][j]=2;
        }
        else if(map_display[i][j]==4)

```

```

        {
            map_display[i][j]=6;
        }
    }
}
xy1=random(3);
xy2=random(3);
if(lr)    //river ru
{
    //lu spread down
    i=xy1;
    j=3;
    map_display[i][0]=3;
    map_display[i][1]=3;
    map_display[i][2]=3;
    while(map_display[i][j]!=3&&map_display[i][j]!=6)
    {
        map_display[i][j]=3;
        j++;
    }
    Added_Road_Convert(xy1,V_ERTICAL,2);
    //rd spread Left
    i=36;
    j=27-xy2;
    map_display[37][j]=3;
    map_display[38][j]=3;
    map_display[39][j]=3;
    while(map_display[i][j]!=3&&map_display[i][j]!=6)
    {
        map_display[i][j]=3;
        i--;
    }
    Added_Road_Convert(j,H_ORIZONTAL,2);
}
else    //river ld
{
    //lu spread right
    i=3;
    j=xy1;
    map_display[0][j]=3;
    map_display[1][j]=3;
    map_display[2][j]=3;
    while(map_display[i][j]!=3&&map_display[i][j]!=6)

```



```

        {
            map_display[i][j]=3;
            i++;
        }
        Added_Road_Convert(xy1,H_ORIZONTAL,2);
        //rd spread up
        i=39-xy2;
        j=24;
        map_display[i][25]=3;
        map_display[i][26]=3;
        map_display[i][27]=3;
        while(map_display[i][j]!=3&&map_display[i][j]!=6)
        {
            map_display[i][j]=3;
            j--;
        }
        Added_Road_Convert(xy2,V_ERTICAL,2);
    }
}

/*****
*****
* 函数名称      Map_Random_Zone2
* 函数作用      随机生成郊区地图
* 函数输入      flag_gui 过程可视化开关
* 函数注意      flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Map_Random_Zone2(int flag_gui)
{
    int x,y,vl;          //road
    int lr,ud,x1,y1,x2,y2; //path
    int ph,xph[10],yph[10],xrh[12],yrh[12],tfph; //house
    & field
    int xr1,xr2,yr1,yr2;
    double k,b,m,c;      //river & bridge
    int tg[40][28];      //tree & grass
    int i,j;             //temp use
    //init
    randomize();
    if(flag_gui)
    {

```

```

        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Road 8");
    }
    //road 8
    v1=random(2);
    if(v1)    //vertical road
    {
        x=random(21)+10;
        for(j=0;j<28;j++)
        {
            map_display[x][j]=8;
            map_display[x+1][j]=8;
        }
        Suburban_Road_Convert(x,V_ERTICAL,2);
    }
    else    //level road
    {
        y=random(15)+7;
        for(i=0;i<40;i++)
        {
            map_display[i][y]=8;
            map_display[i][y+1]=8;
        }
        Suburban_Road_Convert(y,H_ORIZONTAL,2);
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Road 5");
    }
    //path 5
    lr=random(2);
    ud=random(2);
    if(v1)    //road vertical
    {
        y1=random(15)+7;
        if(lr)    //path1 left
        {
            for(i=x-1;i>=0;i--)
            {
                map_display[i][y1]=5;
            }
            Main_Road_Convert(y1,H_ORIZONTAL,2);
        }
    }

```

```

x2=random(x-8)+4;
if(ud)    //path 2 up
{
    for(j=y1;j>=0;j--)
    {
        map_display[x2][j]=5;
    }
    Main_Road_Convert(x2,V_ERTICAL,2);
}
else      //path 2 down
{
    for(j=y1;j<28;j++)
    {
        map_display[x2][j]=5;
    }
    Main_Road_Convert(x2,V_ERTICAL,2);
}
}
else      //path1 right
{
    for(i=x+2;i<40;i++)
    {
        map_display[i][y1]=5;
    }
    Main_Road_Convert(y1,H_ORIZONTAL,2);
    x2=random(32-x)+x+4;
    if(ud)    //path 2 up
    {
        for(j=y1;j>=0;j--)
        {
            map_display[x2][j]=5;
        }
        Main_Road_Convert(x2,V_ERTICAL,2);
    }
    else      //path 2 down
    {
        for(j=y1;j<28;j++)
        {
            map_display[x2][j]=5;
        }
        Main_Road_Convert(x2,V_ERTICAL,2);
    }
}
}

```

```

}
else    //road level
{
    x1=random(21)+10;
    if(ud)    //path1 up
    {
        for(j=y-1;j>=0;j--)
        {
            map_display[x1][j]=5;
        }
        y2=random(y-6)+3;
        Main_Road_Convert(x1,V_ERTICAL,2);
        Main_Road_Convert(y2,H_ORIZONTAL,2);
        if(lr)    //path 2 left
        {
            for(i=x1;i>=0;i--)
            {
                map_display[i][y2]=5;
            }
        }
        else    //path 2 right
        {
            for(i=x1;i<40;i++)
            {
                map_display[i][y2]=5;
            }
        }
    }
    else    //path1 down
    {
        for(j=y+2;j<28;j++)
        {
            map_display[x1][j]=5;
        }
        y2=random(23-y)+y+3;
        Main_Road_Convert(x1,V_ERTICAL,2);
        Main_Road_Convert(y2,H_ORIZONTAL,2);
        if(lr)    //path 2 left
        {
            for(i=x1;i>=0;i--)
            {
                map_display[i][y2]=5;
            }
        }
    }
}

```

```

    }
    else    //path 2 right
    {
        for(i=x1;i<40;i++)
        {
            map_display[i][y2]=5;
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"House 3");
}
//house 3
ph=random(2);
if(v1)
{
    for(i=0;i<12;i++)
    {
        yrh[i]=random(28);
    }
    if(ud)
    {
        for(i=0;i<10;i++)
        {
            yph[i]=random(y1);
        }
    }
    else
    {
        for(i=0;i<10;i++)
        {
            yph[i]=random(27-y1)+y1+1;
        }
    }
}
else
{
    for(i=0;i<12;i++)
    {
        xrh[i]=random(40);
    }
}

```

```

    }
    if(lr)
    {
        for(i=0;i<10;i++)
        {
            xph[i]=random(x1);
        }
    }
    else
    {
        for(i=0;i<10;i++)
        {
            xph[i]=random(39-x1)+x1+1;
        }
    }
}
if(v1)
{
    //12 road house
    for(i=0;i<6;i++)
    {
        if(map_display[x-1][yrh[i]]!=5)
        {
            map_display[x-1][yrh[i]]=3;
        }
    }
    for(;i<12;i++)
    {
        if(map_display[x+2][yrh[i]]!=5)
        {
            map_display[x+2][yrh[i]]=3;
        }
    }
}
if(ud)
{
    //path house
    if(ph) //8 path house
    {
        for(i=0;i<4;i++) //left path house
        {
            map_display[x2-1][yph[i]]=3;
        }
        for(;i<8;i++) //right path house

```

```

        {
            map_display[x2+1][yph[i]]=3;
        }
    }
    else    //10 path house
    {

        for(i=0;i<4;i++)    //left path house
        {
            map_display[x2-1][yph[i]]=3;
        }
        for(;i<8;i++)    //right path house
        {
            map_display[x2+1][yph[i]]=3;
        }
        tfph=random(2);
        if(tfph)    //left path house
        {
            for(;i<10;i++)
            {
                map_display[x2-1][yph[i]]=3;
            }
        }
        else    //right path house
        {
            for(i=8;i<10;i++)
            {
                map_display[x2+1][yph[i]]=3;
            }
        }
    }
}
else
{
    //path house
    if(ph)    //8 path house
    {
        for(i=0;i<4;i++)    //left path house
        {
            map_display[x2-1][yph[i]]=3;
        }
        for(;i<8;i++)    //right path house
        {

```

```

        map_display[x2+1][yph[i]]=3;
    }
}
else    //10 path house
{
    for(i=0;i<4;i++)    //left path house
    {
        map_display[x2-1][yph[i]]=3;
    }
    for(;i<8;i++)    //right path house
    {
        map_display[x2+1][yph[i]]=3;
    }
    tfph=random(2);
    if(tfph)    //left path house
    {
        for(; i<10;i++)
        {
            map_display[x2-1][yph[i]]=3;
        }
    }
    else    //right path house
    {
        for(i=8;i<10;i++)
        {
            map_display[x2+1][yph[i]]=3;
        }
    }
}

}
}
else
{
    //12 road house
    for(i=0;i<6;i++)
    {
        if(map_display[xrh[i]][y-1]!=5)
        {
            map_display[xrh[i]][y-1]=3;
        }
    }
    for(;i<12;i++)

```



```

{
    if(map_display[xrh[i]][y+2]!=5)
    {
        map_display[xrh[i]][y+2]=3;
    }
}
if(lr)
{
    //path house
    if(ph) //8 path house
    {
        for(i=0;i<4;i++)//up path house
        {
            map_display[xph[i]][y2-1]=3;
        }
        for(;i<8;i++) //down path house
        {
            map_display[xph[i]][y2+1]=3;
        }
    }
    else //10 path house
    {
        for(i=0;i<4;i++)//up path house
        {
            map_display[xph[i]][y2-1]=3;
        }
        for(;i<8;i++) //down path house
        {
            map_display[xph[i]][y2+1]=3;
        }
        tfph=random(2);
        if(tfph) //up path house
        {
            for(;i<10;i++)
            {
                map_display[xph[i]][y2-1]=3;
            }
        }
        else //down path house
        {
            for(i=8;i<10;i++)
            {
                map_display[xph[i]][y2+1]=3;
            }
        }
    }
}

```

```

        }
    }
}
else
{
    //path house
    if(ph) //8 path house
    {
        for(i=0;i<4;i++)//up path house
        {
            map_display[xph[i]][y2-1]=3;
        }
        for(;i<8;i++) //down path house
        {
            map_display[xph[i]][y2+1]=3;
        }
    }
    else //10 path house
    {
        for(i=0;i<4;i++)//up path house
        {
            map_display[xph[i]][y2-1]=3;
        }
        for(;i<8;i++) //down path house
        {
            map_display[xph[i]][y2+1]=3;
        }
        tfph=random(2);
        if(tfph) //up path house
        {
            for(;i<10;i++)
            {
                map_display[xph[i]][y2-1]=3;
            }
        }
        else //down path house
        {
            for(i=8;i<10;i++)
            {
                map_display[xph[i]][y2+1]=3;
            }
        }
    }
}

```

```

        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Field 4");
}
//field 4
if(v1)
{
    if(lr)
    {
        if(ud)    //v L u
        {
            Field_Convert(0,0,x,y1,2);
            for(i=0;i<x;i++)
            {
                for(j=0;j<y1;j++)
                {
                    if(map_display[i][j]==0)
                    {
                        map_display[i][j]=4;
                    }
                }
            }
        }
        else    //v L d
        {
            Field_Convert(0,y1,x,27,2);
            for(i=0;i<x;i++)
            {
                for(j=y1;j<28;j++)
                {
                    if(map_display[i][j]==0)
                    {
                        map_display[i][j]=4;
                    }
                }
            }
        }
    }
}
else

```

```

{
    if(ud)  //v r u
    {
        Field_Convert(x,0,39,y1,2);
        for(i=x;i<40;i++)
        {
            for(j=0;j<y1;j++)
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=4;
                }
            }
        }
    }
    else  //v r d
    {
        Field_Convert(x,y1,39,27,2);
        for(i=x;i<40;i++)
        {
            for(j=y1;j<28;j++)
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=4;
                }
            }
        }
    }
}
else
{
    if(ud)
    {
        if(lr)  //l u l
        {
            Field_Convert(0,0,x1,y,2);
            for(i=0;i<x1;i++)
            {
                for(j=0;j<y;j++)
                {
                    if(map_display[i][j]==0)

```

```

        {
            map_display[i][j]=4;
        }
    }
}
else //l u r
{
    Field_Convert(x1,0,39,y,2);
    for(i=x1;i<40;i++)
    {
        for(j=0;j<y;j++)
        {
            if(map_display[i][j]==0)
            {
                map_display[i][j]=4;
            }
        }
    }
}
else
{
    if(lr) //l d l
    {
        Field_Convert(0,y,x1,27,2);
        for(i=0;i<x1;i++)
        {
            for(j=y;j<28;j++)
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=4;
                }
            }
        }
    }
    else //l d r
    {
        Field_Convert(x1,y,39,27,2);
        for(i=x1;i<40;i++)
        {
            for(j=y;j<28;j++)

```

```

        {
            if(map_display[i][j]==0)
            {
                map_display[i][j]=4;
            }
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"River6Bri7");
}
//river 6 & bridge 7
if(v1) //vertical road->level river
{
    yr1=random(25)+3;
    if(yr1<9)
    {
        yr2=random(22-yr1)+(yr1+6);
    }
    else if(yr1>21)
    {
        yr2=random(yr1-8)+3;
    }
    else
    {
        yr2=random(14)+(yr1+6);
        if(yr2>27)
        {
            yr2-=25;
        }
    }
    //y=kx+b
    k=(double)(yr2-yr1)/(39.0);
    b=yr1;
    for(i=0;i<40;i++)
    {
        j=k*i+b;
        if(map_display[i][j]==8||map_display[i][j]==5)
        {

```

```

        map_display[i][j]=7;
    }
    else if(map_display[i][j]==4||map_display[i][j]==0||m
ap_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
    j=j-1;
    if(map_display[i][j]==8||map_display[i][j]==5)
    {
        map_display[i][j]=7;
    }
    else if(map_display[i][j]==4||map_display[i][j]==0||m
ap_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
    j=j+2;
    if(map_display[i][j]==8||map_display[i][j]==5)
    {
        map_display[i][j]=7;
    }
    else if(map_display[i][j]==4||map_display[i][j]==0||m
ap_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
}
for(j=0;j<28;j++)
{
    if(map_display[x][j]==7&&map_display[x+1][j]!=7)
    {
        map_display[x+1][j]=7;
    }
    if(map_display[x][j]!=7&&map_display[x+1][j]==7)
    {
        map_display[x][j]=7;
    }
}
}
else    //level road->vertical river
{
    xr1=random(37)+3;

```

```

if(xr1<12)
{
    xr2=random(31-xr1)+(xr1+9);
}
else if(xr1>30)
{
    xr2=random(xr1-11)+3;
}
else
{
    xr2=random(20)+(x1+9);
    if(xr2>39)
    {
        xr2-=37;
    }
}
//x=my+c
m=(double)(xr2-xr1)/27.0;
c=xr1;
for(j=0;j<28;j++)
{
    i=m*j+c;
    if(map_display[i][j]==8||map_display[i][j]==5)
    {
        map_display[i][j]=7;
    }
    else if(map_display[i][j]==4||map_display[i][j]==0||m
ap_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
    i=i-1;
    if(map_display[i][j]==8||map_display[i][j]==5)
    {
        map_display[i][j]=7;
    }
    else if(map_display[i][j]==4||map_display[i][j]==0||m
ap_display[i][j]==3)
    {
        map_display[i][j]=6;
    }
    i=i+2;
    if(map_display[i][j]==8||map_display[i][j]==5)

```



```

        {
            map_display[i][j]=7;
        }
        else if(map_display[i][j]==4||map_display[i][j]==0||m
ap_display[i][j]==3)
        {
            map_display[i][j]=6;
        }
    }
    for(i=0;i<40;i++)
    {
        if(map_display[i][y]==7&&map_display[i][y+1]!=7)
        {
            map_display[i][y+1]=7;
        }
        if(map_display[i][y]!=7&&map_display[i][y+1]==7)
        {
            map_display[i][y]=7;
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Tree1Gra2");
}
//tree 1 & grass 2
for(i=0;i<40;i++)
{
    for(j=0;j<28;j++)
    {
        tg[i][j]=random(5);
    }
}
for(i=0;i<40;i++)
{
    for(j=0;j<28;j++)
    {
        if(tg[i][j]<3)
        {
            if(map_display[i][j]==0)
            {
                map_display[i][j]=2;
            }
        }
    }
}

```

```

        }
    }
    else
    {
        if(map_display[i][j]==0)
        {
            map_display[i][j]=1;
        }
    }
}
}

/*****
*****
* 函数名称      Map_Random_Zone3
* 函数作用      随机生成山区地图
* 函数输入      flag_gui 过程可视化开关
* 函数注意      flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Map_Random_Zone3(int flag_gui)
{
    int lr,xm1,xm2,xm[28];
    double m,c;          //mountain area
    int xr1,xr2,xr[28];   //river
    int xb1,yb1,xb2,yb2,ud,xb3,yb3;    //bridge
    int lrh,yh1,house,yh2;    //house
    int rl,xl,yl;           //lake
    int maxwidth,maxwidth_y,rma,xma,yma;    //marsh
    int tg[40][28];        //tree & grass
    int i,j,a,b;           //temp use
    //init
    randomize();
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"MountianAr");
    }
    //mountain area
    lr=random(2);
    lr=1;

```

```

if(lr)    //r (xm1,0) (xm2,27)
{
    //x=my+c
    xm1=random(10)+20;
    xm2=random(37-xm1)+xm1+1;
    m=(xm2-xm1)/27.0;
    c=xm1;
    for(j=0;j<28;j++)
    {
        i=m*j+c;
        xm[j]=i;
        a=i;
        for(;a<40;a++)
        {
            map_display[a][j]=10;
        }
    }
}
else    //l (xm1,0) (xm2,27)
{
    //x=my+c
    xm1=random(10)+10;
    xm2=random(xm1-3)+3;
    m=(xm2-xm1)/27.0;
    c=xm1;
    for(j=0;j<28;j++)
    {
        i=m*j+c;
        xm[j]=i;
        a=i;
        for(a=i;a>=0;a--)
        {
            map_display[a][j]=10;
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"River");
}
//river
if(lr)    //river left

```

```

{
    //x=my+c
    xr1=random(xm1-5)+4;
    xr2=random(xm2-4)+3;
    m=(xr2-xr1)/27.0;
    c=xr1;
    for(j=0;j<28;j++)
    {
        i=m*j+c;
        xr[j]=i;
        if(map_display[i][j]==0)
        {
            map_display[i][j]=4;
        }
        if(map_display[i-1][j]==0)
        {
            map_display[i-1][j]=4;
        }
        if(map_display[i+1][j]==0)
        {
            map_display[i+1][j]=4;
        }
    }
}
else    //river right
{
    //x=my+c
    xr1=random(38-xm1)+xm1+1;
    xr2=random(36-xm2)+xm2+1;
    m=(xr2-xr1)/27.0;
    c=xr1;
    for(j=0;j<28;j++)
    {
        i=m*j+c;
        xr[j]=i;
        if(map_display[i][j]==0)
        {
            map_display[i][j]=4;
        }
        if(map_display[i-1][j]==0)
        {
            map_display[i-1][j]=4;
        }
    }
}

```

```

        if(map_display[i+1][j]==0)
        {
            map_display[i+1][j]=4;
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Bridge");
}
//bridge
yb1=random(28);
if(yb1<6)
{
    yb2=random(23-yb1)+(yb1+6);
}
else if(yb1>22)
{
    yb2=random(yb1-5);
}
else
{
    yb2=random(17)+(yb1+6);
    if(yb2>27)
    {
        yb2-=28;
    }
}
//make sure avenue available: add bridge
if(yb1<7||yb2<7||yb1>21||yb2>21)
{
    //Legal
}
else
{
    ud=random(2);
    if(ud)        //add 0-6 bridge
    {
        yb3=random(7);
    }
    else        //add 22-27 bridge
    {

```

```

        yb3=random(6)+22;
    }
}
for(i=0;i<40;i++)
{
    if(map_display[i][yb1]==4)
    {
        map_display[i][yb1]=7;
    }
}
for(i=0;i<40;i++)
{
    if(map_display[i][yb2]==4)
    {
        map_display[i][yb2]=7;
    }
}
for(i=0;i<40;i++)
{
    if(map_display[i][yb3]==4)
    {
        map_display[i][yb3]=7;
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"House");
}
//house
lrh=random(2);
yh1=random(28);
while((yh1<yb1+2&&yh1>yb1-2)|| (yh1<yb2+2&&yh1>yb2-2))
{
    randomize();
    yh1=random(28);
}
i=xr[yh1];
if(lrh)    //right
{
    map_display[i+2][yh1]=8;
}
else    //left

```

```

{
    map_display[i-2][yh1]=8;
}
house=random(2);
if(house)    //add second house
{
    yh2=random(28);
    while((yh2<yb1+2&&yh2>yb1-2)|| (yh2<yb2+2&&yh2>yb2-2))
    {
        randomize();
        yh2=random(28);
    }
    i=xr[yh2];
    if(lrh)    //left
    {
        if(map_display[i-2][yh2]==0)
        {
            map_display[i-2][yh2]=8;
        }
        else if(map_display[i-2][yh2]==10)
        {
            map_display[i-2][yh2]=18;
        }
    }
    else    //right
    {
        if(map_display[i-2][yh2]==0)
        {
            map_display[i-2][yh2]=8;
        }
        else if(map_display[i-2][yh2]==10)
        {
            map_display[i-2][yh2]=18;
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Lake");
}
//Lake
rl=random(2)+2;

```

```

if(lr)      //lake left
{
    x1=0;
    y1=random(16)+6;
    for(i=0;i<=r1;i++)
    {
        for(j=0;j<28;j++)
        {
            if(((i-x1)*(i-x1)+(y1-j)*(y1-j))<=(r1*r1))
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=5;
                }
            }
        }
    }
}
else      //lake right
{
    x1=39;
    y1=random(16)+6;
    for(i=40-r1;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            if(((i-x1)*(i-x1)+(y1-j)*(y1-j))<=(r1*r1))
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=5;
                }
            }
        }
    }
}
if(flag_gui)
{
    Solid_Bar(5,5,246,29,BLACK);
    putzm(5,5,24,16,RED,"Marsh");
}
//marsh

```



```

if(lr)
{
    maxwidth=xm[0]-xr[0]-2;
    maxwidth_y=0;
    for(j=1;j<28;j++)
    {
        if((xm[j]-xr[j]-2)>=maxwidth)
        {
            maxwidth=xm[j]-xr[j]-2;
            maxwidth_y=j;
        }
    }
}
else
{
    maxwidth=xr[0]-xm[0]-2;
    maxwidth_y=0;
    for(j=1;j<28;j++)
    {
        if((xr[j]-xm[j]-2)>=maxwidth)
        {
            maxwidth=xr[j]-xm[j]-2;
            maxwidth_y=j;
        }
    }
}
if(maxwidth%2)
{
    a=(maxwidth-1)/2-1;
}
else
{
    a=maxwidth/2-2;
}
rma=random(a)+2;
b=xr[maxwidth_y]+2+rma;
a=(xm[maxwidth_y]-1-rma)-b+1;
xma=random(a)+b;
yma=maxwidth_y;
for(i=40;i<40;i++)
{
    for(j=0;j<28;j++)
    {

```

```

        if(((i-xma)*(i-xma)+(yma-j)*(yma-j))<=(rma*rma))
        {
            if(map_display[i][j]==0)
            {
                map_display[i][j]=6;
            }
        }
    }
}
if(maxwidth>=5)
{
    if(lr)
    {
        if(maxwidth%2)
        {
            a=(maxwidth-1)/2-1;
        }
        else
        {
            a=maxwidth/2-2;
        }
        rma=random(a)+2;
        b=xr[maxwidth_y]+2+rma;
        a=(xm[maxwidth_y]-1-rma)-b+1;
        xma=random(a)+b;
        yma=maxwidth_y;
        for(i=0;i<40;i++)
        {
            for(j=0;j<28;j++)
            {
                if(((i-xma)*(i-xma)+(yma-j)*(yma-
j))<=(rma*rma))
                {
                    if(map_display[i][j]==0)
                    {
                        map_display[i][j]=6;
                    }
                }
            }
        }
    }
    else
    {

```

```

        if(maxwidth%2)
        {
            a=(maxwidth-1)/2-1;
        }
        else
        {
            a=maxwidth/2-2;
        }
        rma=random(a)+2;
        b=xm[maxwidth_y]+2+rma;
        a=(xr[maxwidth_y]-1-rma)-b+1;
        xma=random(a)+b;
        yma=maxwidth_y;
        for(i=0;i<40;i++)
        {
            for(j=0;j<28;j++)
            {
                if(((i-xma)*(i-xma)+(yma-j)*(yma-
j))<=(rma*rma))
                {
                    if(map_display[i][j]==0)
                    {
                        map_display[i][j]=6;
                    }
                }
            }
        }
    }
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Tree Grass");

    }
    //tree & grass
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            tg[i][j]=random(5);
        }
    }
}

```

```

    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            if(tg[i][j]<3)
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=3;
                }
                else if(map_display[i][j]==10)
                {
                    map_display[i][j]=13;
                }
            }
            else
            {
                if(map_display[i][j]==0)
                {
                    map_display[i][j]=2;
                }
                else if(map_display[i][j]==10)
                {
                    map_display[i][j]=12;
                }
            }
        }
    }
}

/*****
*****
* 函数名称      Map_Random
* 函数作用      随机生成地图
* 函数输入      zone 区域类型, point 点位数量, point_x 点位x 值地
址, point_y 点位y 值地址, weather 天气
*               flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数注意      flag_gui 开启过程可视化
*               flag_debug 开启调试功能
* 函数输出      无
*****
*****/
void Map_Random(int zone,int point,int* point_x,int* point_y,int

```

```

weather,int flag_gui,int flag_debug)
{
    int i,j;
    int x0,y0;
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            map_display[i][j]=0;
        }
    }
    for(i=0;i<4;i++)
    {
        point_x[i]=0;
        point_y[i]=0;
    }
    //generate map
    switch (zone)
    {
        case 1:
        {
            Map_Random_Zone1(&x0,&y0,flag_gui);
            break;
        }
        case 2:
        {
            Map_Random_Zone2(flag_gui);
            break;
        }
        case 3:
        {
            Map_Random_Zone3(flag_gui);
            break;
        }
    }
    //analysis map
    Map_Analysis(zone,weather);
    //generate point
    Point_Random(zone,point,point_x,point_y,x0,y0,flag_gui,flag_d
ebug);
}

/*****

```

```

*****
* 函数名称          Point_Random
* 函数作用          随机生成地图
* 函数输入          zone 区域类型, point 点位数量, point_x 点位x 值地
址, point_y 点位y 值地址
*                  x, y 城区点位分析变量, flag_gui 过程可视化开关,
flag_debug 调试模式开关
* 函数注意          flag_gui 开启过程可视化
*                  flag_debug 开启调试功能
* 函数输出          无
*****
*****/
void Point_Random(int zone,int point,int* point_x,int* point_y,in
t x,int y,int flag_gui,int flag_debug)
{
    int i,lr_p1;
    FILE *fp;
    if(flag_gui)
    {
        Solid_Bar(5,5,246,29,BLACK);
        putzm(5,5,24,16,RED,"Point");
    }
    //init
    randomize();
    lr_p1=random(2);
    //generate start and end point
    switch (point)
    {
    case 2:
        {
            point_x[1]=39;
            point_y[1]=27;
            break;
        }
    case 3:
        {
            point_x[2]=39;
            point_y[2]=27;
            break;
        }
    case 4:
        {
            point_x[3]=39;

```

```

        point_y[3]=27;
        break;
    }
}
switch (zone)
{
case 1:
    {
        switch(point)
        {
        case 3:
            {
                if(lr_p1)
                {
                    point_x[1]=random(20)+10;
                    point_y[1]=random(2)+y;
                }
                else
                {
                    point_x[1]=random(2)+x;
                    point_y[1]=random(8)+14;
                }
            }
            break;
        case 4:
            {
                point_x[1]=random(2)+x;
                point_y[1]=random(8)+14;
                if(point_x[1]>20)
                {
                    //point1: right point
                    if(point_x[1]<7||point_x[1]>22)
                    {
                        point_x[2]=random(10)+10;
                    }
                    else if(point_x[1]<=13)
                    {
                        point_x[2]=random(16-
point_x[1])+point_x[1]+4;
                    }
                    else if(point_x[1]>=16)
                    {
                        point_x[2]=random(point_x[1]-13)+10;

```

```

    }
    else
    {
        point_x[2]=random(3)+point_x[1]+4;
        if(point_x[2]>19)
        {
            point_x[2]=point_x[2]-10;
        }
    }
}
else
{
    //point1: left point
    point_x[2]=random(10)+20;
    if(point_x[1]<17||point_x[1]>32)
    {
        point_x[2]=random(10)+20;
    }
    else if(point_x[1]<=23)
    {
        point_x[2]=random(26-
point_x[1])+point_x[1]+4;
    }
    else if(point_x[1]>=26)
    {
        point_x[2]=random(point_x[1]-23)+20;
    }
    else
    {
        point_x[2]=random(3)+point_x[1]+4;
        if(point_x[2]>29)
        {
            point_x[2]=point_x[2]-10;
        }
    }
}
if(point_y[1]==y)
{
    point_y[2]=y+1;
}
else if(point_y[1]==(y+1))
{
    point_y[2]=y;
}

```



```

        }
        else
        {
            point_y[2]=random(2)+y;
        }
    }
    break;
}
}
break;
default:
{
    switch(point)
    {
        case 3:
        {
            point_x[1]=random(21)+10;
            point_y[1]=random(8)+14;
            while(map_process[point_x[1]][point_y[1]]==1)
            {
                randomize();
                point_x[1]=random(21)+10;
                point_y[1]=random(8)+14;
            }
        }
        break;
        case 4:
        {
            point_x[1]=random(10)+10;
            point_y[1]=random(8)+14;
            while(map_process[point_x[1]][point_y[1]]==1)
            {
                randomize();
                point_x[1]=random(10)+10;
                point_y[1]=random(8)+14;
            }
            point_x[2]=random(11)+20;
            point_y[2]=random(8)+14;
            while(map_process[point_x[2]][point_y[2]]==1)
            {
                randomize();
                point_x[2]=random(11)+20;
                point_y[2]=random(8)+14;
            }
        }
    }
}

```

```

        }
    }
    break;
}
}
break;
}
//debug output
if(flag_debug)
{
    fp = fopen (".\\MAP\\POINT.txt","w+");
    fprintf(fp,"\\nPoint Information\\n");
    fprintf(fp,"\\npoint_x & point_y for obverse\\n");
    fprintf(fp,"x=%d\\ny=%d\\n",x,y);
    for(i=0;i<point;i++)
    {
        fprintf(fp,"point_x[%d]=%d\\n",i,point_x[i]);
        fprintf(fp,"point_y[%d]=%d\\n",i,point_y[i]);
    }
    fclose(fp);
}
}

/*****
*****
* 函数名称      Map_Analysis
* 函数作用      分析地图
* 函数输入      zone 区域类型, weather 天气类型
* 函数输出      无
*****
*****/
void Map_Analysis(int zone,int weather)
{
    int i,j;
    //init
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            map_process[i][j]=0;
            //default 0 mean can access
        }
    }
}

```

```

if(zone==1)
{
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            if(map_display[i][j]==1||map_display[i][j]==4||map_display[i][j]==5)
            {
                map_process[i][j]=1;
            }
        }
    }
}
else if(zone==2)
{
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            if(map_display[i][j]==3||map_display[i][j]==6)
            {
                map_process[i][j]=1;
            }
        }
    }
}
else
{
    if(weather==3)
    {
        for(i=0;i<40;i++)
        {
            for(j=0;j<28;j++)
            {
                if(map_display[i][j]==4||map_display[i][j]==5||map_display[i][j]==6||map_display[i][j]==8)
                {
                    map_process[i][j]=1;
                }
            }
        }
    }
}

```

```

else
{
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            if(map_display[i][j]==4||map_display[i][j]==5
||map_display[i][j]==8)
            {
                map_process[i][j]=1;
            }
        }
    }
}
}
}
}

```

```

/*****
*****

```

```

* 函数名称      Map_Display
* 函数作用      显示地图
* 函数输入      x0, y0, 地图显示左上角坐标, zone 区域类型, weather
天气类型, point 点位数量
*               point_x 点位x 值地址, point_y 点位y 值地址,
flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数注意      x0, y0 一般为63, 95
*               flag_debug 开启调试功能
*               flag_gui 开启过程可视化
* 函数输出      无

```

```

*****
*****/

```

```

void Map_Display(int x0,int y0,int zone,int weather,int point,int
* point_x,int* point_y,int flag_gui,int flag_debug,int clock_hour
)
{
    int i,j,k;
    FILE *fp;
    //draw map
    Sky(weather,clock_hour);
    Map_Create(x0,y0,zone);
    if(weather==RAINY)
    {
        Random_RainDrop();
    }
}

```

```

}
Flag_Create(point_x,point_y,point);
Map_Data_Save();
//debugging function
if(flag_debug)
{
    switch(zone)
    {
    case 1:
        {
            fp = fopen (".\\MAP\\MAP1.txt","w+");
        }
        break;
    case 2:
        {
            fp = fopen (".\\MAP\\MAP2.txt","w+");
        }
        break;
    case 3:
        {
            fp = fopen (".\\MAP\\MAP3.txt","w+");
        }
        break;
    }
    fprintf(fp,"Competition Information:\n");
    fprintf(fp,"\nzone=%d  weather=%d  point==%d\n",zone,weather,point);
    fprintf(fp,"Map Information: \n");
    fprintf(fp,"map_display[40][28] for eval\n");
    for(i=0;i<40;i++)
    {
        for(j=0;j<28;j++)
        {
            if(j==0)
            {
                if(map_display[i][j]<10)
                {
                    fprintf(fp,"{ %d,",map_display[i][j]);
                }
                else
                {
                    fprintf(fp,"{%d,",map_display[i][j]);
                }
            }
        }
    }
}

```

```

    }
    else if(j==27)
    {
        if(i==39)
        {
            if(map_display[i][j]<10)
            {
                fprintf(fp," %d}\n",map_display[i][j]
);
            }
            else
            {
                fprintf(fp,"%d}\n",map_display[i][j])
;
            }
        }
        else
        {
            if(map_display[i][j]<10)
            {
                fprintf(fp," %d},\n",map_display[i][j]
]);
            }
            else
            {
                fprintf(fp,"%d},\n",map_display[i][j]
);
            }
        }
    }
    else
    {
        if(map_display[i][j]<10)
        {
            fprintf(fp," %d,",map_display[i][j]);
        }
        else
        {
            fprintf(fp,"%d,",map_display[i][j]);
        }
    }
}
}
}

```

```

fprintf(fp, "\nmap_display[40][28] for obverse\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)
        {
            if(map_display[i][j]<10)
            {
                fprintf(fp, " %d\n",map_display[i][j]);
            }
            else
            {
                fprintf(fp, " %d\n",map_display[i][j]-10);
            }
        }
        else
        {
            if(map_display[i][j]<10)
            {
                fprintf(fp, " %d",map_display[i][j]);
            }
            else
            {
                fprintf(fp, " %d",map_display[i][j]-10);
            }
        }
    }
}
fprintf(fp, "\nmap_process[40][28] for evalue\n");
for(i=0;i<40;i++)
{
    for(j=0;j<28;j++)
    {
        if(j==0)
        {
            fprintf(fp, "{%d,",map_process[i][j]);
        }
        else if(j==27)
        {
            if(i==39)
            {
                fprintf(fp, "%d}\n",map_process[i][j]);
            }
        }
    }
}

```

```

        }
        else
        {
            fprintf(fp,"%d},\n",map_process[i][j]);
        }
    }
    else
    {
        fprintf(fp,"%d,",map_process[i][j]);
    }
}
}
fprintf(fp,"\nmap_process[40][28] for obverse\n");
for(j=0;j<28;j++)
{
    for(i=0;i<40;i++)
    {
        if(i==39)
        {
            fprintf(fp," %d\n",map_process[i][j]);
        }
        else
        {
            fprintf(fp," %d",map_process[i][j]);
        }
    }
}
fprintf(fp,"\npoint_x & point_y for evalule\n");
for(i=0;i<point;i++)
{
    fprintf(fp,"point_x[%d]=%d;\n",i,point_x[i]);
    fprintf(fp,"point_y[%d]=%d;\n",i,point_y[i]);
}
fclose(fp);
}
//gui function
if(flag_gui)
{
    for(j=0;j<28;j++)
    {
        for(i=0;i<40;i++)
        {
            //for test

```



```

        if(map_display[i][j]<10)
        {
            putsz(x0+i*24,y0+j*24,16,16,WHITE,map_display
[i][j]);
        }
        else
        {
            putsz(x0+i*24,y0+j*24,16,16,GREEN,(map_displa
y[i][j]-10));
        }
    }
}
for(k=0;k<point;k++)
{
    i=point_x[k];
    j=point_y[k];
    //for test
    putsz(x0+i*24,y0+j*24,16,16,RED,map_display[i][j]);
}
}
}

```

20. menu.c

```
#include"headfile.h"
```

```

/*****
*****
* @author          ytm
* @date            2022-4-19b
*****
*****/

```

```
//driver
```

```

#define All_x0      320
#define Play_y0     350
#define Text_xplus  160
#define Text_yplus  24

```

```

#define About_x0    961
#define About_y0    700
#define Login_x0    24
#define Login_y0    700

```

```
//follower
```

```

#define All_x1 (All_x0+383)
#define Play_y1 (Play_y0+79)
#define Help_y0 (Play_y0+85)
#define Help_y1 (Play_y0+165)
#define Quit_y0 (Play_y0+171)
#define Quit_y1 (Play_y0+251)

#define About_x1 (About_x0+39)
#define About_y1 (About_y0+39)
#define Login_x1 (Login_x0+39)
#define Login_y1 (Login_y0+39)

/*****
*****
* 函数名称      Menu
* 函数作用      菜单界面
* 函数输入      page 界面地址
* 函数输出      无
*****
*****/

void Menu(int *page)
{
    /*按钮状态变量*/
    int flag,flag_mouse,flag_mouse_light;
    /*右键菜单变量*/
    int xm1,ym1,xm2,ym2;
Refresh_Menu:
    //SetSVGA64k();
    Solid_Bar(0,0,1023,767,BLACK);
    flag=0,flag_mouse=0,flag_mouse_light=0;
    Draw_Menu();
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        /*刷新界面*/
        if(mouse_press(1,1,1023,767)==3)
        {
            mousehide(MouseX,MouseY);
            if(flag_mouse==0)
            {
                flag_mouse=1;
                /*根据鼠标位置绘制右键菜单*/

```

```

        if(MouseX>0&&MouseX<864&&MouseY>0&&MouseY<728)
        {
            xm1=MouseX,ym1=MouseY,xm2=MouseX+160,ym2=Mous
eY+40;
        }
        else if(MouseX>863&&MouseX<1024&&MouseY>0&&MouseY
<728)
        {
            xm1=MouseX-
160,ym1=MouseY,xm2=MouseX,ym2=MouseY+40;
        }
        else if(MouseX>0&&MouseX<864&&MouseY>727&&MouseY<
768)
        {
            xm1=MouseX,ym1=MouseY-
40,xm2=MouseX+160,ym2=MouseY;
        }
        else
        {
            xm1=MouseX-160,ym1=MouseY-
40,xm2=MouseX,ym2=MouseY;
        }
        save_image(xm1,ym1,xm2+3,ym2+3);
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    continue;
}
if(flag_mouse==1)
{
    if(mouse_out_press(xm1,ym1,xm2,ym2)==1)
    {
        mousehide(MouseX,MouseY);
        flag_mouse=0;
        printf_image(xm1,ym1,xm2+3,ym2+3);
        resetMouse(MouseX,MouseY);
    }
    if(mouse_press(xm1,ym1,xm2,ym2)==2)
    {
        shape=3;
        puthz(xm1+56,ym1+8,24,24,WHITE,"刷新");
        flag_mouse_light=1;
    }
}

```

```

        continue;
    }
    else if(mouse_press(xm1,ym1,xm2,ym2)==1)
    {
        press=0;
        shape=0;
        //refresh page
        delay(100);
        goto Refresh_Menu;
    }
    if(flag_mouse_light!=0)
    {
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        flag_mouse_light=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
//buttons
if(MouseX>All_x0&&MouseX<All_x1&&MouseY>Play_y0&&MouseY<P
lay_y1)
{
    if(mouse_press(All_x0,Play_y0,All_x1,Play_y1)==2)
    {
        shape=3;
        if(flag==0)
        {
            mousehide(MouseX,MouseY);
            Button_Light_Menu(1);
            resetMouse(MouseX,MouseY);
            flag=1;
        }
        continue;
    }
    else if(mouse_press(All_x0,Play_y0,All_x1,Play_y1)==1
)
    {
        *page=2;
        if(Login_State(0)==0)
        {
            Uni_Msgbox("请登录以进行模拟！");

```

```

        Draw_Menu();
        continue;
    }
    //go to play_setting page
    delay(100);
    break;
}
}
if(MouseX>All_x0&&MouseX<All_x1&&MouseY>Help_y0&&MouseY<H
elp_y1)
{
    if(mouse_press(All_x0,Help_y0,All_x1,Help_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Menu(2);
            resetMouse(MouseX,MouseY);
            flag=2;
        }
        continue;
    }
    else if(mouse_press(All_x0,Help_y0,All_x1,Help_y1)==1
)
    {
        *page=1;
        //go to help page
        delay(100);
        break;
    }
}
if(MouseX>All_x0&&MouseX<All_x1&&MouseY>Quit_y0&&MouseY<Q
uit_y1)
{
    if(mouse_press(All_x0,Quit_y0,All_x1,Quit_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Menu(3);
            resetMouse(MouseX,MouseY);

```

```

        flag=3;
    }
    continue;
}
else if(mouse_press(All_x0,Quit_y0,All_x1,Quit_y1)==1
)
{
    *page=9;
    //go to quit page
    Solid_Bar(All_x0,Quit_y1+3,All_x1,Quit_y1+15,BLAC
K);

    //hide mouse
    delay(100);
    break;
}
}
if(MouseX>About_x0&&MouseX<About_x1&&MouseY>About_y0&&Mou
seY<About_y1)
{
    if(mouse_press(About_x0,About_y0,About_x1,About_y1)==
2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Menu(4);
            resetMouse(MouseX,MouseY);
            flag=4;
        }
        continue;
    }
    else if(mouse_press(About_x0,About_y0,About_x1,About_
y1)==1)
    {
        *page=8;
        //go to about page
        delay(100);
        break;
    }
}
if(MouseX>Login_x0&&MouseX<Login_x1&&MouseY>Login_y0&&Mou
seY<Login_y1)

```

```

{
    if(mouse_press(Login_x0,Login_y0,Login_x1,Login_y1)==
2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Menu(5);
            resetMouse(MouseX,MouseY);
            flag=5;
        }
        continue;
    }
    else if(mouse_press(Login_x0,Login_y0,Login_x1,Login_
y1)==1)
    {
        *page=10;
        //hide mouse
        Solid_Bar(Login_x0,Login_y0+1,Login_x1,Login_y1+1
5,BLACK);

        Solid_Bar(Login_x0+1,Login_y0,Login_x1+9,Login_y1
+15,BLACK);

        Button_Darken_Menu(5);
        putzm(19,746,16,12,WHITE,"Hust");
        putzm(20,746,16,12,WHITE,"Hust");
        //go to login page
        delay(100);
        break;
    }
}
if(flag!=0)
{
    mousehide(MouseX,MouseY);
    Button_Darken_Menu(flag);
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    flag=0;
}

```

```

        if(shape!=0)
        {
            shape=0;
        }
    }
}

/*****
*****
* 函数名称      Draw_Menu
* 函数作用      绘制菜单界面
* 函数输入      无
* 函数输出      无
*****
*****/

void Draw_Menu()
{
    //refresh
    Solid_Bar(5,5,45,45,BLACK);
    //title and text
    printf_title(300,215,762,281);
    //Background
    Back();
    /*back up*/
    /*
    puthz(306,221,64,76,DARK_GRAY,"定向越野模拟");
    puthz(305,220,64,76,GRAY,"定向越野模拟");
    puthz(304,219,64,76,GRAY,"定向越野模拟");
    puthz(303,218,64,76,GRAY,"定向越野模拟");
    puthz(302,217,64,76,LIGHT_GRAY,"定向越野模拟");
    puthz(301,216,64,76,LIGHT_GRAY,"定向越野模拟");
    puthz(300,215,64,76,WHITE,"定向越野模拟");
    */
    //bold text 1
    CopyrightText_Draw(3,746,WHITE,BLACK);
    putzm(19,746,16,12,WHITE,"Hust AIA");
    CopyrightText_Draw(4,746,WHITE,BLACK);
    putzm(20,746,16,12,WHITE,"Hust AIA");
    //bold text 2
    putzm(933,746,16,12,WHITE,"v3.1.2b");
    putzm(934,746,16,12,WHITE,"v3.1.2b");
    //buttons: 384*80 DARK_GRAY + LIGHT_GRAY
    Button_Draw(All_x0,Play_y0,All_x1,Play_y1,Hollow_Color,Solid_

```



```

Color);
    puthz(All_x0+Text_xplus,Play_y0+Text_yplus,32,32,BLACK,"模拟
");
    Button_Draw(All_x0,Help_y0,All_x1,Help_y1,Hollow_Color,Solid_
Color);
    puthz(All_x0+Text_xplus,Help_y0+Text_yplus,32,32,BLACK,"帮助
");
    Button_Draw(All_x0,Quit_y0,All_x1,Quit_y1,Hollow_Color,Solid_
Color);
    puthz(All_x0+Text_xplus,Quit_y0+Text_yplus,32,32,BLACK,"退出
");
    Button_Draw(About_x0,About_y0,About_x1,About_y1,Hollow_Color,
Solid_Color);
    InfoImage_Draw(About_x0,About_y0,GRAY);
    Button_Draw(Login_x0,Login_y0,Login_x1,Login_y1,Hollow_Color,
Solid_Color);
    LoginImage_Draw(Login_x0,Login_y0,GRAY);
}

```

```

/*****
*****
* 函数名称      Button_Light_Menu
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void Button_Light_Menu(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(All_x0,Play_y0,All_x1,Play_y1,WHITE);
            Solid_Bar(All_x0,Play_y0,All_x1,Play_y1,GREEN);
            puthz(All_x0+Text_xplus,Play_y0+Text_yplus,32,32,GRAY
,"模拟");
            break;
        }
        case 2:
        {
            Button_Edge(All_x0,Help_y0,All_x1,Help_y1,WHITE);
            Solid_Bar(All_x0,Help_y0,All_x1,Help_y1,YELLOW);

```

```

        puthz(All_x0+Text_xplus,Help_y0+Text_yplus,32,32,GRAY
,"帮助");
        break;
    }
    case 3:
    {
        Button_Edge(All_x0,Quit_y0,All_x1,Quit_y1,WHITE);
        Solid_Bar(All_x0,Quit_y0,All_x1,Quit_y1,RED);
        puthz(All_x0+Text_xplus,Quit_y0+Text_yplus,32,32,GRAY
,"退出");
        break;
    }
    case 4:
    {
        Button_Edge(About_x0,About_y0,About_x1,About_y1,WHITE
);
        InfoImage_Draw(About_x0,About_y0,WHITE);
        break;
    }
    case 5:
    {
        Button_Edge(Login_x0,Login_y0,Login_x1,Login_y1,WHITE
);
        LoginImage_Draw(Login_x0,Login_y0,WHITE);
        break;
    }
}
}

```

```

/*****
*****

```

```

* 函数名称      Button_Darken_Menu
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无

```

```

*****
*****/

```

```

void Button_Darken_Menu(int flag)
{
    switch(flag)
    {
        case 1:
        {

```

```

        Button_Edge(All_x0,Play_y0,All_x1,Play_y1,BLACK);
        Button_Draw(All_x0,Play_y0,All_x1,Play_y1,Hollow_Color,Solid_Color);
        puthz(All_x0+Text_xplus,Play_y0+Text_yplus,32,32,BLACK,"模拟");
        break;
    }
    case 2:
    {
        Button_Edge(All_x0,Help_y0,All_x1,Help_y1,BLACK);
        Button_Draw(All_x0,Help_y0,All_x1,Help_y1,Hollow_Color,Solid_Color);
        puthz(All_x0+Text_xplus,Help_y0+Text_yplus,32,32,BLACK,"帮助");
        break;
    }
    case 3:
    {
        Button_Edge(All_x0,Quit_y0,All_x1,Quit_y1,BLACK);
        Button_Draw(All_x0,Quit_y0,All_x1,Quit_y1,Hollow_Color,Solid_Color);
        puthz(All_x0+Text_xplus,Quit_y0+Text_yplus,32,32,BLACK,"退出");
        break;
    }
    case 4:
    {
        Button_Edge(About_x0,About_y0,About_x1,About_y1,BLACK);
        Button_Draw(About_x0,About_y0,About_x1,About_y1,Hollow_Color,Solid_Color);
        InfoImage_Draw(About_x0,About_y0,GRAY);
        break;
    }
    case 5:
    {
        Button_Edge(Login_x0,Login_y0,Login_x1,Login_y1,BLACK);
        Button_Draw(Login_x0,Login_y0,Login_x1,Login_y1,Hollow_Color,Solid_Color);
        LoginImage_Draw(Login_x0,Login_y0,GRAY);
        break;
    }
}

```

```

    }
}

/*****
*****

* 函数名称      CopyrightText_Draw
* 函数作用      绘制16*16 的版权符号
* 函数输入      无
* 函数输出      无
*****
*****/
void CopyrightText_Draw(int x0,int y0,int color,int bk_color)
{
    Hollow_Circle(x0+10,y0+8,6,color);
    Hollow_Circle(x0+10,y0+8,3,color);
    Solid_Bar(x0+12,y0+6,x0+13,y0+10,bk_color);
}

```

21. mouse.c

```

#include"headfile.h"

/*****
*****

* @author      unknown
* @edictor      ytm
* @date      2022-4-8
* @notice      此版本为SVGA 64k(24 位)显示版本( 针对SVGA 函数
优化)
*              若鼠标器驱动未正确初始化, 将在按键后等待6s 后
退出
*              光标仅适配了浅色输入框
*              重置鼠标为了消除残影, 默认没有隐藏鼠标就取得背
景, 故鼠标也会取入背景, 注意背景鼠标的清除
* @discrpition 2022/2/25: 精简和优化了部分注释
*              2022/3/1: 新增了手型鼠标的图案, 修复了光标图
案的错误, 新增了鼠标形状改变的功能
*              2022/3/2: 修复了鼠标图案绘制的错误, 合并了重
置鼠标背景和重置鼠标形状的函数
*              2022/4/4: 优化了图形显示性能, 完善了注释使之
更易读懂, 修复了一些小问题
*              (注: backgroundChange 与AddFrame
函数未被优化)
*              2022/4/8: 更改了部分鼠标图像的显示方式, 减少
了系统内存的消耗

```

```

*
新增了检测鼠标在范围外的行为的函数
mouse_out_press
*****
*****/

#define xmi 1
#define xma 1023
#define ymi 1
#define yma 767

int MouseX;
int MouseY;
int press;
int shape;
union REGS regs;

/* 鼠标图像数组: 0 是 White, 1 是 Black, 2 是 Dark_Gray, 3 是背景 */
// 箭头鼠标
int ArrowMouse[10][16] = {
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,3,3,3 },
    { 1,0,0,0,0,0,0,0,0,0,0,0,1,3,3,3 },
    { 3,1,0,0,0,0,0,0,0,0,0,0,1,3,3,3 },
    { 3,3,1,0,0,0,0,0,0,0,0,0,1,3,3,3 },
    { 3,3,3,1,0,0,0,0,0,0,0,0,1,1,3,3 },
    { 3,3,3,3,1,0,0,0,0,0,0,0,0,1,1,3 },
    { 3,3,3,3,3,1,0,0,1,1,1,0,0,0,1,3 },
    { 3,3,3,3,3,3,1,0,1,3,3,1,0,0,1,3 },
    { 3,3,3,3,3,3,3,1,1,3,3,3,1,1,3,3 },
    { 3,3,3,3,3,3,3,3,1,3,3,3,3,3,3,3 }
};

/* 十字鼠标图像数组备份 */
/*
// 十字鼠标
int AddMouse[10][16]={
    { 3,3,3,3,3,3,1,1,1,1,1,3,3,3,3,3 },
    { 3,3,3,3,3,3,1,0,0,0,1,3,3,3,3,3 },
    { 3,3,3,3,3,3,1,0,0,0,1,3,3,3,3,3 },
    { 3,3,3,1,1,1,1,0,0,0,1,1,1,1,3,3 },
    { 3,3,3,1,0,0,0,0,0,0,0,0,0,0,1,3 },
    { 3,3,3,1,0,0,0,0,0,0,0,0,0,0,1,3 },
    { 3,3,3,1,1,1,1,0,0,0,1,1,1,1,3,3 },
    { 3,3,3,3,3,3,1,0,0,0,1,3,3,3,3,3 },
    { 3,3,3,3,3,3,1,0,0,0,1,3,3,3,3,3 },

```

```

        { 3,3,3,3,3,3,1,1,1,1,1,3,3,3,3,3 }
};
*/
/* 光标图像数组备份*/
/*
// 光标（适用于浅色输入框）
int InputMouse[10][16]={
    { 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3 },
    { 1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1 },
    { 1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1 },
    { 1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1 },
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
    { 1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1 },
    { 1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1 },
    { 1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1 },
    { 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3 }
};
*/
// 手型鼠标
int HandMouse[10][16]={
    { 3,3,3,3,3,3,3,2,0,0,0,0,3,3,3,3},
    { 3,2,2,2,2,2,2,2,2,2,0,0,0,2,3,3},
    { 2,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3},
    { 3,2,2,2,2,0,0,0,0,0,0,0,0,0,2,3},
    { 3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,2},
    { 3,3,3,2,2,2,2,0,0,0,0,0,0,0,0,2},
    { 3,3,3,3,0,0,0,0,0,0,0,0,0,0,0,2},
    { 3,3,3,3,3,2,2,2,2,0,0,0,0,0,2,3},
    { 3,3,3,3,3,3,0,0,0,0,0,0,0,2,3,3},
    { 3,3,3,3,3,3,3,2,2,2,2,2,2,3,3,3}
};
int bkSave[10][16] = {0};

/*****
*****
* 函数名称      cursor
* 函数作用      绘制鼠标
* 函数输入      x, y 鼠标绘制位置
* 函数注意      shape 为1 绘制十字鼠标
*                shape 为2 绘制光标（适用于浅色输入框）
*                shape 为3 绘制手型鼠标
*                默认绘制箭头鼠标

```

```

* 函数输出      无
*****
*****/
void cursor(int x, int y)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)
0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    //    unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*i 是x 的循环变量; j 是y 的循环变量*/
    int i, j;

    switch(shape)
    {
        case 1:
        {
            /*图像数组输出方式备份*/
            /*
            for (j = 0; j < 16; j++)
            {

                //计算显存地址偏移量和对应的页面号, 做换页操作
                page = ((unsigned long int)(y + j) << 10) + x
;

                new_page = page >> 15;

                SelectPage(new_page);

                for (i = 0; i < 10; i++)
                {
                    if (AddMouse[i][j] == 0)
                        *(video_buffer + page + i) = WHITE;
                    else if (AddMouse[i][j] == 1)
                        *(video_buffer + page + i) = BLACK;
                }
            }
        }
    }
}

```

```

        */
        // 上部
        Line_Plus(x+3,y+3,x+6,y+3,BLACK);
        Line_Plus(x+3,y+4,x+3,y+6,BLACK);
        Line_Plus(x+6,y+4,x+6,y+6,BLACK);
        Line_Plus(x,y+6,x+3,y+6,BLACK);
        Line_Plus(x+6,y+6,x+9,y+6,BLACK);
        Solid_Bar(x+4,y+4,x+5,y+6,WHITE);
        // 中部
        Line_Plus(x,y+7,x,y+9,BLACK);
        Line_Plus(x+9,y+7,x+9,y+9,BLACK);
        Solid_Bar(x+1,y+7,x+8,y+9,WHITE);
        // 下部
        Line_Plus(x+4,y+13,x+5,y+13,BLACK);
        Line_Plus(x+3,y+11,x+3,y+13,BLACK);
        Line_Plus(x+6,y+11,x+6,y+13,BLACK);
        Line_Plus(x,y+10,x+3,y+10,BLACK);
        Line_Plus(x+6,y+10,x+9,y+10,BLACK);
        Solid_Bar(x+4,y+10,x+5,y+12,WHITE);
    }
    break;
case 2:
{
    /* 图像数组输出方式备份*/
    /*
    for (j = 0; j < 16; j++)
    {
        // 计算显存地址偏移量和对应的页面号, 做换页操作
        page = ((unsigned long int)(y + j) << 10) + x

;
        new_page = page >> 15;

        SelectPage(new_page);

        for (i = 0; i < 10; i++)
        {
            if (InputMouse[i][j] == 1)
                *(video_buffer + page + i) = BLACK;
        }
    }
    */
    Line_Plus(x+1,y,x+8,y,BLACK);
    Solid_Bar(x+4,y+1,x+5,y+14,BLACK);

```



```

        Line_Plus(x+1,y+15,x+8,y+15,BLACK);
    }
    break;
case 3:
    {
        for (j = 0; j < 16; j++)
        {
            /* 计算显存地址偏移量和对应的页面号, 做换页操作*/
            page = ((unsigned long int)(y + j) << 10) + x

;

            new_page = page >> 15;

            SelectPage(new_page);

            for (i = 0; i < 10; i++)
            {
                if (HandMouse[i][j] == 0)
                    *(video_buffer + page + i) = WHITE;
                else if (HandMouse[i][j] == 1)
                    *(video_buffer + page + i) = BLACK;
                else if (HandMouse[i][j] == 2)
                    *(video_buffer + page + i) = DARK_GRA
Y;

            }
        }
    }
    break;
default:
    {
        for (j = 0; j < 16; j++)
        {
            for (i = 0; i < 10; i++)
            {
                /* 计算显存地址偏移量和对应的页面号, 做换页操
作*/
                page = ((unsigned long int)(y + j) << 10)
+ x;

                new_page = page >> 15;

                SelectPage(new_page);

                if (ArrowMouse[i][j] == 0)
                    *(video_buffer + page + i) = WHITE;

```

```

        else if (ArrowMouse[i][j] == 1)
            *(video_buffer + page + i) = BLACK;
    }
}
break;
}
}

/*****
*****
* 函数名称      getMousebk
* 函数作用      获取鼠标背景
* 函数输入      x, y 鼠标绘制位置
* 函数输出      无
*****
*****/
void getMousebk(int x, int y)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
    xa000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    /*i 是 x 的循环变量; j 是 y 的循环变量*/
    int i, j;

    for(j=0;j<16;j++)
    {
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y + j) << 10) + x;
        new_page = page >> 15;    /*32k 个点一换页, 除以 32k 的替代算
法*/
        SelectPage(new_page);

        for(i=0;i<10;i++)
        {
            bkSave[i][j] = video_buffer[page + i];

```

```

    }
}

}

/*****
*****
* 函数名称      setMousebk
* 函数作用      设置鼠标背景
* 函数输入      color 背景设置颜色
* 函数输出      无
*****
*****/
void setMousebk(int color)
{
    int i,j;
    for(i=0;i<10;i++)
        for(j=0;j<16;j++)
            bkSave[i][j] = color;
}

/*****
*****
* 函数名称      changeMousebk
* 函数作用      改变鼠标背景
* 函数输入      fromColor 背景被改变颜色, toColor 背景改变颜色
* 函数输出      无
*****
*****/
void changeMousebk(int fromColor,int toColor)
{
    int i,j;
    for(i=0;i<10;i++)
        for(j=0;j<16;j++)
            if(bkSave[i][j] == fromColor)
                bkSave[i][j] == toColor;
}

/*****
*****
* 函数名称      mousehide
* 函数作用      隐藏鼠标
* 函数输入      x, y 鼠标原来位置
* 函数输出      无
*****
*****/

```

```

*****
*****/
void mousehide(int x, int y)
{
    /*显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

    // unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;

    /*i 是 x 的循环变量；j 是 y 的循环变量*/
    int i, j;
    for (j = 0 ;j < 16; j++ )
    {
        /*计算显存地址偏移量和对应的页面号，做换页操作*/
        page = ((unsigned long int)(y + j) << 10) + x;
        new_page = page >> 15;

        SelectPage(new_page);

        for (i = 0 ;i < 10; i++ )
            *(video_buffer + page + i) = bkSave[i][j];
    }
}

/*****
*****
* 函数名称      init
* 函数作用      初始化鼠标驱动
* 函数输入      无
* 函数输出      无
*****
*****/
int init()
{
    int retcode;
    regs.x.ax = 0;
    int86(51, &regs, &regs);

```

```

    retcode = regs.x.ax;
    if (retcode == 0)
        return 0;
    regs.x.ax = 7;
    regs.x.cx = xmi;
    regs.x.dx = xma;
    int86(51, &regs, &regs);
    regs.x.ax = 8;
    regs.x.cx = ymi;
    regs.x.dx = yma;
    int86(51, &regs, &regs);
    return retcode;
}

/*****
*****
* 函数名称      mouseInit
* 函数作用      完全初始化鼠标
* 函数输入      mx, my 鼠标检测位置地址, mbutt 鼠标样式
* 函数输出      无
*****
*****/
void mouseInit(int *mx, int *my, int *mbutt)
{
    int u=init();
    if ( u == 0)
    {
        delay(6000);
        getch();
        exit(1);
    }
    shape=0;
    *mx = 3;
    *my = 460;
    *mbutt = 0;
    getMousebk(*mx, *my);
    cursor(*mx, *my);
}

/*****
*****
* 函数名称      readxy
* 函数作用      读取鼠标位置
*****/

```

```

* 函数输入          mx, my 鼠标检测位置地址, mbutt 鼠标样式
* 函数输出          鼠标状态
*****
*****/
int readxy(int *mx, int *my, int *mbutt)
{
    static int mark = 0; // 按键按松开标志
    int xx0 = *mx, yy0 = *my, buto = *mbutt;
    int xnew, ynew;
    do
    {

        regs.x.ax = 3;
        int86(51, &regs, &regs);

        MouseX = 1.595* regs.x.cx;
        MouseY = 3.825* regs.x.dx;
        *mbutt = regs.x.bx;
        if (mark == 0 && regs.x.bx != 0)
        {
            mark = 1;
            if(regs.x.bx != 0)*mbutt = regs.x.bx;
        }
        else if (regs.x.bx == 0)
        {
            mark = 0;
            *mbutt = 0;
        }
        else *mbutt = 0;

    }while(xnew == xx0&&ynew == yy0&&*mbutt == buto);
    *mx = MouseX;
    *my = MouseY;
    if (*mbutt)
    {
        *mx =MouseX;
        *my =MouseY;
        *mbutt = regs.x.bx;
        return -1;
    }
    else
    {

```

```

        *mx =MouseX;
        *my =MouseY;
        *mbutt = regs.x.bx;
        return 1;
    }
}

/*****
*****

* 函数名称      newxy
* 函数作用      在新的位置处画鼠标
* 函数输入      mx, my 鼠标检测位置地址, mbutt 鼠标样式
* 函数输出      无
*****
*****/
void newxy(int *mx, int *my, int *mbutt)
{
    int ch, xx0 = *mx, yy0 = *my;
    int xm, ym;

    readxy(&xm, &ym, &press);

    if (xm != xx0 || ym != yy0)
    {
        mousehide(xx0, yy0);
        // 恢复原来图像
        getMousebk(xm, ym);
        // 获得背景
        cursor(xm,ym);
        // 绘制鼠标
        *mx = xm;
        *my = ym;
    }
}

/*****
*****

* 函数名称      mouse_press
* 函数作用      检测鼠标在范围内的行为
* 函数输入      x1, y1 鼠标范围左上角坐标, x2, y2 鼠标范围右上角坐标
* 函数输出      如果在框中点击, 则返回1; 在框中未点击, 则返回2; 在
框外右击返回3; 不在框中则返回0
*****
*****/

```

```

*****/
int mouse_press(int x1, int y1, int x2, int y2)
{
    // 在框中点击, 则返回 1
    if(MouseX > x1
    &&MouseX < x2
    &&MouseY > y1
    &&MouseY < y2
    &&press == 1)
    {
        return 1;
    }
    // 在框中未点击, 则返回 2
    else if(MouseX > x1
    &&MouseX < x2
    &&MouseY > y1
    &&MouseY < y2
    &&press == 0)
    {
        return 2;
    }
    // 在框中点击右键, 则返回 3
    else if(MouseX > x1
    &&MouseX < x2
    &&MouseY > y1
    &&MouseY < y2
    &&press == 2)
    {
        return 3;
    }
    else
    {
        return 0;
    }
}

/*****
*****
* 函数名称      mouse_out_press
* 函数作用      检测鼠标在范围外的行为
* 函数输入      x1, y1 鼠标范围左上角坐标, x2, y2 鼠标范围右上角坐标
* 函数输出      如果在框外点击, 则返回 1; 在框外未点击, 则返回 2; 在
框外右击则返回 3; 在框内返回 0

```



```

*****
*****/
int mouse_out_press(int x1, int y1, int x2, int y2)
{
    //在框内
    if(MouseX >= x1
        &&MouseX <= x2
        &&MouseY >= y1
        &&MouseY <= y2)
    {
        return 0;
    }
    else
    {
        //在框中点击, 则返回1
        if(press == 1)
        {
            return 1;
        }
        //在框中未点击, 则返回2
        else if(press == 0)
        {
            return 2;
        }
        //在框中点击右键, 则返回3
        else
        {
            return 3;
        }
    }
}

/*****
*****
* 函数名称      backgroundChange
* 函数作用      为了输入的时候不遮住鼠标
* 函数输入      mx, my 鼠标位置, x1, y1 输入框左上角坐标, x2, y2 输入框右上角坐标
* 函数输出      无
*****
*****/
void backgroundChange(int mx, int my,int x1,int y1,int x2,int y2)
{

```

```

    int i, j;
    int mark = 0;

    for(i=0;i<10;i++)
        for (j = 0;j < 16;j++)
        {
            if (mx + i >= x1&&mx + i <= x2&&my + j >= y1&&my + j
<= y2)
            {
                bkSave[i][j] = Getpixel64k(mx + i, my + j);
                mark = 1;
            }
        }
    if (mark == 1)
    {
        mousehide(mx,my);
        getMousebk(mx, my);
        cursor(mx, my);
    }
}

/*****
*****
* 函数名称      AddFrame
* 函数作用      为鼠标增加边框
* 函数输入      mx, my 鼠标位置, x1, y1 输入框左上角坐标, x2, y2 输
入框右上角坐标, thick 边框厚度, color 边框颜色
* 函数输出      无
*****
*****/
void AddFrame(int mx, int my, int x1, int y1, int x2, int y2,int
thick,int color)
{
    int i, j;
    Solid_Bar(x1, y1, x2, y2, color);
    if (thick == 3)
    {
        for (i = 0;i < 10;i++)
            for (j = 0;j < 16;j++)
            {
                if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 ||
my + j == y2))
                    bkSave[i][j] = Getpixel64k(mx + i, my + j);
            }
        }
    }
}

```

```

        if (my + j <= y2&&my + j >= y1&&mx + i == x1 || m
x + i == x2)
            bkSave[i][j] = Getpixel64k(mx + i, my + j);

            if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 + 1
|| my + j == y2 + 1))
                bkSave[i][j] = Getpixel64k(mx + i, my + j);
            if (my + j <= y2&&my + j >= y1&&(mx + i == x1 + 1
|| mx + i == x2 + 1))
                bkSave[i][j] = Getpixel64k(mx + i, my + j);

            if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 - 1
|| my + j == y2 - 1))
                bkSave[i][j] = Getpixel64k(mx + i, my + j);
            if (my + j <= y2&&my + j >= y1&&(mx + i == x1 - 1
|| mx + i == x2 - 1))
                bkSave[i][j] = Getpixel64k(mx + i, my + j);
        }
    }
    else if (thick == 1)
    {
        for (i = 0;i < 10;i++)
            for (j = 0;j < 16;j++)
            {
                if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 ||
my + j == y2))
                    bkSave[i][j] = Getpixel64k(mx + i, my + j);
                if (my + j <= y2&&my + j >= y1&&(mx + i == x1 ||
mx + i == x2))
                    bkSave[i][j] = Getpixel64k(mx + i, my + j);
            }
    }
}

```

```

/*****
*****

```

```

* 函数名称      resetMouse
* 函数作用      重置鼠标背景
* 函数输入      mx, my 鼠标位置
* 函数注意      用于由于两次newxy 之间的图形绘制导致背景像素未刷新的
情况
*
*              mousehide(mx, my): 该条代码放于绘制之前
*              resetMouse(MouseX,MouseY): 本函数放于绘制之后

```

```

* 函数输出      无
*****
*****/
void resetMouse(int mx,int my)
{
    //mousehide(mx, my);
    /* 该条代码放于绘制之前*/
    //shape=0;
    /* 并不建议使用*/
    getMousebk(mx,my);
    /* 本函数放于绘制之后*/
}

```

22. moveofc.c

```

#include "headfile.h"

/*****
*****
* 函数名称      Child_Movement
* 函数作用      少年动画库
* 函数输入      x,y 绘制坐标, forward 运动方向, move 动作, frame 关
键帧序号, color 衣服颜色
* 函数输出      无
*****
*****/

void Child_Movement(int x,int y,char forward,char move,char frame
,int color)
{
    switch(move)
    {
        case RUN:
            switch(forward)
            {
                case MOVE_RIGHTUP:
                case MOVE_RIGHT:
                case MOVE_RIGHTDOWN:
                    switch(frame)
                    {
                        case 1:
                            Draw_Child(x,y,color);
                            break;
                        case 2:
                            Solid_Circle(x+11,y+11,5,BODYCOLOR);/

```

```

/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Bar(x+18,y+18,x+19,y+21,color);
Solid_Bar(x+19,y+22,x+20,y+25,color);
Solid_Bar(x+20,y+27,x+21,y+28,BODYCOL
OR);
Solid_Bar(x+21,y+29,x+22,y+30,BODYCOL
OR);
Solid_Bar(x+22,y+31,x+23,y+32,BODYCOL
OR);
Solid_Bar(x+4,y+18,x+5,y+20,color);
Solid_Bar(x+3,y+20,x+4,y+22,color);
Solid_Bar(x+2,y+23,x+3,y+25,color);
Solid_Bar(x+2,y+27,x+3,y+29,BODYCOLOR
);
Solid_Bar(x+3,y+30,x+4,y+32,BODYCOLOR
);
Solid_Bar(x+7,y+33,x+10,y+36,color);
Solid_Bar(x+8,y+37,x+11,y+39,color);
Solid_Bar(x+8,y+41,x+11,y+42,BODYCOLO
R);
Solid_Bar(x+7,y+43,x+10,y+45,BODYCOLO
R);
Solid_Bar(x+6,y+46,x+9,y+47,BODYCOLOR
);
Solid_Bar(x+14,y+33,x+17,y+36,color);
Solid_Bar(x+15,y+37,x+18,y+39,color);
Solid_Quadrangle(x+15,y+41,x+18,y+41,
x+14,y+46,x+16,y+47,BODYCOLOR);
break;
case 3:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Quadrangle(x+18,y+19,x+19,y+18,
x+23,y+24,x+24,y+23,color);//R
Solid_Bar(x+25,y+25,x+26,y+26,BODYCOL
OR);
Solid_Bar(x+27,y+26,x+28,y+27,BODYCOL
OR);
Solid_Bar(x+28,y+27,x+29,y+28,BODYCOL

```

```

OR);
Solid_Bar(x+30,y+28,x+31,y+29,BODYCOL
OR);
Solid_Quadrangle(x+4,y+18,x+5,y+19,x+
0,y+26,x+1,y+27,color);
Solid_Quadrangle(x+1,y+29,x+0,y+30,x+
4,y+34,x+3,y+35,BODYCOLOR);//L
Solid_Quadrangle(x+14,y+35,x+16,y+33,
x+19,y+40,x+21,y+38,color);
Solid_Quadrangle(x+21,y+40,x+24,y+41,
x+19,y+46,x+22,y+47,BODYCOLOR);//R
Solid_Quadrangle(x+7,y+33,x+10,y+34,x
+4,y+40,x+8,y+41,color);//L
Solid_Quadrangle(x+6,y+43,x+3,y+42,x-
1,y+46,x+2,y+48,BODYCOLOR);
break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Draw_Child(x,y,color);
break;
case 2:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Bar(x+5,y+18,x+4,y+21,color);
Solid_Bar(x+4,y+22,x+3,y+25,color);
Solid_Bar(x+3,y+27,x+2,y+28,BODYCOLOR
);
Solid_Bar(x+2,y+29,x+1,y+30,BODYCOLOR
);
Solid_Bar(x+1,y+31,x+0,y+32,BODYCOLOR
);
Solid_Bar(x+19,y+18,x+18,y+20,color);
Solid_Bar(x+20,y+20,x+19,y+22,color);
Solid_Bar(x+21,y+23,x+20,y+25,color);
Solid_Bar(x+21,y+27,x+20,y+29,BODYCOL

```

```

OR);
Solid_Bar(x+20,y+30,x+19,y+32,BODYCOL
OR);
Solid_Bar(x+16,y+33,x+13,y+36,color);
Solid_Bar(x+15,y+37,x+12,y+39,color);
Solid_Bar(x+15,y+41,x+12,y+42,BODYCOL
OR);
Solid_Bar(x+16,y+43,x+13,y+45,BODYCOL
OR);
Solid_Bar(x+17,y+46,x+14,y+47,BODYCOL
OR);
Solid_Bar(x+9,y+33,x+6,y+36,color);
Solid_Bar(x+8,y+37,x+5,y+39,color);
Solid_Quadrangle(x+8,y+41,x+5,y+41,x+
9,y+46,x+7,y+47,BODYCOLOR);
break;
case 3:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Quadrangle(x+5,y+19,x+4,y+18,x+
0,y+24,x-1,y+23,color);//R
Solid_Bar(x-2,y+25,x-
3,y+26,BODYCOLOR);
Solid_Bar(x-4,y+26,x-
5,y+27,BODYCOLOR);
Solid_Bar(x-5,y+27,x-
6,y+28,BODYCOLOR);
Solid_Bar(x-7,y+28,x-
8,y+29,BODYCOLOR);
Solid_Quadrangle(x+19,y+18,x+18,y+19,
x+23,y+26,x+22,y+27,color);
Solid_Quadrangle(x+22,y+29,x+23,y+30,
x+19,y+34,x+20,y+35,BODYCOLOR);//L
Solid_Quadrangle(x+9,y+35,x+7,y+33,x+
4,y+40,x+2,y+38,color);
Solid_Quadrangle(x+2,y+40,x-
1,y+41,x+4,y+46,x+1,y+47,BODYCOLOR);//R
Solid_Quadrangle(x+16,y+33,x+13,y+34,
x+19,y+40,x+15,y+41,color);//L
Solid_Quadrangle(x+17,y+43,x+20,y+42,
x+24,y+46,x+21,y+48,BODYCOLOR);

```

```

                                break;
                            }
                        break;
                    case MOVE_UP:
                    case MOVE_DOWN:
                    {
                        switch(frame)
                        {
                            case 1:
                                Draw_Child(x,y,color);
                                break;
                            case 2:
                                Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                                Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
                                Solid_Bar(x+4,y+18,x+5,y+26,color);
                                Solid_Bar(x+4,y+28,x+5,y+33,BODYCOLOR
);
                                Solid_Bar(x+18,y+18,x+19,y+25,color);
                                Solid_Bar(x+18,y+27,x+19,y+30,BODYCOL
OR);
                                Solid_Bar(x+7,y+33,x+10,y+38,color);
                                Solid_Bar(x+7,y+40,x+10,y+45,BODYCOLO
R);
                                Solid_Bar(x+13,y+33,x+16,y+37,color);
                                Solid_Bar(x+13,y+39,x+16,y+42,BODYCOL
OR);
                                break;
                            case 3:
                                Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                                Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
                                Solid_Bar(x+4,y+18,x+5,y+25,color);
                                Solid_Bar(x+4,y+27,x+5,y+31,BODYCOLOR
);
                                Solid_Bar(x+18,y+18,x+19,y+24,color);
                                Line_Plus(x+18,y+26,x+19,y+26,BODYCOL
OR);
                                Solid_Bar(x+7,y+33,x+10,y+38,color);
                                Solid_Bar(x+7,y+40,x+10,y+44,BODYCOLO
R);

```



```

Solid_Bar(x+13,y+33,x+16,y+36,color);
Solid_Bar(x+13,y+38,x+16,y+40,BODYCOL
OR);

break;
case 4:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head

Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body

Solid_Bar(x+19,y+18,x+18,y+26,color);
Solid_Bar(x+19,y+28,x+18,y+33,BODYCOL
OR);

Solid_Bar(x+5,y+18,x+4,y+25,color);
Solid_Bar(x+5,y+27,x+4,y+30,BODYCOLOR
);

Solid_Bar(x+16,y+33,x+13,y+38,color);
Solid_Bar(x+16,y+40,x+13,y+45,BODYCOL
OR);

Solid_Bar(x+10,y+33,x+7,y+37,color);
Solid_Bar(x+10,y+39,x+7,y+42,BODYCOLO
R);

break;
case 5:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head

Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body

Solid_Bar(x+19,y+18,x+18,y+25,color);
Solid_Bar(x+19,y+27,x+18,y+31,BODYCOL
OR);

Solid_Bar(x+5,y+18,x+4,y+24,color);
Line_Plus(x+5,y+26,x+4,y+26,BODYCOLOR
);

Solid_Bar(x+16,y+33,x+13,y+38,color);
Solid_Bar(x+16,y+40,x+13,y+44,BODYCOL
OR);

Solid_Bar(x+10,y+33,x+7,y+36,color);
Solid_Bar(x+10,y+38,x+7,y+40,BODYCOLO
R);

break;
}
}
}

```

```

break;
case WALK:
    switch(forward)
    {
        case MOVE_RIGHTUP:
        case MOVE_RIGHT:
        case MOVE_RIGHTDOWN:
            switch(frame)
            {
                case 1:
                    Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                    Solid_Bar(x+9,y+18,x+14,y+31,color);/
/body
                    Solid_Bar(x+6,y+18,x+7,y+27,color);
                    Solid_Bar(x+6,y+29,x+7,y+35,BODYCOLOR
);
                    Solid_Bar(x+16,y+18,x+17,y+27,color);
                    Solid_Bar(x+16,y+29,x+17,y+35,BODYCOL
OR);
                    Solid_Bar(x+10,y+33,x+13,y+39,color);
                    Solid_Bar(x+10,y+41,x+13,y+47,BODYCOL
OR);
                    break;
                case 2:
                    Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                    Solid_Bar(x+9,y+18,x+14,y+31,color);/
/body
                    Solid_Bar(x+6,y+18,x+7,y+22,color);
                    Solid_Bar(x+5,y+23,x+6,y+27,color);
                    Solid_Bar(x+5,y+29,x+6,y+35,BODYCOLOR
);
                    Solid_Bar(x+16,y+18,x+17,y+22,color);
                    Solid_Bar(x+17,y+23,x+18,y+27,color);
                    Solid_Bar(x+17,y+29,x+18,y+31,BODYCOL
OR);
                    Solid_Bar(x+18,y+32,x+19,y+35,BODYCOL
OR);
                    Putpixel64k(x+9,y+33,color);
                    Putpixel64k(x+10,y+33,color);
                    Putpixel64k(x+13,y+33,color);
                    Putpixel64k(x+14,y+33,color);

```

```

Solid_Bar(x+9,y+34,x+14,y+35,color);
Solid_Bar(x+8,y+36,x+15,y+38,color);
Line_Plus(x+10,y+39,x+13,y+39,color);
Solid_Quadrangle(x+16,y+40,x+13,y+41,
x+19,y+47,x+16,y+47,BODYCOLOR);
Solid_Quadrangle(x+7,y+40,x+10,y+41,x
+6,y+46,x+9,y+47,BODYCOLOR);
break;
case 3:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+9,y+18,x+14,y+31,color);/
/body
Solid_Bar(x+6,y+18,x+7,y+20,color);
Solid_Bar(x+5,y+21,x+6,y+24,color);
Solid_Bar(x+4,y+25,x+5,y+27,color);
Solid_Bar(x+4,y+29,x+5,y+35,BODYCOLOR
);
Solid_Bar(x+16,y+18,x+17,y+20,color);
Solid_Bar(x+17,y+21,x+18,y+24,color);
Solid_Bar(x+18,y+25,x+19,y+27,color);
Solid_Quadrangle(x+20,y+28,x+19,y+29,
x+23,y+33,x+24,y+32,BODYCOLOR);
Solid_Quadrangle(x+9,y+33,x+12,y+34,x
+7,y+38,x+10,y+39,color);
Solid_Quadrangle(x+6,y+40,x+9,y+41,x+
5,y+46,x+8,y+47,BODYCOLOR);
Solid_Quadrangle(x+14,y+33,x+11,y+34,
x+13,y+39,x+16,y+38,color);
Solid_Quadrangle(x+17,y+40,x+14,y+41,
x+21,y+47,x+18,y+47,BODYCOLOR);
break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head

```

```

Solid_Bar(x+9,y+18,x+14,y+31,color);/
/body

Solid_Bar(x+6,y+18,x+7,y+27,color);
Solid_Bar(x+6,y+29,x+7,y+35,BODYCOLOR
);

Solid_Bar(x+16,y+18,x+17,y+27,color);
Solid_Bar(x+16,y+29,x+17,y+35,BODYCOL
OR);

Solid_Bar(x+10,y+33,x+13,y+39,color);
Solid_Bar(x+10,y+41,x+13,y+47,BODYCOL
OR);

break;
case 2:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head

Solid_Bar(x+9,y+18,x+14,y+31,color);/
/body

Solid_Bar(x+17,y+18,x+16,y+22,color);
Solid_Bar(x+18,y+23,x+17,y+27,color);
Solid_Bar(x+18,y+29,x+17,y+35,BODYCOL
OR);

Solid_Bar(x+7,y+18,x+6,y+22,color);
Solid_Bar(x+6,y+23,x+5,y+27,color);
Solid_Bar(x+6,y+29,x+5,y+31,BODYCOLOR
);

Solid_Bar(x+5,y+32,x+4,y+35,BODYCOLOR
);

Putpixel64k(x+9,y+33,color);
Putpixel64k(x+10,y+33,color);
Putpixel64k(x+13,y+33,color);
Putpixel64k(x+14,y+33,color);
Solid_Bar(x+14,y+34,x+9,y+35,color);
Solid_Bar(x+15,y+36,x+8,y+38,color);
Line_Plus(x+13,y+39,x+10,y+39,color);
Solid_Quadrangle(x+7,y+40,x+10,y+41,x
+4,y+47,x+7,y+47,BODYCOLOR);
//Solid_Quadrangle(x+16,y+40,x+13,y+4
1,x+17,y+46,x+14,y+47,BODYCOLOR);
Solid_Bar(x+14,y+41,x+17,y+43,BODYCOL
OR);

Solid_Bar(x+15,y+43,x+18,y+46,BODYCOL
OR);

Putpixel64k(x+16,y+40,BODYCOLOR);

```

```

        Putpixel64k(x+17,y+40,BODYCOLOR);
        Putpixel64k(x+15,y+47,BODYCOLOR);
        Putpixel64k(x+16,y+47,BODYCOLOR);
        break;
    case 3:
        Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
        Solid_Bar(x+9,y+18,x+14,y+31,color);/
/body
        Solid_Bar(x+17,y+18,x+16,y+20,color);
        Solid_Bar(x+18,y+21,x+17,y+24,color);
        Solid_Bar(x+19,y+25,x+18,y+27,color);
        Solid_Bar(x+19,y+29,x+18,y+35,BODYCOL
OR);
        Solid_Bar(x+7,y+18,x+6,y+20,color);
        Solid_Bar(x+6,y+21,x+5,y+24,color);
        Solid_Bar(x+5,y+25,x+4,y+27,color);
        Solid_Quadrangle(x+3,y+28,x+4,y+29,x+
0,y+33,x-1,y+32,BODYCOLOR);
        Solid_Quadrangle(x+14,y+33,x+11,y+34,
x+16,y+38,x+13,y+39,color);
        //Solid_Quadrangle(x+17,y+40,x+14,y+4
1,x+18,y+46,x+15,y+47,BODYCOLOR);
        Solid_Bar(x+15,y+41,x+18,y+43,BODYCOL
OR);
        Solid_Bar(x+16,y+43,x+19,y+46,BODYCOL
OR);
        Putpixel64k(x+17,y+40,BODYCOLOR);
        Putpixel64k(x+18,y+40,BODYCOLOR);
        Putpixel64k(x+17,y+47,BODYCOLOR);
        Putpixel64k(x+18,y+47,BODYCOLOR);
        Solid_Quadrangle(x+9,y+33,x+12,y+34,x
+10,y+39,x+7,y+38,color);
        Solid_Quadrangle(x+6,y+40,x+9,y+41,x+
2,y+47,x+5,y+47,BODYCOLOR);
        break;
    }
    break;
case MOVE_UP:
case MOVE_DOWN:
    switch(frame)
    {
        case 1:

```

```

        Draw_Child(x,y,color);
        break;
    case 2:
        Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
        Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
        Solid_Bar(x+4,y+18,x+5,y+28,color);
        Solid_Bar(x+4,y+30,x+5,y+36,BODYCOLOR
);
        Solid_Bar(x+18,y+18,x+19,y+26,color);
        Solid_Bar(x+18,y+28,x+19,y+33,BODYCOL
OR);
        Solid_Bar(x+16,y+33,x+13,y+40,color);
        Solid_Bar(x+16,y+42,x+13,y+49,BODYCOL
OR);
        Solid_Bar(x+10,y+33,x+7,y+38,color);
        Solid_Bar(x+10,y+40,x+7,y+45,BODYCOLO
R);
        break;
    case 3:
        Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
        Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
        Solid_Bar(x+4,y+18,x+5,y+29,color);
        Solid_Bar(x+4,y+31,x+5,y+37,BODYCOLOR
);
        Solid_Bar(x+18,y+18,x+19,y+25,color);
        Solid_Bar(x+18,y+27,x+19,y+31,BODYCOL
OR);
        Solid_Bar(x+16,y+33,x+13,y+41,color);
        Solid_Bar(x+16,y+43,x+13,y+50,BODYCOL
OR);
        Solid_Bar(x+10,y+33,x+7,y+37,color);
        Solid_Bar(x+10,y+39,x+7,y+43,BODYCOLO
R);
        break;
    case 4:
        Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
        Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body

```

```

Solid_Bar(x+19,y+18,x+18,y+28,color);
Solid_Bar(x+19,y+30,x+18,y+36,BODYCOL
OR);

Solid_Bar(x+5,y+18,x+4,y+26,color);
Solid_Bar(x+5,y+28,x+4,y+33,BODYCOLOR
);

Solid_Bar(x+7,y+33,x+10,y+40,color);
Solid_Bar(x+7,y+42,x+10,y+49,BODYCOLO
R);

Solid_Bar(x+13,y+33,x+16,y+38,color);
Solid_Bar(x+13,y+40,x+16,y+45,BODYCOL
OR);

break;
case 5:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Bar(x+19,y+18,x+18,y+29,color);
Solid_Bar(x+19,y+31,x+18,y+37,BODYCOL
OR);

Solid_Bar(x+5,y+18,x+4,y+25,color);
Solid_Bar(x+5,y+27,x+4,y+31,BODYCOLOR
);

Solid_Bar(x+7,y+33,x+10,y+41,color);
Solid_Bar(x+7,y+43,x+10,y+50,BODYCOLO
R);

Solid_Bar(x+13,y+33,x+16,y+37,color);
Solid_Bar(x+13,y+39,x+16,y+43,BODYCOL
OR);

break;
}
break;
}
break;
case CLIMB:
switch(forward)
{
case MOVE_RIGHT:
case MOVE_RIGHTUP:
case MOVE_RIGHTDOWN:
switch(frame)
{

```

```

                                case 1:
                                    Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                                Solid_Bar(x+7,y+18,x+14,y+31,color);/
/body
                                Solid_Bar(x+9,y+18,x+13,y+31,color);
                                Solid_Bar(x+17,y+41,x+20,y+47,BODYCOL
OR);
                                Solid_Quadrangle(x+12,y+33,x+10,y+35,
x+18,y+39,x+16,y+42,color);
                                Solid_Bar(x+21,y+23,x+22,y+16,BODYCOL
OR);
                                Solid_Quadrangle(x+15,y+19,x+16,y+18,
x+21,y+25,x+22,y+24,color);
                                Line_Plus(x+16,y+15,x+16,y+17,color);
                                Putpixel64k(x+16,y+18,color);
                                Line_Plus(x+17,y+13,x+17,y+18,color);
                                Line_Plus(x+18,y+14,x+18,y+15,color);
                                Line_Plus(x+18,y+9,x+18,y+11,BODYCOLO
R);
                                Line_Plus(x+19,y+6,x+19,y+12,BODYCOLO
R);
                                Line_Plus(x+20,y+7,x+20,y+9,BODYCOLOR
);
                                break;
                                case 2:
                                    Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                                    Solid_Bar(x+7,y+18,x+14,y+31,color);/
/body
                                    Solid_Bar(x+10,y+33,x+13,y+39,color);
                                    Solid_Bar(x+14,y+33,x+17,y+36,color);

                                    Solid_Bar(x+10,y+41,x+13,y+47,BODYCOL
OR);
                                    Solid_Bar(x+18,y+34,x+21,y+40,BODYCOL
OR);
                                    Solid_Quadrangle(x+15,y+18,x+16,y+19,
x+20,y+13,x+21,y+14,color);
                                    Solid_Bar(x+21,y+5,x+22,y+12,BODYCOLO
R);
                                break;
                                }

```



```

        break;
    case MOVE_LEFT:
    case MOVE_LEFTUP:
    case MOVE_LEFTDOWN:
        switch(frame)
        {
            case 1:
                Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                Solid_Bar(x+7,y+18,x+14,y+31,color);/
/body
                Solid_Bar(x+14,y+18,x+10,y+31,color);
                Solid_Bar(x+6,y+41,x+3,y+47,BODYCOLOR
);
                Solid_Quadrangle(x+11,y+33,x+13,y+35,
x+5,y+39,x+7,y+42,color);
                Solid_Bar(x+2,y+23,x+1,y+16,BODYCOLOR
);
                Solid_Quadrangle(x+8,y+19,x+7,y+18,x+
2,y+25,x+1,y+24,color);
                Line_Plus(x+7,y+15,x+7,y+17,color);
                Putpixel64k(x+7,y+18,color);
                Line_Plus(x+6,y+13,x+6,y+18,color);
                Line_Plus(x+5,y+14,x+5,y+15,color);
                Line_Plus(x+5,y+9,x+5,y+11,BODYCOLOR)
;
                Line_Plus(x+4,y+6,x+4,y+12,BODYCOLOR)
;
                Line_Plus(x+3,y+7,x+3,y+9,BODYCOLOR);
                break;
            case 2:
                Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
                Solid_Bar(x+7,y+18,x+14,y+31,color);/
/body
                Solid_Bar(x+13,y+33,x+10,y+39,color);
                Solid_Bar(x+9,y+33,x+6,y+36,color);
                Solid_Bar(x+13,y+41,x+10,y+47,BODYCOL
OR);
                Solid_Bar(x+5,y+34,x+2,y+40,BODYCOLOR
);
                Solid_Quadrangle(x+8,y+18,x+7,y+19,x+
3,y+13,x+2,y+14,color);

```

```

Solid_Bar(x+2,y+5,x+1,y+12,BODYCOLOR)
;
break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Quadrangle(x+3,y+27,x+2,y+28,x-
2,y+22,x-3,y+23,BODYCOLOR);
Line_Plus(x+5,y+19,x+5,y+23,color);
Line_Plus(x+4,y+18,x+4,y+27,color);
Line_Plus(x+3,y+22,x+3,y+26,color);
Solid_Bar(x+20,y+3,x+21,y+9,BODYCOLOR
);
Line_Plus(x+18,y+16,x+18,y+18,color);
Line_Plus(x+19,y+14,x+19,y+19,color);
Line_Plus(x+20,y+11,x+20,y+16,color);
Line_Plus(x+21,y+12,x+21,y+14,color);
Solid_Bar(x+7,y+33,x+10,y+39,color);
Solid_Bar(x+7,y+41,x+10,y+47,BODYCOLO
R);
Solid_Bar(x+13,y+33,x+20,y+36,color);
Solid_Bar(x+17,y+37,x+20,y+43,BODYCOL
OR);
break;
case 2:
Solid_Circle(x+11,y+11,5,BODYCOLOR);/
/head
Solid_Bar(x+7,y+18,x+16,y+31,color);/
/body
Solid_Bar(x-2,y+6,x-
1,y+12,BODYCOLOR);
Solid_Quadrangle(x-1,y+12,x-
2,y+13,x+5,y+18,x+4,y+19,color);
Solid_Bar(x+25,y+6,x+24,y+12,BODYCOLO

```



```

#include "headfile.h"

/*****
*****

* 函数名称      Mid_Movement
* 函数作用      中年动画库
* 函数输入      x,y 绘制坐标, forward 运动方向, move 动作, frame 关
键帧序号, color 衣服颜色
* 函数输出      无
*****
*****/

void Mid_Movement(int x,int y,char forward,char move,char frame,i
nt color)
{
    switch(move)
    {
        case RUN:
            switch(forward)
            {
                case MOVE_RIGHTUP:
                case MOVE_RIGHT:
                case MOVE_RIGHTDOWN:
                    switch(frame)
                    {
                        case 1:
                            Draw_Mid(x,y,color);
                            break;
                        case 2:
                            Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                            Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
                            Solid_Quadrangle(x+19,y+12,x+21,y+11,
x+24,y+20,x+22,y+21,color);//R
                            Solid_Quadrangle(x+23,y+23,x+25,y+22,
x+26,y+29,x+28,y+28,BODYCOLOR);
                            Solid_Quadrangle(x+2,y+11,x+4,y+12,x+
0,y+21,x+2,y+22,color);//L
                            Solid_Quadrangle(x+0,y+25,x+1,y+24,x+
2,y+31,x+4,y+30,BODYCOLOR);
                            Solid_Bar(x+6,y+29,x+10,y+33,color);/
/L

```

```

Solid_Bar(x+7,y+34,x+11,y+38,color);
Solid_Bar(x+5,y+40,x+9,y+42,BODYCOLOR
);
Solid_Bar(x+4,y+43,x+8,y+44,BODYCOLOR
);
Solid_Bar(x+3,y+45,x+7,y+47,BODYCOLOR
);
Solid_Bar(x+13,y+29,x+17,y+31,color);
//R
Solid_Bar(x+14,y+32,x+18,y+35,color);
Solid_Bar(x+15,y+36,x+19,y+38,color);
Solid_Quadrangle(x+15,y+40,x+19,y+40,
x+13,y+46,x+17,y+47,BODYCOLOR);
break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Quadrangle(x+21,y+11,x+28,y+18,
x+26,y+20,x+19,y+14,color);//R
Solid_Quadrangle(x+28,y+22,x+29,y+20,
x+35,y+23,x+34,y+25,BODYCOLOR);
Solid_Quadrangle(x+2,y+11,x+4,y+14,x+
1,y+21,x-2,y+20,color);//L
Solid_Quadrangle(x-
1,y+23,x+1,y+29,x+0,y+30,x-3,y+24,BODYCOLOR);
Solid_Quadrangle(x+6,y+29,x+10,y+32,x
+7,y+38,x+3,y+37,color);//L
Solid_Quadrangle(x+0,y+39,x+3,y+40,x+
0,y+46,x-3,y+43,BODYCOLOR);
Solid_Quadrangle(x+13,y+31,x+19,y+38,
x+22,y+35,x+17,y+29,color);//R
Solid_Quadrangle(x+21,y+39,x+19,y+46,
x+23,y+46,x+25,y+40,BODYCOLOR);
Line_Plus(x+20,y+47,x+22,y+47,BODYCOL
OR);
break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:

```

```

        switch(frame)
        {
            case 1:
                Draw_Mid(x,y,color);
                break;
            case 2:
                Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
                Solid_Quadrangle(x+2,y+11,x-
1,y+20,x+1,y+21,x+4,y+12,color);//R
                Solid_Quadrangle(x-2,y+22,x-5,y+28,x-
3,y+29,x+0,y+23,BODYCOLOR);
                Solid_Quadrangle(x+21,y+11,x+23,y+21,
x+21,y+22,x+19,y+12,color);//L
                Solid_Quadrangle(x+22,y+24,x+24,y+25,
x+21,y+31,x+19,y+30,BODYCOLOR);
                Solid_Bar(x+13,y+29,x+17,y+33,color);
//L
                Solid_Bar(x+12,y+34,x+16,y+38,color);
                Solid_Bar(x+14,y+40,x+18,y+42,BODYCOL
OR);
                Solid_Bar(x+15,y+43,x+19,y+44,BODYCOL
OR);
                Solid_Bar(x+16,y+45,x+20,y+47,BODYCOL
OR);
                Solid_Bar(x+6,y+29,x+10,y+31,color);/
/R
                Solid_Bar(x+5,y+32,x+9,y+35,color);
                Solid_Bar(x+4,y+36,x+8,y+38,color);
                Solid_Quadrangle(x+8,y+40,x+4,y+40,x+
10,y+46,x+6,y+47,BODYCOLOR);
                break;
            case 3:
                Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
                Solid_Quadrangle(x+2,y+11,x+4,y+13,x-
5,y+18,x-3,y+20,color);//R
                Solid_Quadrangle(x-6,y+20,x-5,y+22,x-
11,y+25,x-12,y+23,BODYCOLOR);

```

```

Solid_Quadrangle(x+21,y+11,x+19,y+14,
x+23,y+21,x+26,y+20,color);//L
Solid_Quadrangle(x+25,y+23,x+22,y+29,
x+24,y+30,x+27,y+24,BODYCOLOR);
Solid_Quadrangle(x+17,y+29,x+13,y+31,
x+16,y+38,x+20,y+37,color);//L
Solid_Quadrangle(x+20,y+41,x+22,y+38,
x+28,y+42,x+26,y+45,BODYCOLOR);
Solid_Quadrangle(x+10,y+31,x+4,y+38,x
+1,y+35,x+6,y+29,color);//R
Solid_Quadrangle(x+2,y+39,x+4,y+45,x+
1,y+47,x-2,y+40,BODYCOLOR);
//Line_Plus(x+1,y+47,x+3,y+47,BODYCOL
OR);

break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Draw_Mid(x,y,color);
break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+2,y+11,x+4,y+21,color);
Solid_Bar(x+2,y+23,x+4,y+28,BODYCOLOR
);
Solid_Bar(x+19,y+11,x+21,y+20,color);
Solid_Bar(x+19,y+22,x+21,y+26,BODYCOL
OR);
Solid_Bar(x+6,y+29,x+10,y+37,color);
Solid_Bar(x+6,y+39,x+10,y+46,BODYCOLO
R);
Solid_Bar(x+13,y+29,x+17,y+36,color);
Solid_Bar(x+13,y+38,x+17,y+42,BODYCOL
OR);

break;
case 3:

```

```

Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Bar(x+2,y+11,x+4,y+19,color);
Solid_Bar(x+2,y+22,x+4,y+25,BODYCOLOR
);
Solid_Bar(x+19,y+11,x+21,y+18,color);
Line_Plus(x+19,y+20,x+21,y+20,BODYCOL
OR);
Solid_Bar(x+6,y+29,x+10,y+37,color);
Solid_Bar(x+6,y+39,x+10,y+45,BODYCOLO
R);
Solid_Bar(x+13,y+29,x+17,y+34,color);
Solid_Bar(x+13,y+36,x+17,y+40,BODYCOL
OR);
break;
case 4:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Bar(x+19,y+11,x+21,y+21,color);
Solid_Bar(x+19,y+23,x+21,y+28,BODYCOL
OR);
Solid_Bar(x+2,y+11,x+4,y+20,color);
Solid_Bar(x+2,y+22,x+4,y+26,BODYCOLOR
);
Solid_Bar(x+13,y+29,x+17,y+37,color);
Solid_Bar(x+13,y+39,x+17,y+46,BODYCOL
OR);
Solid_Bar(x+6,y+29,x+10,y+36,color);
Solid_Bar(x+6,y+38,x+10,y+42,BODYCOLO
R);
break;
case 5:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Bar(x+19,y+11,x+21,y+19,color);
Solid_Bar(x+19,y+22,x+21,y+25,BODYCOL
OR);

```



```

Solid_Bar(x+2,y+11,x+4,y+18,color);
Line_Plus(x+2,y+20,x+4,y+20,BODYCOLOR
);
Solid_Bar(x+13,y+29,x+17,y+37,color);
Solid_Bar(x+13,y+39,x+17,y+45,BODYCOL
OR);
Solid_Bar(x+6,y+29,x+10,y+34,color);
Solid_Bar(x+6,y+36,x+10,y+40,BODYCOLO
R);
break;
}
}
break;
case WALK:
switch(forward)
{
case MOVE_RIGHTUP:
case MOVE_RIGHT:
case MOVE_RIGHTDOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
Solid_Bar(x+4,y+11,x+6,y+22,color);
Solid_Bar(x+4,y+24,x+6,y+31,BODYCOLOR
);
Solid_Bar(x+17,y+11,x+19,y+22,color);
Solid_Bar(x+17,y+24,x+19,y+31,BODYCOL
OR);
Solid_Bar(x+9,y+29,x+14,y+38,color);
Solid_Bar(x+9,y+40,x+14,y+47,BODYCOLO
R);
break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
// Solid_Quadrangle(x+4,y+11,x+6,y+12,
x+3,y+21,x+5,y+22,color);

```

```

Solid_Bar(x+3,y+24,x+5,y+31,BODYCOLOR
);

Solid_Bar(x+4,y+11,x+5,y+22,color);
Line_Plus(x+3,y+17,x+3,y+21,color);
Line_Plus(x+6,y+12,x+6,y+16,color);
Solid_Bar(x+18,y+11,x+19,y+22,color);
Line_Plus(x+17,y+12,x+17,y+16,color);
Line_Plus(x+20,y+17,x+20,y+21,color);
Solid_Bar(x+20,y+23,x+21,y+30,BODYCOL
OR);

Line_Plus(x+19,y+24,x+19,y+27,BODYCOL
OR);

Line_Plus(x+22,y+27,x+22,y+29,BODYCOL
OR);

Solid_Bar(x+9,y+30,x+13,y+32,color);
Solid_Bar(x+10,y+32,x+14,y+36,color);
Solid_Bar(x+11,y+36,x+15,y+38,color);
Line_Plus(x+11,y+29,x+13,y+29,color);
Line_Plus(x+11,y+39,x+13,y+39,color);

Solid_Quadrangle(x+12,y+41,x+15,y+47,
x+18,y+46,x+15,y+40 ,BODYCOLOR);//R
Line_Plus(x+10,y+29,x+12,y+29,color);

Solid_Bar(x+10,y+30,x+14,y+31,color);
Solid_Bar(x+9,y+31,x+13,y+34,color);
Solid_Bar(x+8,y+33,x+12,y+36,color);
Line_Plus(x+7,y+37,x+11,y+37,color);
Line_Plus(x+9,y+38,x+11,y+38,color);
Putpixel64k(x+8,y+36,color);
Line_Plus(x+6,y+39,x+8,y+39,BODYCOLOR
);

Solid_Bar(x+6,y+40,x+10,y+43,BODYCOLO
R);

Solid_Bar(x+5,y+43,x+9,y+46,BODYCOLOR
);

Line_Plus(x+7,y+47,x+9,y+47,BODYCOLOR
);

break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);/

```

```

/body
Solid_Bar(x+3,y+13,x+6,y+14,color);
Solid_Bar(x+2,y+15,x+5,y+16,color);
Solid_Bar(x+1,y+17,x+4,y+18,color);
Solid_Bar(x,y+19,x+3,y+20,color);
Line_Plus(x+4,y+12,x+6,y+12,color);
Line_Plus(x+0,y+21,x+2,y+21,color);
Solid_Bar(x+0,y+23,x+2,y+30,BODYCOLOR
);
Line_Plus(x+17,y+12,x+19,y+12,color);

Line_Plus(x+21,y+21,x+23,y+21,color);
Solid_Bar(x+17,y+13,x+20,y+14,color);
Solid_Bar(x+18,y+15,x+21,y+16,color);
Solid_Bar(x+19,y+17,x+22,y+18,color);
Solid_Bar(x+20,y+19,x+23,y+20,color);
Solid_Quadrangle(x+23,y+24,x+25,y+23,
x+27,y+28,x+29,y+27,BODYCOLOR);
Solid_Quadrangle(x+9,y+30,x+13,y+29,x
+13,y+39,x+17,y+38,color);
Solid_Quadrangle(x+14,y+41,x+19,y+47,
x+22,y+46,x+17,y+40,BODYCOLOR);
Solid_Quadrangle(x+10,y+29,x+14,y+30,
x+6,y+37,x+10,y+38,color);
Line_Plus(x+5,y+39,x+7,y+39,BODYCOLOR
);
Solid_Bar(x+5,y+40,x+9,y+43,BODYCOLOR
);
Solid_Bar(x+4,y+43,x+8,y+46,BODYCOLOR
);
Line_Plus(x+6,y+47,x+8,y+47,BODYCOLOR
);
break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

```

```

Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

Solid_Bar(x+4,y+11,x+6,y+22,color);
Solid_Bar(x+4,y+24,x+6,y+31,BODYCOLOR
);

Solid_Bar(x+17,y+11,x+19,y+22,color);
Solid_Bar(x+17,y+24,x+19,y+31,BODYCOL
OR);

Solid_Bar(x+9,y+29,x+14,y+38,color);
Solid_Bar(x+9,y+40,x+14,y+47,BODYCLO
R);

break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body
// Solid_Quadrangle(x+4,y+11,x+6,y+12,
x+3,y+21,x+5,y+22,color);
Solid_Bar(x+20,y+24,x+18,y+31,BODYCOL
OR);

Solid_Bar(x+19,y+11,x+18,y+22,color);
Line_Plus(x+20,y+17,x+20,y+21,color);
Line_Plus(x+17,y+12,x+17,y+16,color);
Solid_Bar(x+5,y+11,x+4,y+22,color);
Line_Plus(x+6,y+12,x+6,y+16,color);
Line_Plus(x+3,y+17,x+3,y+21,color);
Solid_Bar(x+3,y+23,x+2,y+30,BODYCOLOR
);

Line_Plus(x+4,y+24,x+4,y+27,BODYCOLOR
);

Line_Plus(x+1,y+27,x+1,y+29,BODYCOLOR
);

Solid_Bar(x+14,y+30,x+10,y+32,color);
Solid_Bar(x+13,y+32,x+9,y+36,color);
Solid_Bar(x+12,y+36,x+8,y+38,color);
Line_Plus(x+12,y+29,x+10,y+29,color);
Line_Plus(x+12,y+39,x+10,y+39,color);

Solid_Quadrangle(x+11,y+41,x+8,y+47,x
+5,y+46,x+8,y+40,BODYCOLOR);//R
Line_Plus(x+13,y+29,x+11,y+29,color);

```

```

Solid_Bar(x+13,y+30,x+9,y+31,color);
Solid_Bar(x+14,y+31,x+10,y+34,color);
Solid_Bar(x+15,y+33,x+13,y+36,color);
Line_Plus(x+16,y+37,x+12,y+37,color);
Line_Plus(x+14,y+38,x+12,y+38,color);
Putpixel64k(x+15,y+36,color);
Line_Plus(x+17,y+39,x+15,y+39,BODYCOL
OR);

Solid_Bar(x+17,y+40,x+13,y+43,BODYCOL
OR);

Solid_Bar(x+18,y+43,x+14,y+46,BODYCOL
OR);

Line_Plus(x+16,y+47,x+14,y+47,BODYCOL
OR);

break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

Solid_Bar(x+20,y+13,x+17,y+14,color);
Solid_Bar(x+21,y+15,x+18,y+16,color);
Solid_Bar(x+22,y+17,x+19,y+18,color);
Solid_Bar(x+23,y+19,x+20,y+20,color);
Line_Plus(x+19,y+12,x+17,y+12,color);
Line_Plus(x+23,y+21,x+21,y+21,color);
Solid_Bar(x+23,y+23,x+21,y+30,BODYCOL
OR);

Line_Plus(x+6,y+12,x+4,y+12,color);
Line_Plus(x+2,y+21,x+0,y+21,color);
Solid_Bar(x+6,y+13,x+3,y+14,color);
Solid_Bar(x+5,y+15,x+2,y+16,color);
Solid_Bar(x+4,y+17,x+1,y+18,color);
Solid_Bar(x+3,y+19,x+0,y+20,color);
Solid_Quadrangle(x+0,y+24,x-2,y+23,x-
4,y+28,x-6,y+27,BODYCOLOR);
Solid_Quadrangle(x+14,y+30,x+10,y+29,
x+10,y+39,x+6,y+38,color);
Solid_Quadrangle(x+8,y+41,x+3,y+47,x+
0,y+46,x+5,y+40,BODYCOLOR);
Solid_Quadrangle(x+13,y+29,x+9,y+30,x
+17,y+37,x+13,y+38,color);
Line_Plus(x+18,y+39,x+16,y+39,BODYCOL

```

```

OR);
Solid_Bar(x+18,y+40,x+14,y+43,BODYCOL
OR);
Solid_Bar(x+19,y+43,x+15,y+46,BODYCOL
OR);
Line_Plus(x+17,y+47,x+15,y+47,BODYCOL
OR);
break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Draw_Mid(x,y,color);
break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+2,y+11,x+4,y+23,color);
Solid_Bar(x+2,y+25,x+4,y+33,BODYCOLOR
);
Solid_Bar(x+19,y+11,x+21,y+21,color);
Solid_Bar(x+19,y+23,x+21,y+29,BODYCOL
OR);
Solid_Bar(x+17,y+29,x+13,y+39,color);
Solid_Bar(x+17,y+41,x+13,y+49,BODYCOL
OR);
Solid_Bar(x+10,y+29,x+6,y+36,color);
Solid_Bar(x+10,y+38,x+6,y+45,BODYCOLO
R);
break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+2,y+11,x+4,y+23,color);
Solid_Bar(x+2,y+25,x+4,y+34,BODYCOLOR
);

```

```

Solid_Bar(x+19,y+11,x+21,y+20,color);
Solid_Bar(x+19,y+22,x+21,y+27,BODYCOL
OR);

Solid_Bar(x+17,y+29,x+13,y+40,color);
Solid_Bar(x+17,y+42,x+13,y+51,BODYCOL
OR);

Solid_Bar(x+10,y+29,x+6,y+35,color);
Solid_Bar(x+10,y+37,x+6,y+43,BODYCOLO
R);

    break;
case 4:
    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
    head
    Solid_Bar(x+6,y+11,x+17,y+27,color);/
    /body

    Solid_Bar(x+21,y+11,x+19,y+23,color);
    Solid_Bar(x+21,y+25,x+19,y+33,BODYCOL
OR);

    Solid_Bar(x+4,y+11,x+2,y+21,color);
    Solid_Bar(x+4,y+23,x+2,y+29,BODYCOLOR
);

    Solid_Bar(x+6,y+29,x+10,y+39,color);
    Solid_Bar(x+6,y+41,x+10,y+49,BODYCOLO
R);

    Solid_Bar(x+13,y+29,x+17,y+36,color);
    Solid_Bar(x+13,y+38,x+17,y+45,BODYCOL
OR);

    break;
case 5:
    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
    head
    Solid_Bar(x+6,y+11,x+17,y+27,color);/
    /body

    Solid_Bar(x+21,y+11,x+19,y+23,color);
    Solid_Bar(x+21,y+25,x+19,y+34,BODYCOL
OR);

    Solid_Bar(x+4,y+11,x+2,y+20,color);
    Solid_Bar(x+4,y+22,x+2,y+27,BODYCOLOR
);

    Solid_Bar(x+6,y+29,x+10,y+40,color);
    Solid_Bar(x+6,y+42,x+10,y+51,BODYCOLO
R);

    Solid_Bar(x+13,y+29,x+17,y+35,color);

```

```

Solid_Bar(x+13,y+37,x+17,y+43,BODYCOL
OR);

break;

}
break;

}
break;
case CLIMB:
switch(forward)
{
case MOVE_RIGHT:
case MOVE_RIGHTUP:
case MOVE_RIGHTDOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

Solid_Bar(x+10,y+29,x+14,y+38,color);
Solid_Bar(x+10,y+40,x+14,y+47,BODYCOL
OR);

Solid_Bar(x+15,y+29,x+17,y+33,color);
Solid_Bar(x+19,y+30,x+22,y+37,BODYCOL
OR);

Solid_Quadrangle(x+16,y+14,x+18,y+16,
x+21,y+9,x+23,y+11,color);
Solid_Bar(x+22,y+1,x+24,y+8,BODYCOLOR
);

break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

Solid_Quadrangle(x+11,y+32,x+14,y+29,
x+17,y+38,x+20,y+35,color);
Solid_Bar(x+18,y+38,x+21,y+45,BODYCOL
OR);

Solid_Bar(x+22,y+11,x+24,y+17,BODYCOL
OR);

Solid_Quadrangle(x+16,y+14,x+18,y+12,

```



```

x+21,y+19,x+23,y+17,color);
Solid_Bar(x+22,y+18,x+24,y+20,color);
Solid_Bar(x+16,y+11,x+18,y+13,color);
Solid_Bar(x+17,y+9,x+19,y+11,color);
Solid_Bar(x+18,y+7,x+20,y+9,color);
Putpixel64k(x+18,y+6,color);
Putpixel64k(x+19,y+6,color);
Solid_Bar(x+19,y+3,x+21,y+4,BODYCOLOR
);

Putpixel64k(x+21,y+5,BODYCOLOR);
Solid_Bar(x+20,y,x+22,y+3,BODYCOLOR);
Solid_Bar(x+21,y-1,x+23,y,BODYCOLOR);
Putpixel64k(x+21,y-2,BODYCOLOR);
Putpixel64k(x+22,y-2,BODYCOLOR);
break;
}
break;
case MOVE_LEFT:
case MOVE_LEFTUP:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
Solid_Bar(x+13,y+29,x+9,y+38,color);
Solid_Bar(x+13,y+40,x+9,y+47,BODYCOLOR
R);
Solid_Bar(x+8,y+29,x+6,y+33,color);
Solid_Bar(x+4,y+30,x+1,y+37,BODYCOLOR
);
Solid_Quadrangle(x+7,y+14,x+5,y+16,x+
2,y+9,x+0,y+11,color);
Solid_Bar(x+1,y+1,x-1,y+8,BODYCOLOR);
break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
Solid_Quadrangle(x+12,y+32,x+9,y+29,x

```

```

+6,y+38,x+3,y+35,color);
Solid_Bar(x+5,y+38,x+2,y+45,BODYCOLOR
);
Solid_Bar(x+1,y+11,x-
1,y+17,BODYCOLOR);
Solid_Quadrangle(x+7,y+14,x+5,y+12,x+
2,y+19,x+0,y+17,color);
Solid_Bar(x+1,y+18,x-1,y+20,color);
Solid_Bar(x+7,y+11,x+5,y+13,color);
Solid_Bar(x+6,y+9,x+4,y+11,color);
Solid_Bar(x+5,y+7,x+3,y+9,color);
Putpixel64k(x+5,y+6,color);
Putpixel64k(x+4,y+6,color);
Solid_Bar(x+4,y+3,x+2,y+4,BODYCOLOR);
Putpixel64k(x+2,y+5,BODYCOLOR);
Solid_Bar(x+3,y,x+1,y+3,BODYCOLOR);
Solid_Bar(x+2,y-1,x+0,y,BODYCOLOR);
Putpixel64k(x+2,y-2,BODYCOLOR);
Putpixel64k(x+1,y-2,BODYCOLOR);
break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+2,y+11,x+3,y+21,color);
Line_Plus(x+4,y+12,x+4,y+16,color);
Line_Plus(x+1,y+16,x+1,y+20,color);
Solid_Quadrangle(x+3,y+20,x+1,y+22,x-
4,y+17,x-2,y+15,BODYCOLOR);
Line_Plus(x+19,y+9,x+19,y+11,color);
Line_Plus(x+20,y+5,x+20,y+12,color);
Line_Plus(x+21,y+2,x+21,y+12,color);
Line_Plus(x+22,y+2,x+22,y+9,color);
Line_Plus(x+23,y+3,x+23,y+5,color);
Solid_Bar(x+21,y-
7,x+23,y+0,BODYCOLOR);

```

```

Solid_Bar(x+6,y+29,x+10,y+38,color);
Solid_Bar(x+6,y+40,x+10,y+47,BODYCOLOR);
R);

Solid_Bar(x+13,y+29,x+22,y+33,color);
Solid_Bar(x+18,y+35,x+22,y+42,BODYCOLOR);
OR);

break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Bar(x-4,y-1,x-2,y+6,BODYCOLOR);
Solid_Quadrangle(x-2,y+5,x-
4,y+7,x+2,y+13,x+4,y+11,color);
Solid_Bar(x+25,y-
1,x+27,y+6,BODYCOLOR);
Solid_Quadrangle(x+25,y+5,x+27,y+7,x+
21,y+13,x+19,y+11,color);
Solid_Bar(x+1,y+37,x+5,y+44,BODYCOLOR);
);
Solid_Quadrangle(x+1,y+35,x+4,y+38,x+
10,y+32,x+7,y+29,color);
Line_Plus(x+8,y+29,x+10,y+31,color);
Putpixel64k(x+9,y+29,color);
Putpixel64k(x+10,y+30,color);
Solid_Bar(x+22,y+37,x+18,y+44,BODYCOLOR);
OR);
Solid_Quadrangle(x+22,y+35,x+19,y+38,
x+13,y+32,x+16,y+29,color);
Line_Plus(x+13,y+31,x+15,y+29,color);
Putpixel64k(x+13,y+30,color);
Putpixel64k(x+14,y+29,color);
break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Bar(x+21,y+11,x+20,y+21,color);
Line_Plus(x+19,y+12,x+19,y+16,color);
Line_Plus(x+22,y+16,x+22,y+20,color);
Solid_Quadrangle(x+20,y+20,x+22,y+22,

```

```

x+27,y+17,x+25,y+15,BODYCOLOR);
                                Line_Plus(x+4,y+9,x+4,y+11,color);
                                Line_Plus(x+3,y+5,x+3,y+12,color);
                                Line_Plus(x+2,y+2,x+2,y+12,color);
                                Line_Plus(x+1,y+2,x+1,y+9,color);
                                Line_Plus(x+0,y+3,x+0,y+5,color);
                                Solid_Bar(x+2,y-7,x+0,y+0,BODYCOLOR);
                                Solid_Bar(x+17,y+29,x+13,y+38,color);
                                Solid_Bar(x+17,y+40,x+13,y+47,BODYCOL
OR);
                                Solid_Bar(x+10,y+29,x+1,y+33,color);
                                Solid_Bar(x+5,y+35,x+1,y+42,BODYCOLOR
);
                                break;
                                }
                                }
                                }
}

```

24. moveofo.c

```

#include "headfile.h"

/*****
*****
* 函数名称      Old_Movement
* 函数作用      老年动画库
* 函数输入      x,y 绘制坐标, forward 运动方向, move 动作, frame 关
键帧序号, color 衣服颜色
* 函数输出      无
*****
*****/

void Old_Movement(int x,int y,char forward,char move,char frame,i
nt color)
{
    switch(move)
    {
        case RUN:
            switch(forward)
            {
                case MOVE_RIGHTUP:
                case MOVE_RIGHT:
                case MOVE_RIGHTDOWN:

```

```

switch(frame)
{
    case 1:
        Draw_Old(x,y,color);
        break;
    case 2:
        Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
        Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
        Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
        Solid_Quadrangle(x+19,y+12,x+21,y+11,
x+24,y+20,x+22,y+21,color);//R
        Solid_Quadrangle(x+23,y+23,x+25,y+22,
x+26,y+29,x+28,y+28,BODYCOLOR);
        Solid_Quadrangle(x+2,y+11,x+4,y+12,x+
0,y+21,x+2,y+22,color);//L
        Solid_Quadrangle(x+0,y+25,x+1,y+24,x+
2,y+31,x+4,y+30,BODYCOLOR);
        Solid_Bar(x+6,y+29,x+10,y+33,color);/
/L
        Solid_Bar(x+7,y+34,x+11,y+38,color);
        Solid_Bar(x+5,y+40,x+9,y+42,BODYCOLOR
);
        Solid_Bar(x+4,y+43,x+8,y+44,BODYCOLOR
);
        Solid_Bar(x+3,y+45,x+7,y+47,BODYCOLOR
);
        Solid_Bar(x+13,y+29,x+17,y+31,color);
//R
        Solid_Bar(x+14,y+32,x+18,y+35,color);
        Solid_Bar(x+15,y+36,x+19,y+38,color);
        Solid_Quadrangle(x+15,y+40,x+19,y+40,
x+13,y+46,x+17,y+47,BODYCOLOR);
        break;
    case 3:
        Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
        Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
        Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

```

```

Solid_Quadrangle(x+21,y+11,x+28,y+18,
x+26,y+20,x+19,y+14,color);//R
Solid_Quadrangle(x+28,y+22,x+29,y+20,
x+35,y+23,x+34,y+25,BODYCOLOR);
Solid_Quadrangle(x+2,y+11,x+4,y+14,x+
1,y+21,x-2,y+20,color);//L
Solid_Quadrangle(x-
1,y+23,x+1,y+29,x+0,y+30,x-3,y+24,BODYCOLOR);
Solid_Quadrangle(x+6,y+29,x+10,y+32,x
+7,y+38,x+3,y+37,color);//L
Solid_Quadrangle(x+0,y+39,x+3,y+40,x+
0,y+46,x-3,y+43,BODYCOLOR);
Solid_Quadrangle(x+13,y+31,x+19,y+38,
x+22,y+35,x+17,y+29,color);//R
Solid_Quadrangle(x+21,y+39,x+19,y+46,
x+23,y+46,x+25,y+40,BODYCOLOR);
Line_Plus(x+20,y+47,x+22,y+47,BODYCOL
OR);

break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Draw_Old(x,y,color);
break;
case 2:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Quadrangle(x+2,y+11,x-
1,y+20,x+1,y+21,x+4,y+12,color);//R
Solid_Quadrangle(x-2,y+22,x-5,y+28,x-
3,y+29,x+0,y+23,BODYCOLOR);
Solid_Quadrangle(x+21,y+11,x+23,y+21,
x+21,y+22,x+19,y+12,color);//L
Solid_Quadrangle(x+22,y+24,x+24,y+25,
```

```

x+21,y+31,x+19,y+30,BODYCOLOR);
Solid_Bar(x+13,y+29,x+17,y+33,color);
//L
Solid_Bar(x+12,y+34,x+16,y+38,color);
Solid_Bar(x+14,y+40,x+18,y+42,BODYCOL
OR);
Solid_Bar(x+15,y+43,x+19,y+44,BODYCOL
OR);
Solid_Bar(x+16,y+45,x+20,y+47,BODYCOL
OR);
Solid_Bar(x+6,y+29,x+10,y+31,color);/
/R
Solid_Bar(x+5,y+32,x+9,y+35,color);
Solid_Bar(x+4,y+36,x+8,y+38,color);
Solid_Quadrangle(x+8,y+40,x+4,y+40,x+
10,y+46,x+6,y+47,BODYCOLOR);
break;
case 3:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Quadrangle(x+2,y+11,x+4,y+13,x-
5,y+18,x-3,y+20,color);//R
Solid_Quadrangle(x-6,y+20,x-5,y+22,x-
11,y+25,x-12,y+23,BODYCOLOR);
Solid_Quadrangle(x+21,y+11,x+19,y+14,
x+23,y+21,x+26,y+20,color);//L
Solid_Quadrangle(x+25,y+23,x+22,y+29,
x+24,y+30,x+27,y+24,BODYCOLOR);
Solid_Quadrangle(x+17,y+29,x+13,y+31,
x+16,y+38,x+20,y+37,color);//L
Solid_Quadrangle(x+20,y+41,x+22,y+38,
x+28,y+42,x+26,y+45,BODYCOLOR);
Solid_Quadrangle(x+10,y+31,x+4,y+38,x
+1,y+35,x+6,y+29,color);//R
Solid_Quadrangle(x+2,y+39,x+4,y+45,x+
1,y+47,x-2,y+40,BODYCOLOR);
//Line_Plus(x+1,y+47,x+3,y+47,BODYCOL
OR);
break;

```

```

    }
    break;
    case MOVE_UP:
    case MOVE_DOWN:
    {
        switch(frame)
        {
            case 1:
                Draw_Old(x,y,color);
                break;
            case 2:
                Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
                Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
                Solid_Bar(x+6,y+11,x+17,y+27,color);//
body
                Solid_Bar(x+2,y+11,x+4,y+21,color);
                Solid_Bar(x+2,y+23,x+4,y+28,BODYCOLOR
);
                Solid_Bar(x+19,y+11,x+21,y+20,color);
                Solid_Bar(x+19,y+22,x+21,y+26,BODYCOL
OR);
                Solid_Bar(x+6,y+29,x+10,y+37,color);
                Solid_Bar(x+6,y+39,x+10,y+46,BODYCOLO
R);
                Solid_Bar(x+13,y+29,x+17,y+36,color);
                Solid_Bar(x+13,y+38,x+17,y+42,BODYCOL
OR);
                break;
            case 3:
                Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
                Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
                Solid_Bar(x+6,y+11,x+17,y+27,color);//
body
                Solid_Bar(x+2,y+11,x+4,y+19,color);
                Solid_Bar(x+2,y+22,x+4,y+25,BODYCOLOR
);
                Solid_Bar(x+19,y+11,x+21,y+18,color);
                Line_Plus(x+19,y+20,x+21,y+20,BODYCOL
OR);

```



```

Solid_Bar(x+6,y+29,x+10,y+37,color);
Solid_Bar(x+6,y+39,x+10,y+45,BODYCOLO
R);

Solid_Bar(x+13,y+29,x+17,y+34,color);
Solid_Bar(x+13,y+36,x+17,y+40,BODYCOL
OR);

break;
case 4:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head

Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair

Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body

Solid_Bar(x+19,y+11,x+21,y+21,color);
Solid_Bar(x+19,y+23,x+21,y+28,BODYCOL
OR);

Solid_Bar(x+2,y+11,x+4,y+20,color);
Solid_Bar(x+2,y+22,x+4,y+26,BODYCOLOR
);

Solid_Bar(x+13,y+29,x+17,y+37,color);
Solid_Bar(x+13,y+39,x+17,y+46,BODYCOL
OR);

Solid_Bar(x+6,y+29,x+10,y+36,color);
Solid_Bar(x+6,y+38,x+10,y+42,BODYCOLO
R);

break;
case 5:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head

Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair

Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body

Solid_Bar(x+19,y+11,x+21,y+19,color);
Solid_Bar(x+19,y+22,x+21,y+25,BODYCOL
OR);

Solid_Bar(x+2,y+11,x+4,y+18,color);
Line_Plus(x+2,y+20,x+4,y+20,BODYCOLOR
);

Solid_Bar(x+13,y+29,x+17,y+37,color);
Solid_Bar(x+13,y+39,x+17,y+45,BODYCOL
OR);

```

```

Solid_Bar(x+6,y+29,x+10,y+34,color);
Solid_Bar(x+6,y+36,x+10,y+40,BODYCOLOR);
R);

break;
}
}
}
break;
case WALK:
switch(forward)
{
case MOVE_RIGHTUP:
case MOVE_RIGHT:
case MOVE_RIGHTDOWN:
switch(frame)
{
case 1:
Solid_HalfSector_down(x+11,y+4,5,BODYCOLOR); //head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+8,y+11,x+15,y+27,color); //body
Solid_Bar(x+4,y+11,x+6,y+22,color);
Solid_Bar(x+4,y+24,x+6,y+31,BODYCOLOR);
Solid_Bar(x+17,y+11,x+19,y+22,color);
Solid_Bar(x+17,y+24,x+19,y+31,BODYCOLOR);
Solid_Bar(x+9,y+29,x+14,y+38,color);
Solid_Bar(x+9,y+40,x+14,y+47,BODYCOLOR);
R);
break;
case 2:
Solid_HalfSector_down(x+11,y+4,5,BODYCOLOR); //head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+8,y+11,x+15,y+27,color); //body
// Solid_Quadrangle(x+4,y+11,x+6,y+12,
x+3,y+21,x+5,y+22,color);
Solid_Bar(x+3,y+24,x+5,y+31,BODYCOLOR);

```

```

);

Solid_Bar(x+4,y+11,x+5,y+22,color);
Line_Plus(x+3,y+17,x+3,y+21,color);
Line_Plus(x+6,y+12,x+6,y+16,color);
Solid_Bar(x+18,y+11,x+19,y+22,color);
Line_Plus(x+17,y+12,x+17,y+16,color);
Line_Plus(x+20,y+17,x+20,y+21,color);
Solid_Bar(x+20,y+23,x+21,y+30,BODYCOL
OR);

Line_Plus(x+19,y+24,x+19,y+27,BODYCOL
OR);

Line_Plus(x+22,y+27,x+22,y+29,BODYCOL
OR);

Solid_Bar(x+9,y+30,x+13,y+32,color);
Solid_Bar(x+10,y+32,x+14,y+36,color);
Solid_Bar(x+11,y+36,x+15,y+38,color);
Line_Plus(x+11,y+29,x+13,y+29,color);
Line_Plus(x+11,y+39,x+13,y+39,color);

Solid_Quadrangle(x+12,y+41,x+15,y+47,
x+18,y+46,x+15,y+40,BODYCOLOR);//R
Line_Plus(x+10,y+29,x+12,y+29,color);

Solid_Bar(x+10,y+30,x+14,y+31,color);
Solid_Bar(x+9,y+31,x+13,y+34,color);
Solid_Bar(x+8,y+33,x+12,y+36,color);
Line_Plus(x+7,y+37,x+11,y+37,color);
Line_Plus(x+9,y+38,x+11,y+38,color);
Putpixel64k(x+8,y+36,color);
Line_Plus(x+6,y+39,x+8,y+39,BODYCOLOR
);

Solid_Bar(x+6,y+40,x+10,y+43,BODYCOLO
R);

Solid_Bar(x+5,y+43,x+9,y+46,BODYCOLOR
);

Line_Plus(x+7,y+47,x+9,y+47,BODYCOLOR
);

break;
case 3:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head

Solid_HalfSector_up(x+11,y+4,5,GRAY);

//hair

```

```

Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

Solid_Bar(x+3,y+13,x+6,y+14,color);
Solid_Bar(x+2,y+15,x+5,y+16,color);
Solid_Bar(x+1,y+17,x+4,y+18,color);
Solid_Bar(x,y+19,x+3,y+20,color);
Line_Plus(x+4,y+12,x+6,y+12,color);
Line_Plus(x+0,y+21,x+2,y+21,color);
Solid_Bar(x+0,y+23,x+2,y+30,BODYCOLOR
);

Line_Plus(x+17,y+12,x+19,y+12,color);

Line_Plus(x+21,y+21,x+23,y+21,color);
Solid_Bar(x+17,y+13,x+20,y+14,color);
Solid_Bar(x+18,y+15,x+21,y+16,color);
Solid_Bar(x+19,y+17,x+22,y+18,color);
Solid_Bar(x+20,y+19,x+23,y+20,color);
Solid_Quadrangle(x+23,y+24,x+25,y+23,
x+27,y+28,x+29,y+27,BODYCOLOR);
Solid_Quadrangle(x+9,y+30,x+13,y+29,x
+13,y+39,x+17,y+38,color);
Solid_Quadrangle(x+14,y+41,x+19,y+47,
x+22,y+46,x+17,y+40,BODYCOLOR);
Solid_Quadrangle(x+10,y+29,x+14,y+30,
x+6,y+37,x+10,y+38,color);
Line_Plus(x+5,y+39,x+7,y+39,BODYCOLOR
);

Solid_Bar(x+5,y+40,x+9,y+43,BODYCOLOR
);

Solid_Bar(x+4,y+43,x+8,y+46,BODYCOLOR
);

Line_Plus(x+6,y+47,x+8,y+47,BODYCOLOR
);

break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Solid_HalfSector_down(x+11,y+4,5,BODY

```

```

COLOR); //head
//hair
/body
);
OR);
R);
break;
case 2:
COLOR); //head
//hair
/body
// Solid_Quadrangle(x+4,y+11,x+6,y+12,
x+3,y+21,x+5,y+22,color);
OR);
);
);
);
Solid_Bar(x+14,y+30,x+10,y+32,color);
Solid_Bar(x+13,y+32,x+9,y+36,color);
Solid_Bar(x+12,y+36,x+8,y+38,color);
Line_Plus(x+12,y+29,x+10,y+29,color);
Line_Plus(x+12,y+39,x+10,y+39,color);

```

```

Solid_Quadrangle(x+11,y+41,x+8,y+47,x
+5,y+46,x+8,y+40,BODYCOLOR);//R
Line_Plus(x+13,y+29,x+11,y+29,color);

Solid_Bar(x+13,y+30,x+9,y+31,color);
Solid_Bar(x+14,y+31,x+10,y+34,color);
Solid_Bar(x+15,y+33,x+13,y+36,color);
Line_Plus(x+16,y+37,x+12,y+37,color);
Line_Plus(x+14,y+38,x+12,y+38,color);
Putpixel64k(x+15,y+36,color);
Line_Plus(x+17,y+39,x+15,y+39,BODYCOL
OR);

Solid_Bar(x+17,y+40,x+13,y+43,BODYCOL
OR);

Solid_Bar(x+18,y+43,x+14,y+46,BODYCOL
OR);

Line_Plus(x+16,y+47,x+14,y+47,BODYCOL
OR);

break;
case 3:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head

Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair

Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body

Solid_Bar(x+20,y+13,x+17,y+14,color);
Solid_Bar(x+21,y+15,x+18,y+16,color);
Solid_Bar(x+22,y+17,x+19,y+18,color);
Solid_Bar(x+23,y+19,x+20,y+20,color);
Line_Plus(x+19,y+12,x+17,y+12,color);
Line_Plus(x+23,y+21,x+21,y+21,color);
Solid_Bar(x+23,y+23,x+21,y+30,BODYCOL
OR);

Line_Plus(x+6,y+12,x+4,y+12,color);
Line_Plus(x+2,y+21,x+0,y+21,color);
Solid_Bar(x+6,y+13,x+3,y+14,color);
Solid_Bar(x+5,y+15,x+2,y+16,color);
Solid_Bar(x+4,y+17,x+1,y+18,color);
Solid_Bar(x+3,y+19,x+0,y+20,color);
Solid_Quadrangle(x+0,y+24,x-2,y+23,x-
4,y+28,x-6,y+27,BODYCOLOR);

```

```

Solid_Quadrangle(x+14,y+30,x+10,y+29,
x+10,y+39,x+6,y+38,color);
Solid_Quadrangle(x+8,y+41,x+3,y+47,x+
0,y+46,x+5,y+40,BODYCOLOR);
Solid_Quadrangle(x+13,y+29,x+9,y+30,x
+17,y+37,x+13,y+38,color);
Line_Plus(x+18,y+39,x+16,y+39,BODYCOL
OR);
Solid_Bar(x+18,y+40,x+14,y+43,BODYCOL
OR);
Solid_Bar(x+19,y+43,x+15,y+46,BODYCOL
OR);
Line_Plus(x+17,y+47,x+15,y+47,BODYCOL
OR);

break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Draw_Old(x,y,color);
break;
case 2:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+2,y+11,x+4,y+23,color);
Solid_Bar(x+2,y+25,x+4,y+33,BODYCOLOR
);
Solid_Bar(x+19,y+11,x+21,y+21,color);
Solid_Bar(x+19,y+23,x+21,y+29,BODYCOL
OR);
Solid_Bar(x+17,y+29,x+13,y+39,color);
Solid_Bar(x+17,y+41,x+13,y+49,BODYCOL
OR);
Solid_Bar(x+10,y+29,x+6,y+36,color);
Solid_Bar(x+10,y+38,x+6,y+45,BODYCOLO
R);

```

```

                                break;
case 3:
    Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
                                Solid_HalfSector_up(x+11,y+4,5,GRAY);
                                Solid_Bar(x+6,y+11,x+17,y+27,color);//
                                Solid_Bar(x+2,y+11,x+4,y+23,color);
                                Solid_Bar(x+2,y+25,x+4,y+34,BODYCOLOR
);
                                Solid_Bar(x+19,y+11,x+21,y+20,color);
                                Solid_Bar(x+19,y+22,x+21,y+27,BODYCOL
OR);
                                Solid_Bar(x+17,y+29,x+13,y+40,color);
                                Solid_Bar(x+17,y+42,x+13,y+51,BODYCOL
OR);
                                Solid_Bar(x+10,y+29,x+6,y+35,color);
                                Solid_Bar(x+10,y+37,x+6,y+43,BODYCOLO
R);
                                break;
case 4:
    Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
                                Solid_HalfSector_up(x+11,y+4,5,GRAY);
                                Solid_Bar(x+6,y+11,x+17,y+27,color);//
                                Solid_Bar(x+21,y+11,x+19,y+23,color);
                                Solid_Bar(x+21,y+25,x+19,y+33,BODYCOL
OR);
                                Solid_Bar(x+4,y+11,x+2,y+21,color);
                                Solid_Bar(x+4,y+23,x+2,y+29,BODYCOLOR
);
                                Solid_Bar(x+6,y+29,x+10,y+39,color);
                                Solid_Bar(x+6,y+41,x+10,y+49,BODYCOLO
R);
                                Solid_Bar(x+13,y+29,x+17,y+36,color);
                                Solid_Bar(x+13,y+38,x+17,y+45,BODYCOL
OR);
                                break;
case 5:
    Solid_HalfSector_down(x+11,y+4,5,BODY

```



```

COLOR);//head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+21,y+11,x+19,y+23,color);
Solid_Bar(x+21,y+25,x+19,y+34,BODYCOL
OR);
Solid_Bar(x+4,y+11,x+2,y+20,color);
Solid_Bar(x+4,y+22,x+2,y+27,BODYCOLOR
);
Solid_Bar(x+6,y+29,x+10,y+40,color);
Solid_Bar(x+6,y+42,x+10,y+51,BODYCOLO
R);
Solid_Bar(x+13,y+29,x+17,y+35,color);
Solid_Bar(x+13,y+37,x+17,y+43,BODYCOL
OR);
break;
    }
    break;
}
break;
case CLIMB:
    switch(forward)
    {
        case MOVE_RIGHT:
        case MOVE_RIGHTUP:
        case MOVE_RIGHTDOWN:
            switch(frame)
            {
                case 1:
                    Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
                    Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
                    Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
                    Solid_Bar(x+10,y+29,x+14,y+38,color);
                    Solid_Bar(x+10,y+40,x+14,y+47,BODYCOL
OR);
                    Solid_Bar(x+15,y+29,x+17,y+33,color);
                    Solid_Bar(x+19,y+30,x+22,y+37,BODYCOL
OR);

```

```

Solid_Quadrangle(x+16,y+14,x+18,y+16,
x+21,y+9,x+23,y+11,color);
Solid_Bar(x+22,y+1,x+24,y+8,BODYCOLOR
);
break;
case 2:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
Solid_Quadrangle(x+11,y+32,x+14,y+29,
x+17,y+38,x+20,y+35,color);
Solid_Bar(x+18,y+38,x+21,y+45,BODYCOL
OR);
Solid_Bar(x+22,y+11,x+24,y+17,BODYCOL
OR);
Solid_Quadrangle(x+16,y+14,x+18,y+12,
x+21,y+19,x+23,y+17,color);
Solid_Bar(x+22,y+18,x+24,y+20,color);
Solid_Bar(x+16,y+11,x+18,y+13,color);
Solid_Bar(x+17,y+9,x+19,y+11,color);
Solid_Bar(x+18,y+7,x+20,y+9,color);
Putpixel64k(x+18,y+6,color);
Putpixel64k(x+19,y+6,color);
Solid_Bar(x+19,y+3,x+21,y+4,BODYCOLOR
);
Putpixel64k(x+21,y+5,BODYCOLOR);
Solid_Bar(x+20,y,x+22,y+3,BODYCOLOR);
Solid_Bar(x+21,y-1,x+23,y,BODYCOLOR);
Putpixel64k(x+21,y-2,BODYCOLOR);
Putpixel64k(x+22,y-2,BODYCOLOR);
break;
}
break;
case MOVE_LEFT:
case MOVE_LEFTUP:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Solid_HalfSector_down(x+11,y+4,5,BODY

```

```

COLOR);//head
//hair
/body
R);
);
2,y+9,x+0,y+11,color);
Solid_HalfSector_up(x+11,y+4,5,GRAY);
Solid_Bar(x+8,y+11,x+15,y+27,color);/
Solid_Bar(x+13,y+29,x+9,y+38,color);
Solid_Bar(x+13,y+40,x+9,y+47,BODYCOLOR
Solid_Bar(x+8,y+29,x+6,y+33,color);
Solid_Bar(x+4,y+30,x+1,y+37,BODYCOLOR
);
Solid_Quadrangle(x+7,y+14,x+5,y+16,x+
2,y+9,x+0,y+11,color);
Solid_Bar(x+1,y+1,x-1,y+8,BODYCOLOR);
break;
case 2:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
//hair
/body
+6,y+38,x+3,y+35,color);
Solid_Bar(x+5,y+38,x+2,y+45,BODYCOLOR
);
Solid_Bar(x+1,y+11,x-
1,y+17,BODYCOLOR);
Solid_Quadrangle(x+7,y+14,x+5,y+12,x+
2,y+19,x+0,y+17,color);
Solid_Bar(x+1,y+18,x-1,y+20,color);
Solid_Bar(x+7,y+11,x+5,y+13,color);
Solid_Bar(x+6,y+9,x+4,y+11,color);
Solid_Bar(x+5,y+7,x+3,y+9,color);
Putpixel64k(x+5,y+6,color);
Putpixel64k(x+4,y+6,color);
Solid_Bar(x+4,y+3,x+2,y+4,BODYCOLOR);
Putpixel64k(x+2,y+5,BODYCOLOR);
Solid_Bar(x+3,y,x+1,y+3,BODYCOLOR);
Solid_Bar(x+2,y-1,x+0,y,BODYCOLOR);
Putpixel64k(x+2,y-2,BODYCOLOR);
Putpixel64k(x+1,y-2,BODYCOLOR);
break;

```

```

        }
    break;
    case MOVE_UP:
    case MOVE_DOWN:
        switch(frame)
        {
            case 1:
                Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head

                Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair

                Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

                Solid_Bar(x+2,y+11,x+3,y+21,color);
                Line_Plus(x+4,y+12,x+4,y+16,color);
                Line_Plus(x+1,y+16,x+1,y+20,color);
                Solid_Quadrangle(x+3,y+20,x+1,y+22,x-
4,y+17,x-2,y+15,BODYCOLOR);

                Line_Plus(x+19,y+9,x+19,y+11,color);
                Line_Plus(x+20,y+5,x+20,y+12,color);
                Line_Plus(x+21,y+2,x+21,y+12,color);
                Line_Plus(x+22,y+2,x+22,y+9,color);
                Line_Plus(x+23,y+3,x+23,y+5,color);
                Solid_Bar(x+21,y-
7,x+23,y+0, BODYCOLOR);

                Solid_Bar(x+6,y+29,x+10,y+38,color);
                Solid_Bar(x+6,y+40,x+10,y+47,BODYCOLO
R);

                Solid_Bar(x+13,y+29,x+22,y+33,color);
                Solid_Bar(x+18,y+35,x+22,y+42,BODYCOL
OR);

                break;
            case 2:
                Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head

                Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair

                Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

                Solid_Bar(x-4,y-1,x-2,y+6,BODYCOLOR);
                Solid_Quadrangle(x-2,y+5,x-
4,y+7,x+2,y+13,x+4,y+11,color);

                Solid_Bar(x+25,y-

```

```

1,x+27,y+6,BODYCOLOR);
Solid_Quadrangle(x+25,y+5,x+27,y+7,x+
21,y+13,x+19,y+11,color);
Solid_Bar(x+1,y+37,x+5,y+44,BODYCOLOR
);
Solid_Quadrangle(x+1,y+35,x+4,y+38,x+
10,y+32,x+7,y+29,color);
Line_Plus(x+8,y+29,x+10,y+31,color);
Putpixel64k(x+9,y+29,color);
Putpixel64k(x+10,y+30,color);
Solid_Bar(x+22,y+37,x+18,y+44,BODYCOL
OR);
Solid_Quadrangle(x+22,y+35,x+19,y+38,
x+13,y+32,x+16,y+29,color);
Line_Plus(x+13,y+31,x+15,y+29,color);
Putpixel64k(x+13,y+30,color);
Putpixel64k(x+14,y+29,color);
break;
case 3:
Solid_HalfSector_down(x+11,y+4,5,BODY
COLOR);//head
Solid_HalfSector_up(x+11,y+4,5,GRAY);
//hair
Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
Solid_Bar(x+21,y+11,x+20,y+21,color);
Line_Plus(x+19,y+12,x+19,y+16,color);
Line_Plus(x+22,y+16,x+22,y+20,color);
Solid_Quadrangle(x+20,y+20,x+22,y+22,
x+27,y+17,x+25,y+15,BODYCOLOR);
Line_Plus(x+4,y+9,x+4,y+11,color);
Line_Plus(x+3,y+5,x+3,y+12,color);
Line_Plus(x+2,y+2,x+2,y+12,color);
Line_Plus(x+1,y+2,x+1,y+9,color);
Line_Plus(x+0,y+3,x+0,y+5,color);
Solid_Bar(x+2,y-7,x+0,y+0,BODYCOLOR);
Solid_Bar(x+17,y+29,x+13,y+38,color);
Solid_Bar(x+17,y+40,x+13,y+47,BODYCOL
OR);
Solid_Bar(x+10,y+29,x+1,y+33,color);
Solid_Bar(x+5,y+35,x+1,y+42,BODYCOLOR
);
break;

```

```

    }
}
}
}

```

25. moveoftc

```

#include "headfile.h"

/*****
*****

* 函数名称      Teen_Movement
* 函数作用      青年动画库
* 函数输入      x,y 绘制坐标, forward 运动方向, move 动作, frame 关
键帧序号, color 衣服颜色
* 函数输出      无
*****
*****/

void Teen_Movement(int x,int y,char forward,char move,char frame,
int color)
{
    switch(move)
    {
        case RUN:
            switch(forward)
            {
                case MOVE_RIGHTUP:
                case MOVE_RIGHT:
                case MOVE_RIGHTDOWN:
                    switch(frame)
                    {
                        case 1:
                            Draw_Teen(x,y,color);
                            break;
                        case 2:
                            Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                            Solid_Bar( x+7,y+1,x+15,y+2,RED);
                            Putpixel64k(x+6,y+2,RED);
                            Putpixel64k(x+16,y+2,RED);
                            Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
                            Solid_Quadrangle(x+19,y+12,x+21,y+11,

```

```

x+24,y+20,x+22,y+21,color);//R
                                Solid_Quadrangle(x+23,y+23,x+25,y+22,
x+26,y+29,x+28,y+28,BODYCOLOR);
                                Solid_Quadrangle(x+2,y+11,x+4,y+12,x+
0,y+21,x+2,y+22,color);//L
                                Solid_Quadrangle(x+0,y+25,x+1,y+24,x+
2,y+31,x+4,y+30,BODYCOLOR);
                                Solid_Bar(x+6,y+29,x+10,y+33,color);//
/L
                                Solid_Bar(x+7,y+34,x+11,y+38,color);
                                Solid_Bar(x+5,y+40,x+9,y+42,BODYCOLOR
);
                                Solid_Bar(x+4,y+43,x+8,y+44,BODYCOLOR
);
                                Solid_Bar(x+3,y+45,x+7,y+47,BODYCOLOR
);
                                Solid_Bar(x+13,y+29,x+17,y+31,color);
//R
                                Solid_Bar(x+14,y+32,x+18,y+35,color);
                                Solid_Bar(x+15,y+36,x+19,y+38,color);
                                Solid_Quadrangle(x+15,y+40,x+19,y+40,
x+13,y+46,x+17,y+47,BODYCOLOR);
                                break;
                                case 3:
                                Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                                Solid_Bar(x+7,y+1,x+15,y+2,RED);
                                Putpixel64k(x+6,y+2,RED);
                                Putpixel64k(x+16,y+2,RED);
                                Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
                                Solid_Quadrangle(x+21,y+11,x+28,y+18,
x+26,y+20,x+19,y+14,color);//R
                                Solid_Quadrangle(x+28,y+22,x+29,y+20,
x+35,y+23,x+34,y+25,BODYCOLOR);
                                Solid_Quadrangle(x+2,y+11,x+4,y+14,x+
1,y+21,x-2,y+20,color);//L
                                Solid_Quadrangle(x-
1,y+23,x+1,y+29,x+0,y+30,x-3,y+24,BODYCOLOR);
                                Solid_Quadrangle(x+6,y+29,x+10,y+32,x
+7,y+38,x+3,y+37,color);//L
                                Solid_Quadrangle(x+0,y+39,x+3,y+40,x+
0,y+46,x-3,y+43,BODYCOLOR);

```

```

Solid_Quadrangle(x+13,y+31,x+19,y+38,
x+22,y+35,x+17,y+29,color);//R
Solid_Quadrangle(x+21,y+39,x+19,y+46,
x+23,y+46,x+25,y+40,BODYCOLOR);
Line_Plus(x+20,y+47,x+22,y+47,BODYCOL
OR);

break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Draw_Teen(x,y,color);
break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

Solid_Quadrangle(x+2,y+11,x-
1,y+20,x+1,y+21,x+4,y+12,color);//R
Solid_Quadrangle(x-2,y+22,x-5,y+28,x-
3,y+29,x+0,y+23,BODYCOLOR);
Solid_Quadrangle(x+21,y+11,x+23,y+21,
x+21,y+22,x+19,y+12,color);//L
Solid_Quadrangle(x+22,y+24,x+24,y+25,
x+21,y+31,x+19,y+30,BODYCOLOR);
Solid_Bar(x+13,y+29,x+17,y+33,color);
//L

Solid_Bar(x+12,y+34,x+16,y+38,color);
Solid_Bar(x+14,y+40,x+18,y+42,BODYCOL
OR);

Solid_Bar(x+15,y+43,x+19,y+44,BODYCOL
OR);

Solid_Bar(x+16,y+45,x+20,y+47,BODYCOL
OR);

Solid_Bar(x+6,y+29,x+10,y+31,color);/

```



```

/R
Solid_Bar(x+5,y+32,x+9,y+35,color);
Solid_Bar(x+4,y+36,x+8,y+38,color);
Solid_Quadrangle(x+8,y+40,x+4,y+40,x+
10,y+46,x+6,y+47,BODYCOLOR);
break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body
Solid_Quadrangle(x+2,y+11,x+4,y+13,x-
5,y+18,x-3,y+20,color);//R
Solid_Quadrangle(x-6,y+20,x-5,y+22,x-
11,y+25,x-12,y+23,BODYCOLOR);
Solid_Quadrangle(x+21,y+11,x+19,y+14,
x+23,y+21,x+26,y+20,color);//L
Solid_Quadrangle(x+25,y+23,x+22,y+29,
x+24,y+30,x+27,y+24,BODYCOLOR);
Solid_Quadrangle(x+17,y+29,x+13,y+31,
x+16,y+38,x+20,y+37,color);//L
Solid_Quadrangle(x+20,y+41,x+22,y+38,
x+28,y+42,x+26,y+45,BODYCOLOR);
Solid_Quadrangle(x+10,y+31,x+4,y+38,x
+1,y+35,x+6,y+29,color);//R
Solid_Quadrangle(x+2,y+39,x+4,y+45,x+
1,y+47,x-2,y+40,BODYCOLOR);
//Line_Plus(x+1,y+47,x+3,y+47,BODYCOL
OR);
break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Draw_Teen(x,y,color);
break;
case 2:

```

```

Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/

/body

Solid_Bar(x+2,y+11,x+4,y+21,color);
Solid_Bar(x+2,y+23,x+4,y+28,BODYCOLOR
);

Solid_Bar(x+19,y+11,x+21,y+20,color);
Solid_Bar(x+19,y+22,x+21,y+26,BODYCOL
OR);

Solid_Bar(x+6,y+29,x+10,y+37,color);
Solid_Bar(x+6,y+39,x+10,y+46,BODYCOLO
R);

Solid_Bar(x+13,y+29,x+17,y+36,color);
Solid_Bar(x+13,y+38,x+17,y+42,BODYCOL
OR);

break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/

/body

Solid_Bar(x+2,y+11,x+4,y+19,color);
Solid_Bar(x+2,y+22,x+4,y+25,BODYCOLOR
);

Solid_Bar(x+19,y+11,x+21,y+18,color);
Line_Plus(x+19,y+20,x+21,y+20,BODYCOL
OR);

Solid_Bar(x+6,y+29,x+10,y+37,color);
Solid_Bar(x+6,y+39,x+10,y+45,BODYCOLO
R);

Solid_Bar(x+13,y+29,x+17,y+34,color);
Solid_Bar(x+13,y+36,x+17,y+40,BODYCOL
OR);

break;
case 4:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//

```

```

    head
        Solid_Bar(x+7,y+1,x+15,y+2,RED);
        Putpixel64k(x+6,y+2,RED);
        Putpixel64k(x+16,y+2,RED);
        Solid_Bar(x+6,y+11,x+17,y+27,color);/
    /body
        Solid_Bar(x+19,y+11,x+21,y+21,color);
        Solid_Bar(x+19,y+23,x+21,y+28,BODYCOL
OR);

        Solid_Bar(x+2,y+11,x+4,y+20,color);
        Solid_Bar(x+2,y+22,x+4,y+26,BODYCOLOR
);

        Solid_Bar(x+13,y+29,x+17,y+37,color);
        Solid_Bar(x+13,y+39,x+17,y+46,BODYCOL
OR);

        Solid_Bar(x+6,y+29,x+10,y+36,color);
        Solid_Bar(x+6,y+38,x+10,y+42,BODYCOLO
R);

        break;
    case 5:
        Solid_Circle(x+11,y+4,5,BODYCOLOR);//
    head
        Solid_Bar(x+7,y+1,x+15,y+2,RED);
        Putpixel64k(x+6,y+2,RED);
        Putpixel64k(x+16,y+2,RED);
        Solid_Bar(x+6,y+11,x+17,y+27,color);/
    /body
        Solid_Bar(x+19,y+11,x+21,y+19,color);
        Solid_Bar(x+19,y+22,x+21,y+25,BODYCOL
OR);

        Solid_Bar(x+2,y+11,x+4,y+18,color);
        Line_Plus(x+2,y+20,x+4,y+20,BODYCOLOR
);

        Solid_Bar(x+13,y+29,x+17,y+37,color);
        Solid_Bar(x+13,y+39,x+17,y+45,BODYCOL
OR);

        Solid_Bar(x+6,y+29,x+10,y+34,color);
        Solid_Bar(x+6,y+36,x+10,y+40,BODYCOLO
R);

        break;
    }
    break;
}

```

```

break;
case WALK:
    switch(forward)
    {
        case MOVE_RIGHTUP:
        case MOVE_RIGHT:
        case MOVE_RIGHTDOWN:
            switch(frame)
            {
                case 1:
                    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                    Solid_Bar(x+7,y+1,x+15,y+2,RED);
                    Putpixel64k(x+6,y+2,RED);
                    Putpixel64k(x+16,y+2,RED);
                    Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
                    Solid_Bar(x+4,y+11,x+6,y+22,color);
                    Solid_Bar(x+4,y+24,x+6,y+31,BODYCOLOR
);
                    Solid_Bar(x+17,y+11,x+19,y+22,color);
                    Solid_Bar(x+17,y+24,x+19,y+31,BODYCOL
OR);
                    Solid_Bar(x+9,y+29,x+14,y+38,color);
                    Solid_Bar(x+9,y+40,x+14,y+47,BODYCOLO
R);
                    break;
                case 2:
                    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                    Solid_Bar(x+7,y+1,x+15,y+2,RED);
                    Putpixel64k(x+6,y+2,RED);
                    Putpixel64k(x+16,y+2,RED);
                    Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
                    // Solid_Quadrangle(x+4,y+11,x+6,y+12,
x+3,y+21,x+5,y+22,color);
                    Solid_Bar(x+3,y+24,x+5,y+31,BODYCOLOR
);
                    Solid_Bar(x+4,y+11,x+5,y+22,color);
                    Line_Plus(x+3,y+17,x+3,y+21,color);
                    Line_Plus(x+6,y+12,x+6,y+16,color);
                    Solid_Bar(x+18,y+11,x+19,y+22,color);

```

```

Line_Plus(x+17,y+12,x+17,y+16,color);
Line_Plus(x+20,y+17,x+20,y+21,color);
Solid_Bar(x+20,y+23,x+21,y+30,BODYCOL
OR);

Line_Plus(x+19,y+24,x+19,y+27,BODYCOL
OR);

Line_Plus(x+22,y+27,x+22,y+29,BODYCOL
OR);

Solid_Bar(x+9,y+30,x+13,y+32,color);
Solid_Bar(x+10,y+32,x+14,y+36,color);
Solid_Bar(x+11,y+36,x+15,y+38,color);
Line_Plus(x+11,y+29,x+13,y+29,color);
Line_Plus(x+11,y+39,x+13,y+39,color);

Solid_Quadrangle(x+12,y+41,x+15,y+47,
x+18,y+46,x+15,y+40,BODYCOLOR);//R
Line_Plus(x+10,y+29,x+12,y+29,color);

Solid_Bar(x+10,y+30,x+14,y+31,color);
Solid_Bar(x+9,y+31,x+13,y+34,color);
Solid_Bar(x+8,y+33,x+12,y+36,color);
Line_Plus(x+7,y+37,x+11,y+37,color);
Line_Plus(x+9,y+38,x+11,y+38,color);
Putpixel64k(x+8,y+36,color);
Line_Plus(x+6,y+39,x+8,y+39,BODYCOLOR
);

Solid_Bar(x+6,y+40,x+10,y+43,BODYCOLO
R);

Solid_Bar(x+5,y+43,x+9,y+46,BODYCOLOR
);

Line_Plus(x+7,y+47,x+9,y+47,BODYCOLOR
);

break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body

Solid_Bar(x+3,y+13,x+6,y+14,color);
Solid_Bar(x+2,y+15,x+5,y+16,color);

```

```

Solid_Bar(x+1,y+17,x+4,y+18,color);
Solid_Bar(x,y+19,x+3,y+20,color);
Line_Plus(x+4,y+12,x+6,y+12,color);
Line_Plus(x+0,y+21,x+2,y+21,color);
Solid_Bar(x+0,y+23,x+2,y+30,BODYCOLOR
);

Line_Plus(x+17,y+12,x+19,y+12,color);

Line_Plus(x+21,y+21,x+23,y+21,color);
Solid_Bar(x+17,y+13,x+20,y+14,color);
Solid_Bar(x+18,y+15,x+21,y+16,color);
Solid_Bar(x+19,y+17,x+22,y+18,color);
Solid_Bar(x+20,y+19,x+23,y+20,color);
Solid_Quadrangle(x+23,y+24,x+25,y+23,
x+27,y+28,x+29,y+27,BODYCOLOR);
Solid_Quadrangle(x+9,y+30,x+13,y+29,x
+13,y+39,x+17,y+38,color);
Solid_Quadrangle(x+14,y+41,x+19,y+47,
x+22,y+46,x+17,y+40,BODYCOLOR);
Solid_Quadrangle(x+10,y+29,x+14,y+30,
x+6,y+37,x+10,y+38,color);
Line_Plus(x+5,y+39,x+7,y+39,BODYCOLOR
);

Solid_Bar(x+5,y+40,x+9,y+43,BODYCOLOR
);

Solid_Bar(x+4,y+43,x+8,y+46,BODYCOLOR
);

Line_Plus(x+6,y+47,x+8,y+47,BODYCOLOR
);

break;
}
break;
case MOVE_LEFTUP:
case MOVE_LEFT:
case MOVE_LEFTDOWN:
switch(frame)
{
case 1:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);

```

```

Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

Solid_Bar(x+4,y+11,x+6,y+22,color);
Solid_Bar(x+4,y+24,x+6,y+31,BODYCOLOR
);

Solid_Bar(x+17,y+11,x+19,y+22,color);
Solid_Bar(x+17,y+24,x+19,y+31,BODYCOL
OR);

Solid_Bar(x+9,y+29,x+14,y+38,color);
Solid_Bar(x+9,y+40,x+14,y+47,BODYCLO
R);

break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

// Solid_Quadrangle(x+4,y+11,x+6,y+12,
x+3,y+21,x+5,y+22,color);
Solid_Bar(x+20,y+24,x+18,y+31,BODYCOL
OR);

Solid_Bar(x+19,y+11,x+18,y+22,color);
Line_Plus(x+20,y+17,x+20,y+21,color);
Line_Plus(x+17,y+12,x+17,y+16,color);
Solid_Bar(x+5,y+11,x+4,y+22,color);
Line_Plus(x+6,y+12,x+6,y+16,color);
Line_Plus(x+3,y+17,x+3,y+21,color);
Solid_Bar(x+3,y+23,x+2,y+30,BODYCOLOR
);

Line_Plus(x+4,y+24,x+4,y+27,BODYCOLOR
);

Line_Plus(x+1,y+27,x+1,y+29,BODYCOLOR
);

Solid_Bar(x+14,y+30,x+10,y+32,color);
Solid_Bar(x+13,y+32,x+9,y+36,color);
Solid_Bar(x+12,y+36,x+8,y+38,color);
Line_Plus(x+12,y+29,x+10,y+29,color);
Line_Plus(x+12,y+39,x+10,y+39,color);

Solid_Quadrangle(x+11,y+41,x+8,y+47,x

```

```

+5,y+46,x+8,y+40,BODYCOLOR);//R
        Line_Plus(x+13,y+29,x+11,y+29,color);

        Solid_Bar(x+13,y+30,x+9,y+31,color);
        Solid_Bar(x+14,y+31,x+10,y+34,color);
        Solid_Bar(x+15,y+33,x+13,y+36,color);
        Line_Plus(x+16,y+37,x+12,y+37,color);
        Line_Plus(x+14,y+38,x+12,y+38,color);
        Putpixel64k(x+15,y+36,color);
        Line_Plus(x+17,y+39,x+15,y+39,BODYCOL
OR);

        Solid_Bar(x+17,y+40,x+13,y+43,BODYCOL
OR);

        Solid_Bar(x+18,y+43,x+14,y+46,BODYCOL
OR);

        Line_Plus(x+16,y+47,x+14,y+47,BODYCOL
OR);

        break;
    case 3:
        Solid_Circle(x+11,y+4,5,BODYCOLOR);//
    head

        Solid_Bar(x+7,y+1,x+15,y+2,RED);
        Putpixel64k(x+6,y+2,RED);
        Putpixel64k(x+16,y+2,RED);
        Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body

        Solid_Bar(x+20,y+13,x+17,y+14,color);
        Solid_Bar(x+21,y+15,x+18,y+16,color);
        Solid_Bar(x+22,y+17,x+19,y+18,color);
        Solid_Bar(x+23,y+19,x+20,y+20,color);
        Line_Plus(x+19,y+12,x+17,y+12,color);
        Line_Plus(x+23,y+21,x+21,y+21,color);
        Solid_Bar(x+23,y+23,x+21,y+30,BODYCOL
OR);

        Line_Plus(x+6,y+12,x+4,y+12,color);
        Line_Plus(x+2,y+21,x+0,y+21,color);
        Solid_Bar(x+6,y+13,x+3,y+14,color);
        Solid_Bar(x+5,y+15,x+2,y+16,color);
        Solid_Bar(x+4,y+17,x+1,y+18,color);
        Solid_Bar(x+3,y+19,x+0,y+20,color);
        Solid_Quadrangle(x+0,y+24,x-2,y+23,x-
4,y+28,x-6,y+27,BODYCOLOR);
        Solid_Quadrangle(x+14,y+30,x+10,y+29,

```



```

x+10,y+39,x+6,y+38,color);
Solid_Quadrangle(x+8,y+41,x+3,y+47,x+
0,y+46,x+5,y+40,BODYCOLOR);
Solid_Quadrangle(x+13,y+29,x+9,y+30,x
+17,y+37,x+13,y+38,color);
Line_Plus(x+18,y+39,x+16,y+39,BODYCOL
OR);
Solid_Bar(x+18,y+40,x+14,y+43,BODYCOL
OR);
Solid_Bar(x+19,y+43,x+15,y+46,BODYCOL
OR);
Line_Plus(x+17,y+47,x+15,y+47,BODYCOL
OR);

break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
switch(frame)
{
case 1:
Draw_Teen(x,y,color);
break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

Solid_Bar(x+2,y+11,x+4,y+23,color);
Solid_Bar(x+2,y+25,x+4,y+33,BODYCOLOR
);
Solid_Bar(x+19,y+11,x+21,y+21,color);
Solid_Bar(x+19,y+23,x+21,y+29,BODYCOL
OR);
Solid_Bar(x+17,y+29,x+13,y+39,color);
Solid_Bar(x+17,y+41,x+13,y+49,BODYCOL
OR);
Solid_Bar(x+10,y+29,x+6,y+36,color);
Solid_Bar(x+10,y+38,x+6,y+45,BODYCOLO
R);

```

```

        break;
    case 3:
        Solid_Circle(x+11,y+4,5,BODYCOLOR);//
    head

        Solid_Bar(x+7,y+1,x+15,y+2,RED);
        Putpixel64k(x+6,y+2,RED);
        Putpixel64k(x+16,y+2,RED);
        Solid_Bar(x+6,y+11,x+17,y+27,color);//
    /body

        Solid_Bar(x+2,y+11,x+4,y+23,color);
        Solid_Bar(x+2,y+25,x+4,y+34,BODYCOLOR
    );

        Solid_Bar(x+19,y+11,x+21,y+20,color);
        Solid_Bar(x+19,y+22,x+21,y+27,BODYCOL
    OR);

        Solid_Bar(x+17,y+29,x+13,y+40,color);
        Solid_Bar(x+17,y+42,x+13,y+51,BODYCOL
    OR);

        Solid_Bar(x+10,y+29,x+6,y+35,color);
        Solid_Bar(x+10,y+37,x+6,y+43,BODYCOLO
    R);

        break;
    case 4:
        Solid_Circle(x+11,y+4,5,BODYCOLOR);//
    head

        Solid_Bar(x+7,y+1,x+15,y+2,RED);
        Putpixel64k(x+6,y+2,RED);
        Putpixel64k(x+16,y+2,RED);
        Solid_Bar(x+6,y+11,x+17,y+27,color);//
    /body

        Solid_Bar(x+21,y+11,x+19,y+23,color);
        Solid_Bar(x+21,y+25,x+19,y+33,BODYCOL
    OR);

        Solid_Bar(x+4,y+11,x+2,y+21,color);
        Solid_Bar(x+4,y+23,x+2,y+29,BODYCOLOR
    );

        Solid_Bar(x+6,y+29,x+10,y+39,color);
        Solid_Bar(x+6,y+41,x+10,y+49,BODYCOLO
    R);

        Solid_Bar(x+13,y+29,x+17,y+36,color);
        Solid_Bar(x+13,y+38,x+17,y+45,BODYCOL
    OR);

        break;

```

```

                                case 5:
                                    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                                Solid_Bar(x+7,y+1,x+15,y+2,RED);
                                Putpixel64k(x+6,y+2,RED);
                                Putpixel64k(x+16,y+2,RED);
                                Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
                                Solid_Bar(x+21,y+11,x+19,y+23,color);
                                Solid_Bar(x+21,y+25,x+19,y+34,BODYCOL
OR);
                                Solid_Bar(x+4,y+11,x+2,y+20,color);
                                Solid_Bar(x+4,y+22,x+2,y+27,BODYCOLOR
);
                                Solid_Bar(x+6,y+29,x+10,y+40,color);
                                Solid_Bar(x+6,y+42,x+10,y+51,BODYCOLO
R);
                                Solid_Bar(x+13,y+29,x+17,y+35,color);
                                Solid_Bar(x+13,y+37,x+17,y+43,BODYCOL
OR);
                                break;
                                }
                                break;
                                }
break;
case CLIMB:
    switch(forward)
    {
        case MOVE_RIGHT:
        case MOVE_RIGHTUP:
        case MOVE_RIGHTDOWN:
            switch(frame)
            {
                case 1:
head
                    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
                    Solid_Bar(x+7,y+1,x+15,y+2,RED);
                    Putpixel64k(x+6,y+2,RED);
                    Putpixel64k(x+16,y+2,RED);
                    Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
                    Solid_Bar(x+10,y+29,x+14,y+38,color);
                    Solid_Bar(x+10,y+40,x+14,y+47,BODYCOL

```

```

OR);

Solid_Bar(x+15,y+29,x+17,y+33,color);
Solid_Bar(x+19,y+30,x+22,y+37,BODYCOL

OR);

Solid_Quadrangle(x+16,y+14,x+18,y+16,
x+21,y+9,x+23,y+11,color);
Solid_Bar(x+22,y+1,x+24,y+8,BODYCOLOR

);

break;
case 2:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+8,y+11,x+15,y+27,color);/
/body
Solid_Quadrangle(x+11,y+32,x+14,y+29,
x+17,y+38,x+20,y+35,color);
Solid_Bar(x+18,y+38,x+21,y+45,BODYCOL

OR);

Solid_Bar(x+22,y+11,x+24,y+17,BODYCOL

OR);

Solid_Quadrangle(x+16,y+14,x+18,y+12,
x+21,y+19,x+23,y+17,color);
Solid_Bar(x+22,y+18,x+24,y+20,color);
Solid_Bar(x+16,y+11,x+18,y+13,color);
Solid_Bar(x+17,y+9,x+19,y+11,color);
Solid_Bar(x+18,y+7,x+20,y+9,color);
Putpixel64k(x+18,y+6,color);
Putpixel64k(x+19,y+6,color);
Solid_Bar(x+19,y+3,x+21,y+4,BODYCOLOR

);

Putpixel64k(x+21,y+5,BODYCOLOR);
Solid_Bar(x+20,y,x+22,y+3,BODYCOLOR);
Solid_Bar(x+21,y-1,x+23,y,BODYCOLOR);
Putpixel64k(x+21,y-2,BODYCOLOR);
Putpixel64k(x+22,y-2,BODYCOLOR);
break;
}
break;
case MOVE_LEFT:
case MOVE_LEFTUP:

```

```

        case MOVE_LEFTDOWN:
            switch(frame)
            {
                case 1:
                    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                    Solid_Bar(x+7,y+1,x+15,y+2,RED);
                    Putpixel64k(x+6,y+2,RED);
                    Putpixel64k(x+16,y+2,RED);
                    Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
                    Solid_Bar(x+13,y+29,x+9,y+38,color);
                    Solid_Bar(x+13,y+40,x+9,y+47,BODYCOLOR);
R);
                    Solid_Bar(x+8,y+29,x+6,y+33,color);
                    Solid_Bar(x+4,y+30,x+1,y+37,BODYCOLOR);
);
                    Solid_Quadrangle(x+7,y+14,x+5,y+16,x+
2,y+9,x+0,y+11,color);
                    Solid_Bar(x+1,y+1,x-1,y+8,BODYCOLOR);
                    break;
                case 2:
                    Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
                    Solid_Bar(x+7,y+1,x+15,y+2,RED);
                    Putpixel64k(x+6,y+2,RED);
                    Putpixel64k(x+16,y+2,RED);
                    Solid_Bar(x+8,y+11,x+15,y+27,color);//
/body
                    Solid_Quadrangle(x+12,y+32,x+9,y+29,x
+6,y+38,x+3,y+35,color);
                    Solid_Bar(x+5,y+38,x+2,y+45,BODYCOLOR);
);
                    Solid_Bar(x+1,y+11,x-
1,y+17,BODYCOLOR);
                    Solid_Quadrangle(x+7,y+14,x+5,y+12,x+
2,y+19,x+0,y+17,color);
                    Solid_Bar(x+1,y+18,x-1,y+20,color);
                    Solid_Bar(x+7,y+11,x+5,y+13,color);
                    Solid_Bar(x+6,y+9,x+4,y+11,color);
                    Solid_Bar(x+5,y+7,x+3,y+9,color);
                    Putpixel64k(x+5,y+6,color);
                    Putpixel64k(x+4,y+6,color);

```

```

Solid_Bar(x+4,y+3,x+2,y+4,BODYCOLOR);
Putpixel64k(x+2,y+5,BODYCOLOR);
Solid_Bar(x+3,y,x+1,y+3,BODYCOLOR);
Solid_Bar(x+2,y-1,x+0,y,BODYCOLOR);
Putpixel64k(x+2,y-2,BODYCOLOR);
Putpixel64k(x+1,y-2,BODYCOLOR);
break;
}
break;
case MOVE_UP:
case MOVE_DOWN:
    switch(frame)
    {
        case 1:
            Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head
            Solid_Bar(x+7,y+1,x+15,y+2,RED);
            Putpixel64k(x+6,y+2,RED);
            Putpixel64k(x+16,y+2,RED);
            Solid_Bar(x+6,y+11,x+17,y+27,color);//
/body
            Solid_Bar(x+2,y+11,x+3,y+21,color);
            Line_Plus(x+4,y+12,x+4,y+16,color);
            Line_Plus(x+1,y+16,x+1,y+20,color);
            Solid_Quadrangle(x+3,y+20,x+1,y+22,x-
4,y+17,x-2,y+15,BODYCOLOR);
            Line_Plus(x+19,y+9,x+19,y+11,color);
            Line_Plus(x+20,y+5,x+20,y+12,color);
            Line_Plus(x+21,y+2,x+21,y+12,color);
            Line_Plus(x+22,y+2,x+22,y+9,color);
            Line_Plus(x+23,y+3,x+23,y+5,color);
            Solid_Bar(x+21,y-
7,x+23,y+0,BODYCOLOR);
            Solid_Bar(x+6,y+29,x+10,y+38,color);
            Solid_Bar(x+6,y+40,x+10,y+47,BODYCOLO
R);
            Solid_Bar(x+13,y+29,x+22,y+33,color);
            Solid_Bar(x+18,y+35,x+22,y+42,BODYCOL
OR);
            break;
        case 2:
            Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

```

```

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

Solid_Bar(x-4,y-1,x-2,y+6,BODYCOLOR);
Solid_Quadrangle(x-2,y+5,x-
4,y+7,x+2,y+13,x+4,y+11,color);
Solid_Bar(x+25,y-
1,x+27,y+6,BODYCOLOR);
Solid_Quadrangle(x+25,y+5,x+27,y+7,x+
21,y+13,x+19,y+11,color);
Solid_Bar(x+1,y+37,x+5,y+44,BODYCOLOR
);
Solid_Quadrangle(x+1,y+35,x+4,y+38,x+
10,y+32,x+7,y+29,color);
Line_Plus(x+8,y+29,x+10,y+31,color);
Putpixel64k(x+9,y+29,color);
Putpixel64k(x+10,y+30,color);
Solid_Bar(x+22,y+37,x+18,y+44,BODYCOL
OR);
Solid_Quadrangle(x+22,y+35,x+19,y+38,
x+13,y+32,x+16,y+29,color);
Line_Plus(x+13,y+31,x+15,y+29,color);
Putpixel64k(x+13,y+30,color);
Putpixel64k(x+14,y+29,color);
break;
case 3:
Solid_Circle(x+11,y+4,5,BODYCOLOR);//
head

Solid_Bar(x+7,y+1,x+15,y+2,RED);
Putpixel64k(x+6,y+2,RED);
Putpixel64k(x+16,y+2,RED);
Solid_Bar(x+6,y+11,x+17,y+27,color);/
/body

Solid_Bar(x+21,y+11,x+20,y+21,color);
Line_Plus(x+19,y+12,x+19,y+16,color);
Line_Plus(x+22,y+16,x+22,y+20,color);
Solid_Quadrangle(x+20,y+20,x+22,y+22,
x+27,y+17,x+25,y+15,BODYCOLOR);
Line_Plus(x+4,y+9,x+4,y+11,color);
Line_Plus(x+3,y+5,x+3,y+12,color);
Line_Plus(x+2,y+2,x+2,y+12,color);

```



```

#define Cancel_x1 (General_x0+326)
#define Cancel_y1 (General_y0+371)

//button 0
#define Message_button0_x0 (General_x0+8)
#define Message_button0_y0 (General_y0+19)
#define Message_button0_x1 (General_x1-8)
#define Message_button0_y1 (General_y1-19)

/*****
*****
* 函数名称      MsgBox2_Draw_Setting
* 函数作用      画 2 个按钮的消息弹窗
* 函数输入      无
* 函数输出      无
*****
*****/
void MsgBox2_Draw_Setting()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_x0,Message_y0,Message_x1,Message_y1,Hollo
w_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,24,WHITE,"注意: 未完全设
置!");
    puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"点击确定将采用
默认值");
    puthz(Message_x0+14,Message_y0+86,24,26,WHITE,"点击取消以继续
进行设置");

    //buttons: 328*86 DARK_GRAY + LIGHT_GRAY
    Button_Draw(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,Hollo
w_Color,Solid_Color);
    puthz(Confirm_x0+130,Confirm_y0+25,32,32,BLACK,"确定");

    Button_Draw(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,Hollow_Co
lor,Solid_Color);
    puthz(Cancel_x0+130,Cancel_y0+25,32,32,BLACK,"取消");
}

```

```

/*****
*****

* 函数名称      MsgBox2Button_Light
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

void MsgBox2Button_Light(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,WHITE);
            Solid_Bar(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1
,GREEN);
            puthz(Confirm_x0+130,Confirm_y0+25,32,32,GRAY,"确定
");
            break;
        }
        case 2:
        {
            Button_Edge(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,W
HITE);
            Solid_Bar(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,RED
);
            puthz(Cancel_x0+130,Cancel_y0+25,32,32,GRAY,"取消");
            break;
        }
    }
}

/*****
*****

* 函数名称      Button_Darken_Help
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

void MsgBox2Button_Darken(int flag)

```

```

{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,GRAY);
            Button_Draw(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,Hollow_Color,Solid_Color);
            puthz(Confirm_x0+130,Confirm_y0+25,32,32,BLACK,"确定
");
            break;
        }
        case 2:
        {
            Button_Edge(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,G
RAY);
            Button_Draw(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,H
ollow_Color,Solid_Color);
            puthz(Cancel_x0+130,Cancel_y0+25,32,32,BLACK,"取消");
            break;
        }
    }
}

```

```

/*****
*****
* 函数名称      MsgBox2_Draw0_Play
* 函数作用      画0个按钮的消息弹窗
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void MsgBox0_Draw0_Play()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_button0_x0,Message_button0_y0,Message_but
ton0_x1,Message_button0_y1,Hollow_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"请点击地图上对

```

```

应位置，");
    puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"规划自己的路
径！");
    puthz(Message_x0+14,Message_y0+86,24,26,WHITE,"注意：");
    puthz(Message_x0+14,Message_y0+122,24,26,WHITE,"上一选择位置到
本次选择");
    puthz(Message_x0+14,Message_y0+158,24,24,WHITE,"位置的路径仅能
为水平线、");
    puthz(Message_x0+14,Message_y0+194,24,26,WHITE,"竖直线或斜");
    putzm(Message_x0+138,Message_y0+194,24,13,WHITE,"45");
    puthz(Message_x0+170,Message_y0+194,24,26,WHITE,"度线！");
    puthz(Message_x0+14,Message_y0+230,24,26,WHITE,"不能选择与上一
选择位置");
    puthz(Message_x0+14,Message_y0+266,24,26,WHITE,"超过");
    putzm(Message_x0+60,Message_y0+266,24,13,WHITE,"10");
    puthz(Message_x0+92,Message_y0+266,24,26,WHITE,"格的位置！");
    puthz(Message_x0+14,Message_y0+308,24,26,WHITE,"请按回车键开始
选择！");
}

```

```

/*****
*****
* 函数名称      MsgBox2_Draw0_Notice_Play
* 函数作用      画0个按钮的消息弹窗
* 函数输入      flag 消息索引
* 函数输出      无
*****
*****/
void MsgBox0_Draw0_Notice_Play(int flag)
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);
    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_button0_x0,Message_button0_y0,Message_but
ton0_x1,Message_button0_y1,Hollow_Color,BLACK);
    switch (flag) {
        case 1:
            {
                puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"注意：
所选路径错误！");
                puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"请确保
路径皆可通行！");
            }
        }
    }
}

```

```

        puthz(Message_x0+14,Message_y0+98,24,26,WHITE,"如需了解更多, 请参阅帮");
        puthz(Message_x0+14,Message_y0+134,24,26,WHITE,"助界面下的图例板块");
        puthz(Message_x0+14,Message_y0+182,24,26,WHITE,"请按回车键重新选择! ");
        break;
    }
    case 2:
    {
        puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"注意: 所选位置错误! ");
        puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"请确保择位置距离上一选");
        puthz(Message_x0+14,Message_y0+86,24,26,WHITE,"择位置距离小于");
        putzm(Message_x0+190,Message_y0+86,24,13,WHITE,"10");
        puthz(Message_x0+222,Message_y0+86,24,26,WHITE,"格!");
    };
        puthz(Message_x0+14,Message_y0+134,24,26,WHITE,"请按回车键重新选择! ");
        break;
    }
    case 3:
    {
        puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"注意: 所选路径错误! ");
        puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"请确保上一选择位置到本");
        puthz(Message_x0+14,Message_y0+86,24,26,WHITE,"次选择位置的路径为水平");
        puthz(Message_x0+14,Message_y0+122,24,26,WHITE,"线、竖直线或斜");
        putzm(Message_x0+190,Message_y0+122,24,13,WHITE,"45");
    };
        puthz(Message_x0+222,Message_y0+122,24,26,WHITE,"度线! ");
        puthz(Message_x0+14,Message_y0+170,24,26,WHITE,"请按回车键重新选择! ");
        break;
    }
    case 4:
    {

```

```

        puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"注意:
所选点位错误!");
        puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"请确保
点位依次选择!");
        puthz(Message_x0+14,Message_y0+98,24,26,WHITE,"请按回
车键重新选择!");
        break;
    }
}
}

/*****
*****
* 函数名称      MsgBox0_Draw1_Play
* 函数作用      画0个按钮的消息弹窗
* 函数输入      your_number 用户参赛序号
* 函数输出      无
*****
*****/
void MsgBox0_Draw1_Play(int your_number)
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_button0_x0,Message_button0_y0,Message_but
ton0_x1,Message_button0_y1,Hollow_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"模拟地图上单个
方块边长");
    puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"为六十米");
    puthz(Message_x0+14,Message_y0+86,24,26,WHITE,"上方状态栏显示
比赛信息");
    puthz(Message_x0+14,Message_y0+122,24,26,WHITE,"和当前时间");
    puthz(Message_x0+14,Message_y0+158,24,26,WHITE,"左边状态栏显示
比赛成员");
    puthz(Message_x0+14,Message_y0+194,24,26,WHITE,"和完成进度");
    puthz(Message_x0+14,Message_y0+242,24,26,WHITE,"你所分配的序号
是:");
    putsz(Message_x0+248,Message_y0+242,24,26,RED,your_number);
    puthz(Message_x0+14,Message_y0+290,24,26,WHITE,"请按回车键开始
模拟!");
}

```

```

/*****
*****
* 函数名称      MsgBox0_Draw2_Play
* 函数作用      画0个按钮的消息弹窗
* 函数输入      无
* 函数输出      无
*****
*****/
void MsgBox0_Draw2_Play()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_button0_x0,Message_button0_y0,Message_but
ton0_x1,Message_button0_y1,Hollow_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"本次比赛模拟结
束!");
    puthz(Message_x0+14,Message_y0+62,24,26,WHITE,"请按回车键进入
搜索界面");
}

/*****
*****
* 函数名称      MsgBox2_Draw_Search
* 函数作用      画2个按钮的消息弹窗
* 函数输入      无
* 函数输出      无
*****
*****/
void MsgBox2_Draw_Search()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_x0,Message_y0,Message_x1,Message_y1,Hollo
w_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,24,WHITE,"注意: 退出后将
无法搜索");
}

```

```

        puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"本次比赛!");
        puthz(Message_x0+14,Message_y0+86,24,26,WHITE,"点击确定将返回
主菜单");
        puthz(Message_x0+14,Message_y0+122,24,26,WHITE,"点击取消继续进行
搜索");

        //buttons: 328*86 DARK_GRAY + LIGHT_GRAY
        Button_Draw(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,Hollow_
w_Color,Solid_Color);
        puthz(Confirm_x0+130,Confirm_y0+25,32,32,BLACK,"确定");

        Button_Draw(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,Hollow_Co
lor,Solid_Color);
        puthz(Cancel_x0+130,Cancel_y0+25,32,32,BLACK,"取消");
    }

/*****
*****
* 函数名称      MsgBox0NoInput_Draw_Search
* 函数作用      画0个按钮的消息弹窗
* 函数输入      无
* 函数输出      无
*****
*****/
void MsgBox0NoInput_Draw_Search()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_button0_x0,Message_button0_y0,Message_but
ton0_x1,Message_button0_y1,Hollow_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"未输入数值,请
检查!");
    puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"请按回车键继续
查询");
}

/*****
*****
* 函数名称      MsgBox0NoSearch_Draw_Search
* 函数作用      画0个按钮的消息弹窗

```



```

* 函数输入      无
* 函数输出      无
*****
*****/
void MsgBox0NoSearch_Draw_Search()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_button0_x0,Message_button0_y0,Message_but
ton0_x1,Message_button0_y1,Hollow_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,24,26,WHITE,"未输入正确数
值, 请检查");
    puthz(Message_x0+14,Message_y0+50,24,26,WHITE,"请按回车键继续
查询");
}

```

27. people.c

```

#include"headfile.h"

/*****
*****
* @author      ytm
* @date        2022-4-15
*****
*****/

/*****
*****
* 函数名称      People_Init
* 函数作用      初始化参赛人员数据结构体
* 函数输入      person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无
*****
*****/
void People_Init(people_basic *person_basic,people_competition *p
erson_competition)
{
    int i,j;
    for(i=0;i<6;i++)
    {

```

```

        person_basic[i].number=i+1;
        person_basic[i].ps_consume=0;
        person_basic[i].ps_recover=0;
        person_basic[i].verb_walk=0;
        person_basic[i].verb_run=0;
        person_basic[i].verb_climb=0;
        person_competition[i].distance=0;
        person_competition[i].rank=i+1;
        person_competition[i].hour_score=0;
        person_competition[i].minute_score=0;
        for(j=0;j<4;j++)
        {
            person_competition[i].hour_point[j]=0;
            person_competition[i].minute_point[j]=0;
            person_competition[i].hour_interval[j]=0;
            person_competition[i].minute_interval[j]=0;
        }
    }
}

/*****
*****
* 函数名称      People_Random
* 函数作用      随机化参赛人员基本信息
* 函数输入      people 人数, age 年龄, weather 天气类型,
clock_hour, clock_minute 当前时间
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无
*****
*****/
void People_Random(int people,int age,int weather,int clock_hour,
int clock_minute,people_basic *person_basic,people_competition *p
erson_competition,int flag_debug)
{
    int i,j[6],addminute[6];
    FILE *fp;
    randomize();
    for(i=0;i<6;i++)
    {
        addminute[i]=random(7)+3;
        if(age==1||age==3)
        {

```

```

        j[i]=random(3);
    }
    else
    {
        j[i]=random(2);
    }
}
for(i=0;i<6;i++)
{
    //random start time
    if((clock_minute+adminute[i])<60)
    {
        person_competition[i].hour_point[0]=clock_hour;
        person_competition[i].minute_point[0]=clock_minute+ad
dminute[i];
    }
    else
    {
        person_competition[i].hour_point[0]=clock_hour+1;
        person_competition[i].minute_point[0]=clock_minute+ad
dminute[i]-60;
    }
    //random ps & verb
    switch(age)
    {
    case 1:
    {
        person_basic[i].ps_consume=20+5*j[i];
        person_basic[i].ps_recover=5;
        person_basic[i].verb_walk=1;
        person_basic[i].verb_run=2;
        person_basic[i].verb_climb=0.5;
        break;
    }
    case 2:
    {
        person_basic[i].ps_consume=45+15*j[i];
        person_basic[i].ps_recover=15;
        person_basic[i].verb_walk=2;
        person_basic[i].verb_run=4;
        person_basic[i].verb_climb=1.5;
        break;
    }
    }
}

```

```

        case 3:
        {
            person_basic[i].ps_consume=30+10*j[i];
            person_basic[i].ps_recover=10;
            person_basic[i].verb_walk=1.5;
            person_basic[i].verb_run=3;
            person_basic[i].verb_climb=1;
            break;
        }
        case 4:
        {
            person_basic[i].ps_consume=10+10*j[i];
            person_basic[i].ps_recover=5;
            person_basic[i].verb_walk=1;
            person_basic[i].verb_run=1.5;
            person_basic[i].verb_climb=0.5;
            break;
        }
    }
    //adjust verb
    switch(weather)
    {
        case 1:
        {
            person_basic[i].verb_walk=0.8*person_basic[i].verb_walk;
            person_basic[i].verb_run=0.5*person_basic[i].verb_run;
            person_basic[i].verb_climb=0.5*person_basic[i].verb_climb;
            break;
        }
        case 3:
        {
            person_basic[i].verb_walk=0.5*person_basic[i].verb_walk;
            person_basic[i].verb_run=0.5*person_basic[i].verb_run;
            person_basic[i].verb_climb=0.5*person_basic[i].verb_climb;
            break;
        }
    }
}

```

```

    }
    if(flag_debug)
    {
        fp = fopen (".\\PEOPLE\\Basic.txt","w+");
        for(i=0;i<people;i++)
        {
            fprintf(fp,"Basic Information\n");
            fprintf(fp,"Number: %d Number In Competition: %d\n",p
erson_basic[i].number,person_basic[i].number+1);
            fprintf(fp,"Start time: %d:%d\n",person_competition[i
].hour_point[0],person_competition[i].minute_point[0]);
            fprintf(fp,"PS:\nps_consume: %d\nps_recover: %d\n",pe
rson_basic[i].ps_consume,person_basic[i].ps_recover);
            fprintf(fp,"Verb\nwalk_verb: %f\nrun_verb: %f\nclimb_
verb: %f\n",person_basic[i].verb_walk,person_basic[i].verb_run,pe
rson_basic[i].verb_climb);
            fprintf(fp,"\n");
        }
        fclose(fp);
    }
}

```

```

/*****
*****
* 函数名称      People_Interval
* 函数作用      计算参赛人员比赛信息
* 函数输入      people 人数, point 点位数量
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无
*****
*****/

```

```

void People_Interval(int people,int point,people_basic *person_ba
sic,people_competition *person_competition)
{
    int i,j,minute,hour;
    for(i=0;i<people;i++)
    {
        person_competition[i].hour_interval[0]=0;
        person_competition[i].minute_interval[0]=0;
        for(j=1;j<point;j++)
        {
            minute=Clock_Minus(person_competition[i].hour_point[j

```

```

],person_competition[i].minute_point[j],person_competition[i].hour_point[j-1],person_competition[i].minute_point[j-1]);
    if(minute>=60)
    {
        hour=(minute-minute%60)/60;
        minute=minute%60;
    }
    else
    {
        hour=0;
    }
    person_competition[i].hour_interval[j]=hour;
    person_competition[i].minute_interval[j]=minute;
}
}
}

```

```

/*****
*****

```

```

* 函数名称      People_Score
* 函数作用      计算参赛人员比赛信息
* 函数输入      people 人数, point 点位数量
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无

```

```

*****
*****/

```

```

void People_Score(int people,int point,people_basic *person_basic
,people_competition *person_competition)
{
    int i,hour,minute;
    for(i=0;i<people;i++)
    {
        minute=Clock_Minus(person_competition[i].hour_point[point-1],person_competition[i].minute_point[point-1],person_competition[i].hour_point[0],person_competition[i].minute_point[0]);
        if(minute>=60)
        {
            hour=(minute-minute%60)/60;
            minute=minute%60;
        }
        else

```

```

        {
            hour=0;
        }
        person_competition[i].hour_score=hour;
        person_competition[i].minute_score=minute;
    }
}

/*****
*****
* 函数名称      People_Rank
* 函数作用      计算参赛人员比赛信息
* 函数输入      people 人数, person_basic 比赛成员基本信息结构体地址, person_competition 比赛成员参赛信息结构体地址
* 函数输出      无
*****
*****/
void People_Rank(int people, people_basic *person_basic, people_competition *person_competition)
{
    int i, j, temp, rank[6];
    for(i=0; i<people; i++)
    {
        rank[i]=i;
    }
    for(i=0; i<people; i++)
    {
        for(j=people-1; j>0; j--)
        {
            if((60*person_competition[rank[j]].hour_score+person_competition[rank[j]].minute_score)<(60*person_competition[rank[j-1]].hour_score+person_competition[rank[j-1]].minute_score))
            {
                temp=rank[j];
                rank[j]=rank[j-1];
                rank[j-1]=temp;
            }
        }
    }
    for(i=0; i<people; i++)
    {
        person_competition[rank[i]].rank=i+1;
    }
}

```

```

    }
}
28. play.c
#include "headfile.h"

/*****
*****

* @author          ytm
* @date            2022-4-19b
*****
*****/

#define General_x0 344
#define General_y0 230
#define General_x1 680
#define General_y1 610

#define Arrow_Color RED

/*****
*****

* 函数名称          Play
* 函数作用          模拟界面
* 函数输入          page 界面地址, people 人数, age 年龄, zone 区域类型,
weather 天气类型, point 点位数量, point_x 点位x 值地址
*                  point_y 点位y 值地址, clock_hour, clock_minute 时
间地址, clock_hour_end, clock_minute_end 结束时间地址
*                  person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
*                  your_number 用户序号地址
* 函数输出          无
*****
*****/

void Play(int *page,int *people,int *age,int *zone,int *weather,int
 *point,int* point_x,int* point_y,int* clock_hour,int *clock_mi
nute,int *clock_hour_end,int *clock_minute_end,people_basic *pers
on_basic,people_competition *person_competition,int* your_number)
{
    int i,j;
    int **direction = (int**)malloc(sizeof(int*)(*people));
    //flag_gui enable process GUI,flag_debug enable debugging fun
ction
    int flag_gui = FALSE,flag_debug = FALSE;

```



```

//create malloc direction[i][j] and init
for(i=0;i<(*people);i++)
{
    direction[i]=(int*)malloc(sizeof(int)*270);
}
for(i=0;i<(*people);i++)
{
    for(j=0;j<270;j++)
    {
        direction[i][j]=0;
    }
}
(*your_number)=random((*people));
//caculate direction
if((*people)!=1)
{
    Direction_Calculate((*people),(*zone),(*weather),(*point)
,point_x,point_y,direction,person_basic,person_competition,(*your
_number),flag_gui,flag_debug);
}
//draw play page
Draw_Play((*age),(*zone));
//msg: choose your way
MsgBox0_Draw0_Play();
while(1)
{
    if(bioskey(0)==ENTER)
    {
        break;
    }
}
Solid_Bar(General_x0,General_y0,General_x1,General_y1,BLACK);
delay(300);
//map: (63,95)-(1024,768)
Map_Display(63,95,(*zone),(*weather),(*point),point_x,point_y
,flag_gui,flag_debug,*clock_hour);
//choose your way
Choose_Your_Way((*zone),(*weather),(*point),point_x,point_y,d
irection[(*your_number)],person_basic,person_competition,(*your_n
umber),flag_gui,flag_debug);
//msg: start play
delay(300);
MsgBox0_Draw1_Play((*your_number)+1);

```

```

while(1)
{
    if(bioskey(0)==ENTER)
    {
        break;
    }
}
Solid_Bar(General_x0,General_y0,General_x1,General_y1,BLACK);
//map: (63,95)-(1024,768)
Map_Display(63,95,(*zone),(*weather),(*point),point_x,point_y
,flag_gui,flag_debug,*clock_hour);
//clock
Clock_Display(929,13,(*clock_hour),(*clock_minute),BLACK);
//animation
Animate(direction,*people,*age,clock_hour,clock_minute,person
_basic,person_competition,*point,point_x,point_y,(*your_number),*
weather);
//free malloc direction[i][j]
for (i=0;i<(*people);i++)
{
    free(direction[i]);
}
free(direction);
//msg: start searching
delay(300);
MsgBox0_Draw2_Play();
while(1)
{
    if(bioskey(0)==ENTER)
    {
        delay(100);
        break;
    }
}
//loading animation
//SetSVGA64k();
Solid_Bar(0,0,1023,767,BLACK);
Loading_Search(512,372,(*people),(*age),(*zone),(*weather),(*
point),clock_hour_end,clock_minute_end,person_basic,person_compet
ition);
//loading 1.6s
(*page)=4;
delay(100);

```

```

}

/*****
*****
* 函数名称      Draw_Play
* 函数作用      结束加载动画, 绘制模拟界面
* 函数输入      age 年龄, zone 区域类型
* 函数输出      无
*****
*****/

void Draw_Play(int age,int zone)
{
    int i,temp=594;
    //loading animation
    for(i=12;i<16;i++)
    {
        delay(100);
        Solid_Bar(temp,445,temp+19,452,LIGHT_GRAY);
        temp=temp+20;
    }
    //refresh
    //SetSVGA64k();
    Solid_Bar(0,0,1023,767,BLACK);
    //upperBar: 1024*50 DARK_GRAY + LIGHT_GRAY
    UpperBar_Draw(50,4,LIGHT_GRAY,DARK_GRAY);
    //boundaryBar: 60-63 DARK_GRAY
    Boundary_Draw(60,53,3,DARK_GRAY);
    //title: center_x is 544
    Title_Draw(age,zone);
}

/*****
*****
* 函数名称      Title_Draw
* 函数作用      绘制比赛标题
* 函数输入      age 年龄, zone 区域类型
* 函数输出      无
*****
*****/

void Title_Draw(int age,int zone)
{
    TitleZone_Draw(zone);
    TitleAge_Draw(age);
}

```

```

        puthz(528,9,32,32,BLACK,"组比赛");
    }

/*****
*****
* 函数名称      TitleZone_Draw
* 函数作用      绘制比赛区域信息
* 函数输入      zone 区域类型
* 函数输出      无
*****
*****/
void TitleZone_Draw(int zone)
{
    switch(zone)
    {
        case 1:
        {
            puthz(400,9,32,32,BLACK,"城区");
            break;
        }
        case 2:
        {
            puthz(400,9,32,32,BLACK,"郊区");
            break;
        }
        case 3:
        {
            puthz(400,9,32,32,BLACK,"山区");
            break;
        }
    }
}

/*****
*****
* 函数名称      TitleAge_Draw
* 函数作用      绘制比赛年龄信息
* 函数输入      age 年龄
* 函数输出      无
*****
*****/
void TitleAge_Draw(int age)
{

```

```

switch(age)
{
    case 1:
        {
            puthz(464,9,32,32,BLACK,"少年");
            break;
        }
    case 2:
        {
            puthz(464,9,32,32,BLACK,"青年");
            break;
        }
    case 3:
        {
            puthz(464,9,32,32,BLACK,"中年");
            break;
        }
    case 4:
        {
            puthz(464,9,32,32,BLACK,"老年");
            break;
        }
}
}

/*****
*****
* 函数名称      Check_Direciton
* 函数作用      检测所选路径方向信息
* 函数输入      last_i, last_j 上一次点击坐标, i, j 本次点击坐标
* 函数输出      0: 路径错误
*               1 2 3 4 : 上, 下, 左, 右
*               5 6 7 8 : 左上, 左下, 右上, 右下
*****
*****/
int Check_Direciton(int last_i,int last_j,int i,int j)
{
    if(i==last_i)
    {
        if(j<last_j)
            return 1;
        else
            return 2;
    }
}

```

```

    }
    else if(j==last_j)
    {
        if(i<last_i)
            return 3;
        else
            return 4;
    }
    else
    {
        if((i-last_i)==(j-last_j)|| (i-last_i)==(last_j-j))
        {
            if(i<last_i)
            {
                if(j<last_j)
                    return 5;
                else
                    return 6;
            }
            else
            {
                if(j<last_j)
                    return 7;
                else
                    return 8;
            }
        }
        else
        {
            //direction error!
            return FALSE;
        }
    }
}
}

```

```

/*****
*****/

```

```

* 函数名称      Check_Distance
* 函数作用      检测所选路径距离信息
* 函数输入      last_i, last_j 上一次点击坐标, i, j 本次点击坐标,
flag_gui 过程可视化开关
* 函数注意      flag_gui 开启过程可视化
* 函数输出      0: 距离错误

```

```

*                                     1: 距离正确
*****
*****/
int Check_Distance(int last_i,int last_j,int i,int j,int flag_gui
)
{
    int ds;
    ds=last_i-i+last_j-j;
    if(flag_gui)
    {
        Solid_Bar(260,9,324,42,LIGHT_GRAY);
        putsz(260,9,32,32,YELLOW,ds);
    }
    if((i==last_i)||(j==last_j))
    {
        if((ds<=10)&&(ds>=-10))
            return TRUE;
        else
            return FALSE;
    }
    else
    {
        if((ds<=12)&&(ds>=-12))
            return TRUE;
        else
            return FALSE;
    }
}

}

/*****
*****
* 函数名称      Check_Availablility
* 函数作用      检测所选路径可行性信息
* 函数输入      last_i, last_j 上一次点击坐标, i, j 本次点击坐标
* 函数输出      0: 可行性错误
*               1: 可行性正确
*****
*****/
int Check_Availablility(int last_i,int last_j,int i,int j)
{
    int x=Min2(last_i,i),y=Min2(last_j,j);
    int xmax=Max2(last_i,i),ymax=Max2(last_j,j);

```

```

    if(last_i==i)                //straight
    {
        for(;y<=ymax;y++)
        {
            if(map_process[x][y]==1)
                return FALSE;
        }
    }
    else if(last_j==j)           //straight
    {
        for(;x<=xmax;x++)
        {
            if(map_process[x][y]==1)
                return FALSE;
        }
    }
    else                          //slash
    {
        if(((i<last_i)&&(j<last_j))||((i>last_i)&&(j>last_j)))
        {
            for(;x<=xmax;x++,y++)
            {
                if(map_process[x][y]==1)
                    return FALSE;
            }
            //right-down scan
        }
        else
        {
            for(;xmax>=x;xmax--,y++)
            {
                if(map_process[xmax][y]==1)
                    return FALSE;
            }
            //right-down scan
        }
    }
    return TRUE;
}

/*****
*****
* 函数名称      Check_Point
* 函数作用      检测所选处点位信息
* 函数输入      i, j 本次点击坐标, last_point 已点击点位数量, point

```


点位数量, point_x 点位x 值地址, point_y 点位y 值地址

* 函数输出 0 非点位
* 1 错误顺序的点位
* 2 正确顺序的点位
* 3 最后一个点位: 出!

*****/

```
int Check_Point(int i,int j,int last_point,int point,int* point_x  
,int *point_y)
```

```
{  
    int cur_point;  
    for(cur_point=1;cur_point<point;cur_point++)  
    {  
        if((i==point_x[cur_point])&&(j==point_y[cur_point]))  
        {  
            if(cur_point!=(last_point+1))  
            {  
                return 1;  
            }  
            else  
            {  
                if(cur_point==(point-1))  
                {  
                    return 3;  
                }  
                else  
                {  
                    return 2;  
                }  
            }  
        }  
    }  
    return 0;  
}
```

/******

* 函数名称 Min2
* 函数作用 获得两个数字最小者
* 函数输入 a, b 两个数字
* 函数输出 最小数

*****/

```

int Min2(int a,int b)
{
    if(a>b)
        return b;
    else
        return a;
}

/*****
*****
* 函数名称      Max2
* 函数作用      获得两个数字最大者
* 函数输入      a, b 两个数字
* 函数输出      最大数
*****
*****/

int Max2(int a,int b)
{
    if(a<b)
        return b;
    else
        return a;
}

/*****
*****
* 函数名称      BlockColor_Change
* 函数作用      改变地图某处的整体颜色
* 函数输入      i, j 改变处在地图上的坐标, color 需要改变的颜色
* 函数输出      无
*****
*****/

void BlockColor_Change(int i,int j,int color)
{
    Solid_Bar(63+24*i,95+24*j,87+24*i,119+24*j,color);
}

/*****
*****
* 函数名称      Choose_Your_Way
* 函数作用      用户选择自己的路径
* 函数输入      zone 区域类型, weather 天气类型, point 点位数量,
point_x 点位x 值地址, point_y 点位y 值地址

```

```

*          direction 用户序号的方向一维数组地址, person_basic
比赛成员基本信息结构体地址
*          person_competition 比赛成员参赛信息结构体地址,
your_number 用户参赛序号
*          flag_gui 过程可视化开关, flag_debug 调试模式开关
* 函数注意      flag_debug 开启调试功能
*          flag_gui 开启过程可视化
* 函数输出      无
*****
*****/
void Choose_Your_Way(int zone,int weather,int point,int *point_x,
int *point_y,int *direction,people_basic *person_basic,people_com
petition *person_competition,int your_number,int flag_gui,int fla
g_debug)
{
    int x,y,i,j,h,t,xmax,ymax,temp;
    int last_i=0,last_j=0,true_last_i=0,true_last_j=0,last_point=
-1,choose_direction,check_point;
    FILE *fp;
    //init point for choose
    for(i=0;i<point;i++)
    {
        BlockColor_Change(point_x[i],point_y[i],BLACK);
        Draw_Choose(63+24*point_x[i],95+24*point_y[i],0);
        putsz(65+24*point_x[i],97+24*point_y[i],16,16,RED,i+1);
        putsz(66+24*point_x[i],97+24*point_y[i],16,16,RED,i+1);
        putsz(65+24*point_x[i],98+24*point_y[i],16,16,RED,i+1);
    }
    //choose your way
    i=0,j=0,temp=0;
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        if(MouseX>62&&MouseX<1024&&MouseY>94&&MouseY<768)
        {
            if(mouse_press(63,95,1023,767)==2)
            {
                shape=3;
                continue;
            }
            else if(mouse_press(63,95,1023,767)==1)
            {

```

```

last_i=i;           //last click position
last_j=j;
i=floor((MouseX-63)/24);
j=floor((MouseY-98)/24);
//get j with 3 offset!!!
if(flag_gui)
{
    Solid_Bar(50,9,178,42,LIGHT_GRAY);
    putsz(50,9,32,32,BLACK,i);
    putsz(114,9,32,32,BLACK,j);
}
if(last_point==-1)    //first point
{
    if((i==0)&&(j==0))
    {
        mousehide(MouseX,MouseY);
        BlockColor_Change(0,0,WHITE);
        Draw_Choose(63+24*i,95+24*j,1);
        resetMouse(MouseX,MouseY);
        last_point++;
        true_last_i=i;           //last true click p
osition

        true_last_j=j;
        delay(210);
        continue;
    }
    else
    {
        delay(100);
        continue;
    }
}
else
{
    if((i==last_i)&&(j==last_j))
    {
        //repetition!
        delay(100);
        continue;
    }
    check_point=Check_Point(i,j,last_point,point,
point_x,point_y);
    choose_direction=Check_Direciton(true_last_i,

```

```

true_last_j,i,j);
    if(flag_gui)
    {
        Solid_Bar(178,9,210,42,LIGHT_GRAY);
        putsz(178,9,32,32,BLUE,check_point);
        Solid_Bar(211,9,259,42,LIGHT_GRAY);
        putsz(211,9,32,32,RED,choose_direction);
    }
    if(!choose_direction)
    {
        //msg: direction error!
        delay(300);
        mousehide(MouseX,MouseY);
        save_image(General_x0,General_y0,General_
x1+3,General_y1+3);
        MsgBox0_Draw0_Notice_Play(3);
        while(1)
        {
            if(bioskey(0)==ENTER)
            {
                break;
            }
        }
        printf_image(General_x0,General_y0,Genera
l_x1+3,General_y1+3);
        delay(210);
        continue;
    }
    else if(!Check_Distance(true_last_i,true_last
_j,i,j,flag_gui))
    {
        //msg: distance error!
        delay(300);
        mousehide(MouseX,MouseY);
        save_image(General_x0,General_y0,General_
x1+3,General_y1+3);
        MsgBox0_Draw0_Notice_Play(2);
        while(1)
        {
            if(bioskey(0)==ENTER)
            {
                break;
            }
        }
    }
}

```

```

    }
    printf_image(General_x0,General_y0,Genera
l_x1+3,General_y1+3);
    delay(210);
    continue;
}
else if(!Check_Availablility(true_last_i,true
_last_j,i,j))
{
    //msg: availablility error!
    delay(300);
    mousehide(MouseX,MouseY);
    save_image(General_x0,General_y0,General_
x1+3,General_y1+3);
    MsgBox0_Draw0_Notice_Play(1);
    while(1)
    {
        if(bioskey(0)==ENTER)
        {
            break;
        }
    }
    printf_image(General_x0,General_y0,Genera
l_x1+3,General_y1+3);
    delay(210);
    continue;
}
else
{
    if(check_point)    //point
    {
        if(check_point==3)
        {
            mousehide(MouseX,MouseY);
            BlockColor_Change(39,27,WHITE);
            Draw_Choose(63+24*i,95+24*j,1);
            resetMouse(MouseX,MouseY);
            x=true_last_i,y=true_last_j;
            xmax=39,ymax=27;
            if(true_last_i==39)    //stra
            ight
            {
                for(y++;y<ymax;y++)

```

```

LOW);
y,choose_direction,Arrow_Color);
;

        {
            mousehide(MouseX,MouseY);
            BlockColor_Change(x,y,YEL

        Draw_Arrow(63+24*x,95+24*

        resetMouse(MouseX,MouseY)

        }
    }
    else if(true_last_j==27)    //str
right
    {
        for(x++;x<xmax;x++)
        {
            mousehide(MouseX,MouseY);
            BlockColor_Change(x,y,YEL

        Draw_Arrow(63+24*x,95+24*

        resetMouse(MouseX,MouseY)

        }
    }
    else    //slash
    {
        //right-
down output: choose_direction must be 8
        for(x++,y++;x<xmax;x++,y++)
        {
            mousehide(MouseX,MouseY);
            BlockColor_Change(x,y,YEL

        Draw_Arrow(63+24*x,95+24*

        resetMouse(MouseX,MouseY)

        }
        //will never be Left-down
    }
    if(choose_direction==8)
    {
        for(y=true_last_j;y<28;y++,te

```

```

mp++)
    {
        direction[temp]=8;
        person_competition[your_n
umber].distance+=90;
    }
}
else if(choose_direction==2)
{
    for(y=true_last_j;y<28;y++,te
mp++)
    {
        direction[temp]=2;
        person_competition[your_n
umber].distance+=60;
    }
}
else
{
    for(x=true_last_i;x<40;x++,te
mp++)
    {
        direction[temp]=4;
        person_competition[your_n
umber].distance+=60;
    }
}
if(flag_gui)
{
    Solid_Bar(2,9,48,48,LIGHT_GRA
Y);
    putsz(2,9,24,24,GREEN,temp);
}
delay(210);
break;
}
else if(check_point==2)
{
    mousehide(MouseX,MouseY);
    BlockColor_Change(i,j,WHITE);
    Draw_Choose(63+24*i,95+24*j,1);
    resetMouse(MouseX,MouseY);
    x=Min2(true_last_i,i),y=Min2(true

```



```

_last_j,j));
xmax=Max2(true_last_i,i),ymax=Max
2(true_last_j,j);
if(true_last_i==i)          //strai
ght
{
    for(y++;y<ymax;y++)
    {
        mousehide(MouseX,MouseY);
        BlockColor_Change(x,y,YEL
LOW);
        Draw_Arrow(63+24*x,95+24*
y,choose_direction,Arrow_Color);
        resetMouse(MouseX,MouseY)
;
    }
}
else if(true_last_j==j)    //stra
ight
{
    for(x++;x<xmax;x++)
    {
        mousehide(MouseX,MouseY);
        BlockColor_Change(x,y,YEL
LOW);
        Draw_Arrow(63+24*x,95+24*
y,choose_direction,Arrow_Color);
        resetMouse(MouseX,MouseY)
;
    }
}
else                        //slash
{
    if(choose_direction==5||choos
e_direction==8)    //right-down output
    {
        for(x++,y++;x<xmax;x++,y+
+)
        {
            mousehide(MouseX,Mous
eY);
            BlockColor_Change(x,y
,YELLOW);

```

```

Draw_Arrow(63+24*x,95
+24*y,choose_direction,Arrow_Color);
resetMouse(MouseX,Mou
seY);

    }
}
else
    //left-down output
    {
        for(xmax=
-,y++;xmax>x;xmax--,y++)
        {
            mousehide(MouseX,Mous
eY);
            BlockColor_Change(xma
x,y,YELLOW);
            Draw_Arrow(63+24*xmax
,95+24*y,choose_direction,Arrow_Color);
            resetMouse(MouseX,Mou
seY);
        }
    }
}
if(true_last_j!=j)
{
    for(y=Min2(true_last_j,j);y<y
max;y++,temp++)
    {
        direction[temp]=choose_di
rection;
        if(choose_direction<5)
        {
            person_competition[yo
ur_number].distance+=60;
        }
        else
        {
            person_competition[yo
ur_number].distance+=90;
        }
    }
}
else

```

```

max;x++,temp++)
{
    xmax=Max2(true_last_i,i);
    for(x=Min2(true_last_i,i);x<x
    {
        direction[temp]=choose_di
        rection;

        if(choose_direction<5)
        {
            person_competition[yo
            ur_number].distance+=60;

        }
        else
        {
            person_competition[yo
            ur_number].distance+=90;

        }
    }
}
if(flag_gui)
{
    Solid_Bar(2,9,48,48,LIGHT_GRA
    Y);

    putsz(2,9,24,24,GREEN,temp);
}
last_point++;
true_last_i=i;
true_last_j=j;
delay(210);
continue;
}
else
{
    //msg: point error!
    delay(300);
    mousehide(MouseX,MouseY);
    save_image(General_x0,General_y0,
    General_x1+3,General_y1+3);

    MsgBox0_Draw0_Notice_Play(4);
    while(1)
    {
        if(bioskey(0)==ENTER)
        {

```

```

                                break;
                            }
                        }
                        printf_image(General_x0,General_y
0,General_x1+3,General_y1+3);
                        delay(210);
                        continue;
                    }
                }
                else    //non-point
                {
                    mousehide(MouseX,MouseY);
                    BlockColor_Change(i,j,BLUE);
                    Draw_Choose(63+24*i,95+24*j,1);
                    resetMouse(MouseX,MouseY);

                    x=Min2(true_last_i,i),y=Min2(true_las
t_j,j);
                    xmax=Max2(true_last_i,i),ymax=Max2(tr
ue_last_j,j);

                    if(true_last_i==i)    //straight
                    {
                        for(y++;y<ymax;y++)
                        {
                            mousehide(MouseX,MouseY);
                            BlockColor_Change(x,y,YELLOW)
;
                            Draw_Arrow(63+24*x,95+24*y,ch
oose_direction,Arrow_Color);
                            resetMouse(MouseX,MouseY);
                        }
                    }
                    else if(true_last_j==j)    //straight
                    {
                        for(x++;x<xmax;x++)
                        {
                            mousehide(MouseX,MouseY);
                            BlockColor_Change(x,y,YELLOW)
;
                            Draw_Arrow(63+24*x,95+24*y,ch
oose_direction,Arrow_Color);
                            resetMouse(MouseX,MouseY);
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    else //slash
    {
        if(choose_direction==5||choose_di
rection==8) //right-down output
        {
            for(x++,y++;x<xmax;x++,y++)
            {
                mousehide(MouseX,MouseY);
                BlockColor_Change(x,y,YEL
LOW);
                Draw_Arrow(63+24*x,95+24*
y,choose_direction,Arrow_Color);
                resetMouse(MouseX,MouseY)
;
            }
        }
        else
        //left-down output
        {
            for(xmax--,y++;xmax>x;xmax-
-,y++)
            {
                mousehide(MouseX,MouseY);
                BlockColor_Change(xmax,y,
YELLOW);
                Draw_Arrow(63+24*xmax,95+
24*y,choose_direction,Arrow_Color);
                resetMouse(MouseX,MouseY)
;
            }
        }
    }
    if(true_last_j!=j)
    {
        for(y=Min2(true_last_j,j);y<ymax;
y++,temp++)
        {
            direction[temp]=choose_direct
ion;

            if(choose_direction<5)
            {
                person_competition[your_n

```



```

        //reset mouse
        if(shape!=0)
        {
            shape=0;
        }
    }
    //analysis direction
    if(zone!=3)
    {
        t=2*person_basic[your_number].ps_consume/person_basic[your_number].ps_recover;
        j=0;
        while(direction[j]!=0)
        {
            if(j%t<(t/2))
            {
                direction[j]+=20;
            }
            else
            {
                direction[j]+=10;
            }
            j++;
        }
    }
    else
    {
        t=2*person_basic[your_number].ps_consume/person_basic[your_number].ps_recover;
        x=0,y=0,j=0,h=0;
        if( map_display[0][0]>10)
        {
            while(direction[j]!=0)
            {
                if(direction[j]==1)
                {
                    y--;
                }
                else if(direction[j]==2)
                {
                    y++;
                }
                else if(direction[j]==3)

```

```

    {
        x--;
    }
    else if(direction[j]==4)
    {
        x++;
    }
    else if(direction[j]==5)
    {
        x--;
        y--;
    }
    else if(direction[j]==6)
    {
        x--;
        y++;
    }
    else if(direction[j]==7)
    {
        x++;
        y--;
    }
    else if(direction[j]==8)
    {
        x++;
        y++;
    }
    if(map_display[x][y]<10)
    {
        break;
    }
    direction[j]+=30;
    j++;
}
h=0;
while(direction[j]!=0)
{
    if(h%t<(t/2))
    {
        direction[j]+=20;
    }
    else
    {

```



```

        direction[j]+=10;
    }
    j++,h++;
}
}
else
{
    while(direction[j]!=0)
    {
        if(direction[j]==1)
        {
            y--;
        }
        else if(direction[j]==2)
        {
            y++;
        }
        else if(direction[j]==3)
        {
            x--;
        }
        else if(direction[j]==4)
        {
            x++;
        }
        else if(direction[j]==5)
        {
            x--;
            y--;
        }
        else if(direction[j]==6)
        {
            x--;
            y++;
        }
        else if(direction[j]==7)
        {
            x++;
            y--;
        }
        else if(direction[j]==8)
        {
            x++;

```

```

        y++;
    }
    if(map_display[x][y]>10)
    {
        break;
    }
    if(j%t<(t/2))
    {
        direction[j]+=20;
    }
    else
    {
        direction[j]+=10;
    }
    j++;
}
while(direction[j]!=0)
{
    direction[j]+=30;
    j++;
}
}
}
if(flag_gui)
{
    Solid_Bar(2,9,324,42,LIGHT_GRAY);
}
if(flag_debug)
{
    fp = fopen (".\\WAY\\YOURWAY.txt","w+");
    fprintf(fp,"Your Way Information\n");
    fprintf(fp,"\nYour Number = %d\n",your_number+1);
    fprintf(fp,"\nReference\n1:up 2:down 3:left 4:right\n5
:left-up 6:left-down 7:right-up 8:right-down\n");
    i=0,x=0,y=0;
    while(direction[i]!=0)
    {
        fprintf(fp,"direction[%d]=%d;\n",i,direction[i]);
        if(direction[i]==1)
        {
            y--;
        }
        else if(direction[i]==2)

```

```

        {
            y++;
        }
        else if(direction[i]==3)
        {
            x--;
        }
        else if(direction[i]==4)
        {
            x++;
        }
        else if(direction[i]==5)
        {
            x--;
            y--;
        }
        else if(direction[i]==6)
        {
            x--;
            y++;
        }
        else if(direction[i]==7)
        {
            x++;
            y--;
        }
        else if(direction[i]==8)
        {
            x++;
            y++;
        }
        i++;
    }
    fclose(fp);
}
}

```

29. quit.c

```
#include"headfile.h"
```

```

/*****
*****
* @author          ytm
* @date            2022-4-4

```

```

*****
*****/

//driver
#define General_x0 344
#define General_y0 230
#define General_x1 680
#define General_y1 610
#define Text_xplus 130
#define Text_yplus 25

//follower
#define Message_x0 (General_x0+8)
#define Message_y0 (General_y0+19)
#define Message_x1 (General_x0+326)
#define Message_y1 (General_y0+191)
#define Confirm_x0 (General_x0+8)
#define Confirm_y0 (General_y0+197)
#define Confirm_x1 (General_x0+326)
#define Confirm_y1 (General_y0+282)
#define Cancel_x0 (General_x0+8)
#define Cancel_y0 (General_y0+286)
#define Cancel_x1 (General_x0+326)
#define Cancel_y1 (General_y0+371)

/*****
*****
* 函数名称      Quit
* 函数作用      退出界面
* 函数输入      page 界面地址
* 函数输出      无
*****
*****/

void Quit(int *page)
{
    int flag=0;
    Button_Darken_Menu(1);
    Button_Darken_Menu(2);
    Button_Darken_Menu(3);
    Draw_Quit();
    resetMouse(MouseX,MouseY);
    while(1)
    {

```

```

        newxy(&MouseX,&MouseY,&press);
        if(MouseX>Confirm_x0&&MouseX<Confirm_x1&& MouseY>Confirm_
y0&&MouseY<Confirm_y1)
        {
            if(mouse_press(Confirm_x0,Confirm_y0,Confirm_x1,Confir
rm_y1)==2)
            {
                shape=3;
                if(flag==0);
                {
                    mousehide(MouseX,MouseY);
                    Button_Light_Quit(1);
                    resetMouse(MouseX,MouseY);
                    flag=1;
                }
                continue;
            }
            else if(mouse_press(Confirm_x0,Confirm_y0,Confirm_x1,
Confirm_y1)==1)
            {
                delay(100);
                exit(0);
            }
        }
        if(MouseX>Cancel_x0&&MouseX<Cancel_x1&& MouseY>Cancel_y0&
&MouseY<Cancel_y1)
        {
            if(mouse_press(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y
1)==2)
            {
                shape=3;
                if(flag==0);
                {
                    mousehide(MouseX,MouseY);
                    Button_Light_Quit(2);
                    resetMouse(MouseX,MouseY);
                    flag=2;
                }
                continue;
            }
            else if(mouse_press(Cancel_x0,Cancel_y0,Cancel_x1,Can
cel_y1)==1)
            {

```

```

        *page=0;
        getMousebk(MouseX,MouseY);
        delay(100);
        //go to menu page
        break;
    }
}
if(flag!=0)
{
    mousehide(MouseX,MouseY);
    Button_Darken_Quit(flag);
    resetMouse(MouseX,MouseY);
    flag=0;
}
if(shape!=0)
{
    shape=0;
}
}
}

/*****
*****
* 函数名称      Draw_Quit
* 函数作用      绘制退出界面
* 函数输入      无
* 函数输出      无
*****
*****/
void Draw_Quit()
{
    //general button: 336*380 DARK_GRAY + GRAY
    Button_Draw(General_x0,General_y0,General_x1,General_y1,DARK_
GRAY,GRAY);

    //message box: 328*172 DARK_GRAY + BLACK
    Button_Draw(Message_x0,Message_y0,Message_x1,Message_y1,Hollo
w_Color,BLACK);
    puthz(Message_x0+14,Message_y0+14,32,32,WHITE,"确定退出软件?
");

    //buttons: 328*86 DARK_GRAY + LIGHT_GRAY
    Button_Draw(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,Hollo

```

```

w_Color,Solid_Color);
    puthz(Confirm_x0+Text_xplus,Confirm_y0+Text_yplus,32,32,BLACK
,"确定");

    Button_Draw(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,Hollow_Color,Solid_Color);
    puthz(Cancel_x0+Text_xplus,Cancel_y0+Text_yplus,32,32,BLACK,"
取消");
}

/*****
*****
* 函数名称      Button_Light_Quit
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Light_Quit(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,WHITE);
            Solid_Bar(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,GREEN);
            puthz(Confirm_x0+Text_xplus,Confirm_y0+Text_yplus,32,32,GRAY,"确定");
            break;
        }
        case 2:
        {
            Button_Edge(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,WHITE);
            Solid_Bar(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,RED);
            puthz(Cancel_x0+Text_xplus,Cancel_y0+Text_yplus,32,32,GRAY,"取消");
            break;
        }
    }
}

```

```

}

/*****
*****
* 函数名称      Button_Darken_Quit
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Darken_Quit(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,GRAY);
            Button_Draw(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,Hollow_Color,Solid_Color);
            puthz(Confirm_x0+Text_xplus,Confirm_y0+Text_yplus,32,
32,BLACK,"确定");
            break;
        }
        case 2:
        {
            Button_Edge(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,G
RAY);
            Button_Draw(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,H
ollow_Color,Solid_Color);
            puthz(Cancel_x0+Text_xplus,Cancel_y0+Text_yplus,32,32
,BLACK,"取消");
            break;
        }
    }
}

```

30. register.c

```

#include "headfile.h"

#define Button_x_left 256
#define Button_x_right 767
#define Button_y_up 192-8-32

```



```

#define Buttom_y_down 575+8+32
#define Message_x_left (Buttom_x_left+8)
#define Message_x_right (Buttom_x_right-8)
#define Message_y_up (Buttom_y_up+16+64+16)
#define Message_y_down Message_y_up+8+32+8+32+8+32+8+32+16+8+32+8
+32

#define register_x0 Buttom_x_left+8
#define register_y0 Message_y_down+8
#define register_x1 512-4
#define register_y1 Buttom_y_down-8
#define out_x0 512+4
#define out_y0 Message_y_down+8
#define out_x1 Buttom_x_right-8
#define out_y1 Buttom_y_down-8
#define loginword_x0 Buttom_x_left+100
#define loginword_y0 Message_y_down+38
#define outword_x0 608
#define outword_y0 Message_y_down+8+30
#define input1_x0 Message_x_left+8+8
#define input1_y0 Message_y_up+8+32+8
#define input1_x1 Message_x_right-8-8
#define input1_y1 Message_y_up+8+32+8+32
#define input2_x0 Message_x_left+8+8
#define input2_y0 Message_y_up+8+32+8+32+8+32+8
#define input2_x1 Message_x_right-8-8
#define input2_y1 Message_y_up+8+32+8+32+8+32+8+32
#define input3_x0 Message_x_left+8+8
#define input3_y0 Message_y_up+8+32+8+32+8+32+8+32+8+32+8
#define input3_x1 Message_x_right-8-8
#define input3_y1 Message_y_up+8+32+8+32+8+32+8+32+8+32+8+32

#define msg_x0 312
#define msg_y0 309
#define msg_x1 712
#define msg_y1 459

/*
#define register_x0 Buttom_x_right-8-100
#define register_y0 Buttom_y_up+8
#define register_x1 Buttom_x_right-8
#define register_y1 Message_y_up-8
*/

```

```
//#define regiword_x0 Buttom_x_right-8-100+50-32
//#define regiword_y0 Buttom_y_up+32

/*****
*****

* 函数名称      Register
* 函数作用      注册界面
* 函数输入      page 显存页
* 函数输出      无
*****
*****/
```

```
void Register(int *page)
{
    int flag=0;
    char user_buffer[11]={0},pass_buffer[11]={0},confirm_buffer[1
1]={0};
    int usernum;
    int* p=&usernum;
    FILE* fp;
    if((fp=fopen(".\\DATA\\ALL.DAT", "rb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    fseek(fp,2L,SEEK_SET);
    fread(p,sizeof(int),1,fp);
    fclose(fp);
    Draw_Register();
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        if(MouseX>register_x0&&MouseX<register_x1&& MouseY>regist
er_y0&&MouseY<register_y1)
        {
            if(mouse_press(register_x0,register_y0,register_x1,re
gister_y1)==2)
            {
                shape=3;
                if(flag==0);
                {
                    mousehide(MouseX,MouseY);
                    Button_Light_Register(1);
```

```

        resetMouse(MouseX,MouseY);
        flag=1;
    }
    continue;
}
else if(mouse_press(register_x0,register_y0,register_
x1,register_y1)==1)
{
    //New_User(user_buffer,pass_buffer);
    if(Scan_Nullinput(user_buffer)==1)
    {
        Uni_Msgbox_1();
        Draw_Register();
        continue;
    }
    else if(Scan_Nullinput(pass_buffer)==1)
    {
        Uni_Msgbox_2();
        Draw_Register();
        continue;
    }
    else if(Scan_Nullinput(confirm_buffer)==1)
    {
        Uni_Msgbox_3();
        Draw_Register();
        continue;
    }
    else if(Scan_reRegister(usernum,user_buffer)==1)
    {
        Uni_Msgbox_4();
        Draw_Register();
        continue;
    }
    else if(Scan_Confirm(pass_buffer,confirm_buffer)=
=0)
    {
        Uni_Msgbox_5();
        Draw_Register();
        continue;
    }
    else
    {
        Uni_Msgbox_6();

```

```

        New_User(user_buffer,pass_buffer);
        Darken_Register();
        *page=10;
        getMousebk(MouseX,MouseY);
        delay(100);
        //go to menu page
        break;
        delay(100);
    }
}
}
if(MouseX>out_x0&&MouseX<out_x1&& MouseY>out_y0&&MouseY<out_y1)
{
    if(mouse_press(out_x0,out_y0,out_x1,out_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Register(2);
            resetMouse(MouseX,MouseY);
            flag=2;
        }
        continue;
    }
    else if(mouse_press(out_x0,out_y0,out_x1,out_y1)==1)
    {
        Darken_Register();
        *page=10;
        getMousebk(MouseX,MouseY);
        delay(100);
        //go to menu page
        break;
    }
}
if(MouseX>input1_x0&&MouseX<input1_x1&& MouseY>input1_y0&&MouseY<input1_y1)
{
    if(mouse_press(input1_x0,input1_y0,input1_x1,input1_y1)==2)
    {
        shape=2;

```

```

        continue;
    }
    else if(mouse_press(input1_x0,input1_y0,input1_x1,inp
ut1_y1)==1)
    {
        Button_Draw(Message_x_left+8+8,Message_y_up+8+32+
8,Message_x_right-8-8,Message_y_up+8+32+8+32,DARK_GRAY,GRAY);
        Reg_Input_Username(user_buffer);
        resetMouse(MouseX,MouseY);
        delay(100);
        //go to menu page
    }
}
if(MouseX>input2_x0&&MouseX<input2_x1&& MouseY>input2_y0&
&MouseY<input2_y1)
{
    if(mouse_press(input2_x0,input2_y0,input2_x1,input2_y
1)==2)
    {
        shape=2;
        continue;
    }
    else if(mouse_press(input2_x0,input2_y0,input2_x1,inp
ut2_y1)==1)
    {
        Button_Draw(input2_x0,input2_y0,input2_x1,input2_
y1,DARK_GRAY,GRAY);
        Reg_Input_Password(pass_buffer);
        resetMouse(MouseX,MouseY);
        delay(100);
        //go to menu page
    }
}
if(MouseX>input3_x0&&MouseX<input3_x1&& MouseY>input3_y0&
&MouseY<input3_y1)
{
    if(mouse_press(input3_x0,input3_y0,input3_x1,input3_y
1)==2)
    {
        shape=2;
        continue;
    }
    else if(mouse_press(input3_x0,input3_y0,input3_x1,inp

```

```

ut3_y1)==1)
    {
        Button_Draw(input3_x0,input3_y0,input3_x1,input3_
y1,DARK_GRAY,GRAY);
        Reg_Confirm_Password(confirm_buffer);
        resetMouse(MouseX,MouseY);
        delay(100);
        //go to menu page
    }
}
if(flag!=0)
{
    mousehide(MouseX,MouseY);
    Button_Darken_Register(flag);
    resetMouse(MouseX,MouseY);
    flag=0;
}
if(shape!=0)
{
    shape=0;
}
}
}

```

```

/*****
*****
* 函数名称      Register_Check
* 函数作用      注册检测
* 函数输入      page 显存页
* 函数输出      无
*****
*****/

```

```

void Register_Check(int *page)
{
    //    puthz(32,767-40,32,32,WHITE,"调试模式: 按返回进入主界面");
    Register(&page);
    //Input_Username();
    Solid_Bar(0,0,1023,767,BLACK);
}

```

```

/*****
*****

```

```

* 函数名称      New_User
* 函数作用      新用户生成
* 函数输入      username 用户名, password 密码
* 函数输出      无

```

```

*****
*****/

```

```

void New_User(char* username,char* password)
{
    int usernum=2;
    int* p=&usernum;
    FILE* fp;
    if((fp=fopen(".\\DATA\\USER.DAT","ab"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    fwrite(username,11,1,fp);
    fwrite(password,11,1,fp);
    fclose(fp);
    if((fp=fopen(".\\DATA\\ALL.DAT","rb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    fseek(fp,2L,SEEK_SET);
    fread(p,sizeof(int),1,fp);
    usernum++;
    fclose(fp);
    if((fp=fopen(".\\DATA\\ALL.DAT","wb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    fseek(fp,2L,SEEK_SET);
    fwrite(p,sizeof(int),1,fp);
    fclose(fp);
}

```

```

/*****
*****

```

```

* 函数名称      Scan_reRegister
* 函数作用      扫描是否用户名重复
* 函数输入      usernum 当前用户数, user_buffer 用户名
* 函数输出      是返回1, 否返回0

```

```

*****

```

```

*****/

int Scan_reRegister(int usernum,char* user_buffer)
{
    char user_lib[11]={0};
    FILE* fp;
    int i;
    if((fp=fopen(".\\DATA\\USER.DAT","rb"))==NULL)
    {
        puthz(0,0,32,32,WHITE,"FILE ERROR!");
    }
    for(i=0;i<usernum;i++)
    {
        fseek(fp,(long)(22*i),SEEK_SET);
        fread(user_lib,11,1,fp);
        if(strcmp(user_buffer,user_lib)==0)
        {
            fclose(fp);
            return 1;
        }
    }
    fclose(fp);
    return 0;
}

/*****
*****
* 函数名称      Scan_Confirm
* 函数作用      扫描确认密码是否一致
* 函数输入      pass_buffer 密码,  confirm_buffer 确认密码
* 函数输出      是返回1, 否返回0
*****
*****/

int Scan_Confirm(char* pass_buffer,char* confirm_buffer)
{
    if(strcmp(pass_buffer,confirm_buffer)==0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

```



```

    }
}

/*****
*****
* 函数名称      Scan_Nullinput
* 函数作用      扫描输入框为空
* 函数输入      buffer 字符串
* 函数输出      是返回1，否返回0
*****
*****/

```

```

int Scan_Nullinput(char* buffer)
{
    char* lib[11]={0};
    if(strcmp(buffer,lib)==0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

```

31. search.c

```

#include"headfile.h"

/*****
*****
* @author      ytm
* @date        2022-4-16
*****
*****/

//driver
#define Revert_x0 5
#define Revert_y0 5

#define General_x0 344
#define General_y0 230
#define General_x1 680
#define General_y1 610

```

```

#define Text1_x0          12
#define Text1_y0          70
#define Text_xplus        96
#define Text_yplus        24

//follower
#define Revert_x1  (Revert_x0+39)
#define Revert_y1  (Revert_y0+39)

#define Confirm_x0 (General_x0+8)
#define Confirm_y0 (General_y0+197)
#define Confirm_x1 (General_x0+326)
#define Confirm_y1 (General_y0+282)
#define Cancel_x0  (General_x0+8)
#define Cancel_y0  (General_y0+286)
#define Cancel_x1  (General_x0+326)
#define Cancel_y1  (General_y0+371)

#define UserCompetition_x0  12
#define UserCompetition_y0  (Text1_y0+46)
#define UserCompetition_x1  (Text1_x0+383)
#define UserCompetition_y1  (Text1_y0+132)

#define Text2_x0          12
#define Text2_y0          (Text1_y0+146)

#define GenCompetition_x0  12
#define GenCompetition_y0  (Text1_y0+192)
#define GenCompetition_x1  (Text1_x0+383)
#define GenCompetition_y1  (Text1_y0+271)
#define PerCompetition_x0  12
#define PerCompetition_y0  (Text1_y0+284)
#define PerCompetition_x1  (Text1_x0+383)
#define PerCompetition_y1  (Text1_y0+364)

//driver
#define TextPer_xplus      154
#define TextPer_yplus      8

//follower
#define Num_x0              PerCompetition_x0+2
#define Num_y0              (PerCompetition_y0+2)
#define Num_x1              PerCompetition_x1-2

```

```

#define Num_y1          (PerCompetition_y0+41)
#define Rank_x0          PerCompetition_x0+2
#define Rank_y0          (PerCompetition_y0+42)
#define Rank_x1          PerCompetition_x1-2
#define Rank_y1          (PerCompetition_y0+81)

//driver
#define InputButton_x0    429
#define InputButton_y0    70
#define InputText_xplus   62
#define InputText_yplus   24

//follower
#define InputButton_x1    (InputButton_x0+381)
#define InputButton_y1    (InputButton_y0+79)
#define InputConfirm_x0   (InputButton_x0+387)
#define InputConfirm_y0   70
#define InputConfirm_x1   (InputButton_x0+575)
#define InputConfirm_y1   (InputButton_y0+79)

//driver
#define gentitle_x        543  //word 11, size 32

#define geny_first        70
#define geny_second       120
#define geny_third        161
#define geny_fourth       202
#define geny_fifth        243
#define geny_sixth        284
#define geny_seventh      325
#define geny_eighth       366
#define geny_ninth        407
#define geny_tenth        448  //every row +41

#define pertitle_x        543  //word 11, size 32
#define pery_first        165
#define pery_second       215
#define pery_third        256
#define pery_fourth       297
#define pery_fifth        338
#define pery_sixth        379  //every row +41

#define genperline12_x    610  //for ':'

```

```

#define genperline1_word1_x 512
#define genperline1_word2_x 501
#define genperline1_word4_x 477
#define genperline2_word1_x 706
#define genperline2_word2_x 694
#define genperline2_word4_x 670
#define genperline2_word5_x 678 //part 16
#define genperline3_word2_x 887
#define genperline3_word5_x 871 //part 16

/*****
*****
* 函数名称      Draw_Search
* 函数作用      绘制搜索界面
* 函数输入      无
* 函数输出      无
*****
*****/
void Draw_Search()
{
    //upperBar: 1024*50 DARK_GRAY + LIGHT_GRAY
    UpperBar_Draw(50,4,LIGHT_GRAY,DARK_GRAY);

    //BoundaryBar: 410-413 DARK_GRAY
    Boundary_Draw(410,53,4,DARK_GRAY);

    //button 1: 40*40 BLACK + LIGHT_GRAY
    RevertImage_Draw(25,25,BLACK);
    puthz(50,9,32,32,BLACK,"战况");

    //button 4: 384*86 DARK_GRAY + LIGHT_GRAY
    puthz(Text1_x0,Text1_y0,32,32,WHITE,"用户成绩查询");

    Button_Draw(UserCompetition_x0,UserCompetition_y0,UserCompetition_x1,UserCompetition_y1,Hollow_Color,Solid_Color);
    puthz(UserCompetition_x0+Text_xplus,UserCompetition_y0+Text_yplus,32,32,BLACK,"用户成绩展示");

    //button 2-3: 384*86 DARK_GRAY + LIGHT_GRAY
    puthz(Text2_x0,Text2_y0,32,32,WHITE,"比赛成绩查询");

    Button_Draw(GenCompetition_x0,GenCompetition_y0,GenCompetition_x1,GenCompetition_y1,Hollow_Color,Solid_Color);

```

```
    puthz(GenCompetition_x0+Text_xplus,GenCompetition_y0+Text_yplus,32,32,BLACK,"整体战况展示");
```

```
    Button_Draw(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1,Hollow_Color,Solid_Color);
```

```
    puthz(PerCompetition_x0+Text_xplus,PerCompetition_y0+Text_yplus,32,32,BLACK,"选手成绩查询");
```

```
    TriangleImage(PerCompetition_x0+311,PerCompetition_y0+31,DARK_GRAY);
}
```

```

/*****
*****/

```

```
* 函数名称      MsgBox2Button_Light_Search
```

```
* 函数作用      点亮按钮
```

```
* 函数输入      flag 按钮序号
```

```
* 函数输出      无
```

```

*****/
*****/

```

```
void MsgBox2Button_Light_Search(int flag)
```

```
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,WHITE);
            Solid_Bar(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1,SKY_BLUE);
            puthz(Confirm_x0+130,Confirm_y0+25,32,32,GRAY,"确定");
            break;
        }
        case 2:
        {
            Button_Edge(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,WHITE);
            Solid_Bar(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,RED);
            puthz(Cancel_x0+130,Cancel_y0+25,32,32,GRAY,"取消");
            break;
        }
    }
}
```

```

}

/*****
*****
* 函数名称      MsgBox2Button_Darken_Search
* 函数作用      熄灭亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

void MsgBox2Button_Darken_Search(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,GRAY);
            Button_Draw(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_
y1,Hollow_Color,Solid_Color);
            puthz(Confirm_x0+130,Confirm_y0+25,32,32,BLACK,"确定
");
            break;
        }
        case 2:
        {
            Button_Edge(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,G
RAY);
            Button_Draw(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1,H
ollow_Color,Solid_Color);
            puthz(Cancel_x0+130,Cancel_y0+25,32,32,BLACK,"取消");
            break;
        }
    }
}

/*****
*****
* 函数名称      Button_Light_Search
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

*****/
void Button_Light_Search(int flag)
{
    switch(flag)
    {
        case 1:
        {
            RevertButton_Draw(25,25,DARK_GRAY);
            RevertImage_Draw(25,25,WHITE);
            //revert button on
            break;
        }
        case 2:
        {
            Button_Edge(GenCompetition_x0,GenCompetition_y0,GenCompetition_x1,GenCompetition_y1,WHITE);
            Solid_Bar(GenCompetition_x0,GenCompetition_y0,GenCompetition_x1,GenCompetition_y1,SKY_BLUE);
            puthz(GenCompetition_x0+Text_xplus,GenCompetition_y0+Text_yplus,32,32,GRAY,"整体战况展示");
            break;
        }
        case 3:
        {
            Button_Edge(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1,WHITE);
            Solid_Bar(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1,SKY_BLUE);
            puthz(PerCompetition_x0+Text_xplus,PerCompetition_y0+Text_yplus,32,32,GRAY,"选手成绩查询");
            TriangleImage(PerCompetition_x0+311,PerCompetition_y0+31,DARK_GRAY);
            break;
        }
        case 4:
        {
            Button_Edge(UserCompetition_x0,UserCompetition_y0,UserCompetition_x1,UserCompetition_y1,WHITE);
            Solid_Bar(UserCompetition_x0,UserCompetition_y0,UserCompetition_x1,UserCompetition_y1,SKY_BLUE);
            puthz(UserCompetition_x0+Text_xplus,UserCompetition_y0+Text_yplus,32,32,GRAY,"用户成绩展示");
            break;
        }
    }
}

```

```

    }
}

}

/*****
*****
* 函数名称      Button_Darken_Search
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void Button_Darken_Search(int flag)
{
    switch(flag)
    {
        case 1:
        {
            RevertButton_Draw(25,25,LIGHT_GRAY);
            RevertImage_Draw(25,25,BLACK);
            //revert button off
            break;
        }
        case 2:
        {
            Button_Edge(GenCompetition_x0,GenCompetition_y0,GenCompetition_x1,GenCompetition_y1,BLACK);
            Button_Draw(GenCompetition_x0,GenCompetition_y0,GenCompetition_x1,GenCompetition_y1,Hollow_Color,Solid_Color);
            puthz(GenCompetition_x0+Text_xplus,GenCompetition_y0+Text_yplus,32,32,BLACK,"整体战况展示");
            break;
        }
        case 3:
        {
            Button_Edge(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1,BLACK);
            Button_Draw(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1,Hollow_Color,Solid_Color);
            puthz(PerCompetition_x0+Text_xplus,PerCompetition_y0+Text_yplus,32,32,BLACK,"选手成绩查询");
            TriangleImage(PerCompetition_x0+311,PerCompetition_y0+31,DARK_GRAY);

```



```

        break;
    }
    case 4:
    {
        Button_Edge(UserCompetition_x0,UserCompetition_y0
,UserCompetition_x1,UserCompetition_y1,BLACK);
        Button_Draw(UserCompetition_x0,UserCompetition_y0
,UserCompetition_x1,UserCompetition_y1,Hollow_Color,Solid_Color);
        puthz(UserCompetition_x0+Text_xplus,UserCompetiti
on_y0+Text_yplus,32,32,BLACK,"用户成绩展示");
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      Button_Press_Search
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void Button_Press_Search(int flag)
{
    Solid_Bar(623,9,815,41,LIGHT_GRAY);
    switch(flag)
    {
        case 2:
        {
            Button_Edge(GenCompetition_x0,GenCompetition_y0,G
enCompetition_x1,GenCompetition_y1,BLACK);
            Button_Draw(GenCompetition_x0,GenCompetition_y0,G
enCompetition_x1,GenCompetition_y1,LIGHT_GRAY,GRAY);
            puthz(GenCompetition_x0+Text_xplus,GenCompetition
_y0+Text_yplus,32,32,BLACK,"整体战况展示");
            puthz(623,9,32,32,BLACK,"整体战况展示");
            break;
        }
        case 3:
        {
            Button_Edge(PerCompetition_x0,PerCompetition_y0,P
erCompetition_x1,PerCompetition_y1,BLACK);

```

```

        Button_Draw(PerCompetition_x0,PerCompetition_y0,P
erCompetition_x1,PerCompetition_y1,LIGHT_GRAY,GRAY);
        puthz(PerCompetition_x0+Text_xplus,PerCompetition
_y0+Text_yplus,32,32,BLACK,"选手成绩查询");
        TriangleImage(PerCompetition_x0+311,PerCompetition_y0+31,DARK_GRAY);
        puthz(623,9,32,32,BLACK,"选手成绩查询");
        break;
    }
    case 4:
    {
        Button_Edge(UserCompetition_x0,UserCompetition_y0
,UserCompetition_x1,UserCompetition_y1,BLACK);
        Button_Draw(UserCompetition_x0,UserCompetition_y0
,UserCompetition_x1,UserCompetition_y1,LIGHT_GRAY,GRAY);
        puthz(UserCompetition_x0+Text_xplus,UserCompetition_y0+Text_yplus,32,32,BLACK,"用户成绩展示");
        puthz(623,9,32,32,BLACK,"用户成绩展示");
        break;
    }
}
}

```

```

/*****
*****/

```

```

* 函数名称      Message_Title_Draw
* 函数作用      绘制标题
* 函数输入      x0, y0 绘制左上角位置
* 函数输出      无

```

```

*****/

```

```

void Message_Title_Draw(int x0,int y0,int age,int zone)
{
    if(age==1)
    {
        if(zone==1)
        {
            puthz(x0,y0,32,32,WHITE,"城区少年组定向越野比赛");
        }
        else if(zone==2)
        {
            puthz(x0,y0,32,32,WHITE,"郊区少年组定向越野比赛");
        }
    }
}

```

```

        else
        {
            puthz(x0,y0,32,32,WHITE,"山区少年组定向越野比赛");
        }
    }
    else if(age==2)
    {
        if(zone==1)
        {
            puthz(x0,y0,32,32,WHITE,"城区青年组定向越野比赛");
        }
        else if(zone==2)
        {
            puthz(x0,y0,32,32,WHITE,"郊区青年组定向越野比赛");
        }
        else
        {
            puthz(x0,y0,32,32,WHITE,"山区青年组定向越野比赛");
        }
    }
    else if(age==3)
    {
        if(zone==1)
        {
            puthz(x0,y0,32,32,WHITE,"城区中年组定向越野比赛");
        }
        else if(zone==2)
        {
            puthz(x0,y0,32,32,WHITE,"郊区中年组定向越野比赛");
        }
        else
        {
            puthz(x0,y0,32,32,WHITE,"山区中年组定向越野比赛");
        }
    }
    else
    {
        if(zone==1)
        {
            puthz(x0,y0,32,32,WHITE,"城区老年组定向越野比赛");
        }
        else if(zone==2)
        {

```

```

        puthz(x0,y0,32,32,WHITE,"郊区老年组定向越野比赛");
    }
    else
    {
        puthz(x0,y0,32,32,WHITE,"山区老年组定向越野比赛");
    }
}

}

/*****
*****
* 函数名称      UserMessage_Light_Search
* 函数作用      显示用户信息
* 函数输入      people 人数, age 年龄, zone 区域类型, point 点位数量
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
*               your_number 用户参赛序号, your_name 用户名字
* 函数输出      无
*****
*****/
void UserMessage_Light_Search(int people,int age,int zone,int poi
nt,people_basic *person_basic,people_competition *person_competit
ion,int your_number,char* your_name)
{
    int i,y;
    double verb;
    //refresh right whole page
    Solid_Bar(414,54,1023,767,BLACK);
    //display information
    Message_Title_Draw(gentitle_x,geny_first,age,zone);
    puthz(genperline1_word4_x,geny_second,24,24,WHITE,"您的姓名
");
    puthz(genperline12_x+3,geny_second,24,16,WHITE,"：");
    putzm(genperline2_word5_x,geny_second,24,16,WHITE,your_name);
    puthz(genperline1_word4_x,geny_third,24,24,WHITE,"您的成绩");
    puthz(genperline12_x,geny_third,24,24,WHITE,"：");
    Clock_Display(genperline2_word5_x,geny_third,person_competiti
on[your_number].hour_score,person_competition[your_number].minute
_score,WHITE);
    puthz(genperline1_word4_x,geny_fourth,24,24,WHITE,"开始时间
");
    puthz(genperline12_x,geny_fourth,24,24,WHITE,"：");
    Clock_Display(genperline2_word5_x,geny_fourth,person_competit

```

```

ion[your_number].hour_point[0],person_competition[your_number].minute_point[0],WHITE);
    puthz(genperline1_word4_x,geny_fifth,24,24,WHITE,"完成时间");
    puthz(genperline12_x,geny_fifth,24,24,WHITE," : ");
    Clock_Display(genperline2_word5_x,geny_fifth,person_competition[your_number].hour_point[point-1],person_competition[your_number].minute_point[point-1],WHITE);
    puthz(genperline1_word4_x,geny_sixth,24,24,WHITE,"运动距离");
    puthz(genperline12_x,geny_sixth,24,24,WHITE," : ");
    putsz(genperline2_word5_x,geny_sixth,24,16,WHITE,person_competition[your_number].distance);
    putzm(genperline3_word5_x+28,geny_sixth,24,24,WHITE,"m");
    puthz(genperline1_word4_x,geny_seventh,24,24,WHITE,"平均速度");
    puthz(genperline12_x,geny_seventh,24,24,WHITE," : ");
    verb=(double)(person_competition[your_number].distance)/(double)(person_competition[your_number].hour_score*60+person_competition[your_number].minute_score);
    putsz_double(genperline2_word5_x,geny_seventh,24,16,WHITE,verb);
    putzm(genperline3_word5_x-3,geny_seventh,24,16,WHITE,"m/min");
    Solid_Bar(441,geny_eighth+11,993,geny_eighth+13,DARK_GRAY);
    puthz(genperline1_word2_x,geny_ninth,24,24,WHITE,"点位");
    puthz(genperline2_word2_x,geny_ninth,24,24,WHITE,"用时");
    puthz(genperline3_word2_x,geny_ninth,24,24,WHITE,"间隔");
    y=geny_ninth+41;
    for(i=0;i<point;i++)
    {
        putsz(genperline1_word1_x,y,24,24,WHITE,i+1);
        Clock_Display(genperline2_word5_x,y,person_competition[your_number].hour_point[i],person_competition[your_number].minute_point[i],WHITE);
        Clock_Display(genperline3_word5_x,y,person_competition[your_number].hour_interval[i],person_competition[your_number].minute_interval[i],WHITE);
        y=y+41;
    }
}

```

```

/*****
****

```

* 函数名称 *GenMessage_Light_Search*

```

* 函数作用          显示整体战况
* 函数输入          people 人数, age 年龄, zone 区域类型, point 点位数量,
clock_hour_start, clock_minute_start 比赛开始时间
*                  clock_hour_end, clock_minute_end 比赛结束时间,
person_basic 比赛成员基本信息结构体地址
*                  person_competition 比赛成员参赛信息结构体地址,
flag_debug 调试模式开关
* 函数注意          flag_debug 开启调试功能
* 函数输出          无
*****
*****/

void GenMessage_Light_Search(int people,int age,int zone,int clock_hour_start,int clock_minute_start,int clock_hour_end,int clock_minute_end,people_basic *person_basic,people_competition *person_competition)
{
    int i,j,y;
    //refresh right whole page
    Solid_Bar(414,54,1023,767,BLACK);
    //display information
    Message_Title_Draw(gentitle_x,geny_first,age,zone);
    puthz(genperline1_word4_x,geny_second,24,24,WHITE,"起始时间");
    puthz(genperline12_x,geny_second,24,24,WHITE,": ");
    Clock_Display(genperline2_word5_x,geny_second,clock_hour_start,clock_minute_start,WHITE);
    puthz(genperline1_word4_x,geny_third,24,24,WHITE,"结束时间");
    puthz(genperline12_x,geny_third,24,24,WHITE,": ");
    Clock_Display(genperline2_word5_x,geny_third,clock_hour_end,clock_minute_end,WHITE);
    Solid_Bar(441,geny_fourth+11,993,geny_fourth+13,DARK_GRAY);
    puthz(genperline2_word4_x,geny_fifth,24,24,WHITE,"成员成绩");
    puthz(genperline1_word2_x,geny_sixth,24,24,WHITE,"名次");
    puthz(genperline2_word2_x,geny_sixth,24,24,WHITE,"序号");
    puthz(genperline3_word2_x,geny_sixth,24,24,WHITE,"成绩");
    y=geny_sixth+41;
    for(i=0;i<people;i++)
    {
        putsz(genperline1_word1_x,y,24,24,WHITE,i+1);
        for(j=0;j<people;j++)
        {
            if(person_competition[j].rank==i+1)
            {

```

```

        putsz(genperline2_word1_x,y,24,24,WHITE,person_ba
sic[j].number);
        Clock_Display(genperline3_word5_x,y,person_compet
ition[j].hour_score,person_competition[j].minute_score,WHITE);
    }
}
y=y+41;
}
}

```

```

/*****
*****

```

```

* 函数名称      PerMessage_Light_Search
* 函数作用      显示个人战况
* 函数输入      flag 个人战况搜索模式, numorrank 搜索内容, people 人
数, age 年龄, zone 区域类型, point 点位数量
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无

```

```

*****
*****/

```

```

void PerMessage_Light_Search(int flag,int numorrank,int people,in
t age,int zone,int point,people_basic *person_basic,people_compet
ition *person_competition)

```

```

{
    int i,j,num,y;
    if(flag==1)
    {
        j=numorrank-1; //get subscript
    }
    else
    {
        for(i=0;i<people;i++)
        {
            if(person_competition[i].rank==numorrank)
            {
                j=i; //get subscript
            }
        }
    }
    //refresh right page excluding search button
    Solid_Bar(415,150,1023,767,BLACK);
    Message_Title_Draw(pertitle_x,pery_first,age,zone);
}

```

```

        puthz(genperline1_word2_x,pery_second,24,24,WHITE,"序号");
        puthz(genperline12_x,pery_second,24,24,WHITE,"：");
        putsz(genperline2_word1_x,pery_second,24,24,WHITE,person_basi
c[j].number);
        puthz(genperline1_word2_x,pery_third,24,24,WHITE,"名次");
        puthz(genperline12_x,pery_third,24,24,WHITE,"：");
        putsz(genperline2_word1_x,pery_third,24,24,WHITE,person_compe
tition[j].rank);
        puthz(genperline1_word2_x,pery_fourth,24,24,WHITE,"成绩");
        puthz(genperline12_x,pery_fourth,24,24,WHITE,"：");
        Clock_Display(genperline2_word5_x,pery_fourth,person_competit
ion[j].hour_score,person_competition[j].minute_score,WHITE);
        Solid_Bar(441,pery_fifth+11,993,pery_fifth+13,DARK_GRAY);
        puthz(genperline1_word2_x,pery_sixth,24,24,WHITE,"点位");
        puthz(genperline2_word2_x,pery_sixth,24,24,WHITE,"用时");
        puthz(genperline3_word2_x,pery_sixth,24,24,WHITE,"间隔");
        y=pery_sixth+41;
        //output point information
        for(i=0;i<point;i++)
        {
            putsz(genperline1_word1_x,y,24,24,WHITE,i+1);
            Clock_Display(genperline2_word5_x,y,person_competition[j]
.hour_point[i],person_competition[j].minute_point[i],WHITE);
            Clock_Display(genperline3_word5_x,y,person_competition[j]
.hour_interval[i],person_competition[j].minute_interval[i],WHITE)
;
            y=y+41;
        }
    }
}

```

```

/*****
*****

```

```

* 函数名称      ChooseButton_Light_Per
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无

```

```

*****
*****/

```

```

void ChooseButton_Light_Per(int flag)
{
    switch(flag)
    {
        case 1:

```



```

        {
            Button_Edge(Num_x0,Num_y0,Num_x1,Num_y1,WHITE);
            Solid_Bar(Num_x0,Num_y0,Num_x1,Num_y1,SKY_BLUE);
            puthz(Num_x0+TextPer_xplus,Num_y0+TextPer_yplus,2
4,24,BLACK,"按序号");
            break;
        }
    case 2:
    {
        Button_Edge(Rank_x0,Rank_y0,Rank_x1,Rank_y1,WHITE
);
        Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y1,SKY_BLU
E);
        puthz(Rank_x0+TextPer_xplus,Rank_y0+TextPer_yplus
,24,24,BLACK,"按名次");
        break;
    }
}
}

```

```

/*****
*****

```

```

* 函数名称      ChooseButton_Darken_Per
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无

```

```

*****
*****/

```

```

void ChooseButton_Darken_Per(int flag,int flag_choose)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Num_x0,Num_y0,Num_x1,Num_y1,LIGHT_GRA
Y);
            if(flag_choose==2)
            {
                Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y0+1,G
RAY);
            }
            else
            {

```

```

        Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y0+1,DARK_GRAY);
    }
    Solid_Bar(Num_x0,Num_y0,Num_x1,Num_y1,DARK_GRAY);
    puthz(Num_x0+TextPer_xplus,Num_y0+TextPer_yplus,24,24,WHITE,"按序号");
    break;
}
case 2:
{
    Button_Edge(Rank_x0,Rank_y0,Rank_x1,Rank_y1,LIGHT_GRAY);

    if(flag_choose==1)
    {
        Solid_Bar(Num_x0,Num_y1-1,Num_x1,Num_y1,GRAY);
    }
    else
    {
        Solid_Bar(Num_x0,Num_y1-1,Num_x1,Num_y1,DARK_GRAY);
    }
    Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y1,DARK_GRAY);

    puthz(Rank_x0+TextPer_xplus,Rank_y0+TextPer_yplus,24,24,WHITE,"按名次");
    break;
}
}
}

```

```

/*****
*****/

```

```

* 函数名称      ChooseButton_Press_Per
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数输出      无

```

```

*****/
*****/

```

```

void ChooseButton_Press_Per(int flag)
{
    switch(flag)
    {

```

```

        case 1:
        {
            Button_Edge(Num_x0,Num_y0,Num_x1,Num_y1,LIGHT_GRAY);

            Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y0+1,DARK_GRAY);

            Solid_Bar(Num_x0,Num_y0,Num_x1,Num_y1,GRAY);
            puthz(Num_x0+TextPer_xplus,Num_y0+TextPer_yplus,24,24,WHITE,"按序号");
            break;
        }
        case 2:
        {
            Button_Edge(Rank_x0,Rank_y0,Rank_x1,Rank_y1,LIGHT_GRAY);

            Solid_Bar(Rank_x0,Rank_y0-2,Rank_x1,Rank_y0-1,DARK_GRAY);

            Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y1,GRAY);
            puthz(Rank_x0+TextPer_xplus,Rank_y0+TextPer_yplus,24,24,WHITE,"按名次");
            break;
        }
    }
}

```

```

/*****
*****
* 函数名称      PerPage_Light_Search
* 函数作用      绘制右边搜索界面
* 函数输入      flag 绘制页面序号
* 函数输出      无
*****
*****/

```

```

void PerPage_Light_Search(int flag)
{
    Solid_Bar(414,54,1023,767,BLACK);    //refresh right whole page

    switch(flag)
    {
        case 1:
        {
            //input button: 382*80
            Button_Draw(InputButton_x0,InputButton_y0,InputBu

```

```

tton_x1,InputButton_y1,LIGHT_GRAY,DARK_GRAY);
        puthz(InputButton_x0+32,InputButton_y0+24,32,32,L
LIGHT_GRAY,"请输入需要查询的序号");
        //confirm button: 188*80
        Button_Draw(InputConfirm_x0,InputConfirm_y0,Input
Confirm_x1,InputConfirm_y1,Hollow_Color,Solid_Color);
        puthz(InputConfirm_x0+InputText_xplus,InputConfir
m_y0+InputText_yplus,32,32,BLACK,"查询");
        break;
    }
    case 2:
    {
        //input button: 382*80
        Button_Draw(InputButton_x0,InputButton_y0,InputBu
tton_x1,InputButton_y1,LIGHT_GRAY,DARK_GRAY);
        puthz(InputButton_x0+32,InputButton_y0+24,32,32,L
LIGHT_GRAY,"请输入需要查询的名次");
        //confirm button: 188*80
        Button_Draw(InputConfirm_x0,InputConfirm_y0,Input
Confirm_x1,InputConfirm_y1,Hollow_Color,Solid_Color);
        puthz(InputConfirm_x0+InputText_xplus,InputConfir
m_y0+InputText_yplus,32,32,BLACK,"查询");
        break;
    }
}
}

/*****
*****

* 函数名称      ButtonNumRank_Light_Search
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonNumRank_Light_Search(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(InputButton_x0,InputButton_y0,InputBu
tton_x1,InputButton_y1,WHITE);

```

```

        break;
    }
    case 2:
    {
        Button_Edge(InputConfirm_x0,InputConfirm_y0,Input
Confirm_x1,InputConfirm_y1,WHITE);
        Solid_Bar(InputConfirm_x0,InputConfirm_y0,InputCo
nfirm_x1,InputConfirm_y1,SKY_BLUE);
        puthz(InputConfirm_x0+InputText_xplus,InputConfir
m_y0+InputText_yplus,32,32,GRAY,"查询");
        break;
    }
}
}

/*****
*****
* 函数名称      ButtonNumRank_Darken_Search
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonNumRank_Darken_Search(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(InputButton_x0,InputButton_y0,InputBu
tton_x1,InputButton_y1,BLACK);
            break;
        }
        case 2:
        {
            Button_Edge(InputConfirm_x0,InputConfirm_y0,Input
Confirm_x1,InputConfirm_y1,BLACK);
            Button_Draw(InputConfirm_x0,InputConfirm_y0,Input
Confirm_x1,InputConfirm_y1,Hollow_Color,Solid_Color);
            puthz(InputConfirm_x0+InputText_xplus,InputConfir
m_y0+InputText_yplus,32,32,BLACK,"查询");
            break;
        }
    }
}

```

```

    }
}

/*****
*****
* 函数名称      Search
* 函数作用      搜索界面
* 函数输入      page 界面地址, people 人数, age 年龄, zone 区域类型,
weather 天气类型, point 点位数量
*               point_x 点位x 值地址, point_y 点位y 值地址,
clock_hour_start, clock_minute_start 比赛开始时间
*               clock_hour_end, clock_minute_end 比赛结束时间
*               person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
*               your_number 用户参赛序号, your_name 用户名字
* 函数输出      无
*****
*****/
void Search(int *page,int *people,int *age,int *zone,int *weather
,int *point,int clock_hour_start,int clock_minute_start,int clock
_hour_end,int clock_minute_end,people_basic *person_basic,people_
competition *person_competition,int* your_number,char *your_name)
{
    /*按钮状态变量*/
    int flag,flag_press,flag_press_last,flag_revert,flag_msg,check,flag_per,flag_choose,flag_num,flag_rank,numorrank,flag_input,place,i,flag_inputconfirm,flag_mouse,flag_mouse_light;
    /*右键菜单变量*/
    int xm1,ym1,xm2,ym2;
Refresh_Search:
    //SetSVGA64k();
    Solid_Bar(0,0,1023,767,BLACK);
    flag=0,flag_press=0,flag_press_last=0,flag_revert=0,flag_msg=0,check=0,flag_per=0,flag_choose=0,flag_num=0,flag_rank=0,numorrank=0,flag_input=0,place=0,flag_inputconfirm=0,flag_mouse=0,flag_mouse_light=0;
    Draw_Search();
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        /*刷新界面*/
        if(mouse_press(1,1,1023,767)==3)

```

```

{
    mousehide(MouseX,MouseY);
    if(flag_mouse==0)
    {
        flag_mouse=1;
        /*根据鼠标位置绘制右键菜单*/
        if(MouseX>0&&MouseX<864&&MouseY>0&&MouseY<728)
        {
            xm1=MouseX,ym1=MouseY,xm2=MouseX+160,ym2=Mous
eY+40;
        }
        else if(MouseX>863&&MouseX<1024&&MouseY>0&&MouseY
<728)
        {
            xm1=MouseX-
160,ym1=MouseY,xm2=MouseX,ym2=MouseY+40;
        }
        else if(MouseX>0&&MouseX<864&&MouseY>727&&MouseY<
768)
        {
            xm1=MouseX,ym1=MouseY-
40,xm2=MouseX+160,ym2=MouseY;
        }
        else
        {
            xm1=MouseX-160,ym1=MouseY-
40,xm2=MouseX,ym2=MouseY;
        }
        save_image(xm1,ym1,xm2+3,ym2+3);
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    continue;
}
if(flag_mouse==1)
{
    if(mouse_out_press(xm1,ym1,xm2,ym2)==1)
    {
        mousehide(MouseX,MouseY);
        flag_mouse=0;
        printf_image(xm1,ym1,xm2+3,ym2+3);
        resetMouse(MouseX,MouseY);
    }
}

```

```

    }
    if(mouse_press(xm1,ym1,xm2,ym2)==2)
    {
        shape=3;
        puthz(xm1+56,ym1+8,24,24,WHITE,"刷新");
        flag_mouse_light=1;
        continue;
    }
    else if(mouse_press(xm1,ym1,xm2,ym2)==1)
    {
        press=0;
        shape=0;
        //refresh page
        delay(100);
        goto Refresh_Search;
    }
    if(flag_mouse_light!=0)
    {
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        flag_mouse_light=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
//button 1: revert
if(MouseX>Revert_x0&&MouseX<Revert_x1&&MouseY>Revert_y0&&
MouseY<Revert_y1)
{
    if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Revert_y
1)==2)
    {
        shape=3;
        if(flag_revert==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Search(1);
            resetMouse(MouseX,MouseY);
            flag_revert=1;
        }
        continue;
    }
}

```



```

else if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Revert_y1)==1)
{
    mousehide(MouseX,MouseY);
    save_image(General_x0,General_y0,General_x1+3,General_y1+3);
    MsgBox2_Draw_Search();
    resetMouse(MouseX,MouseY);
    /* 绘制二级菜单*/
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        if(MouseX>Confirm_x0&&MouseX<Confirm_x1&& MouseY>Confirm_y0&&MouseY<Confirm_y1)
        {
            if(mouse_press(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1)==2)
            {
                shape=3;
                if(flag_msg==0);
                {
                    mousehide(MouseX,MouseY);
                    MsgBox2Button_Light_Search(1);
                    resetMouse(MouseX,MouseY);
                    flag_msg=1;
                }
                continue;
            }
            else if(mouse_press(Confirm_x0,Confirm_y0,Confirm_x1,Confirm_y1)==1)
            {
                check=1;
                delay(100);
                break;
            }
        }
        if(MouseX>Cancel_x0&&MouseX<Cancel_x1&& MouseY>Cancel_y0&&MouseY<Cancel_y1)
        {
            if(mouse_press(Cancel_x0,Cancel_y0,Cancel_x1,Cancel_y1)==2)
            {
                shape=3;

```

```

        if(flag_msg==0);
        {
            mousehide(MouseX,MouseY);
            MsgBox2Button_Light_Search(2);
            resetMouse(MouseX,MouseY);
            flag_msg=2;
        }
        continue;
    }
    else if(mouse_press(Cancel_x0,Cancel_y0,C
ancel_x1,Cancel_y1)==1)
    {
        mousehide(MouseX,MouseY);
        check=0;
        resetMouse(MouseX,MouseY);
        delay(100);
        break;
    }
}
if(flag_msg!=0)
{
    mousehide(MouseX,MouseY);
    MsgBox2Button_Darken_Search(flag_msg);
    resetMouse(MouseX,MouseY);
    flag_msg=0;
}
if(shape!=0 )
{
    shape=0;
}
}
if(check==1)
{
    *page=0;
    delay(100);
    //go to menu page
    break;
}
else
{
    printf_image(General_x0,General_y0,General_x1
+3,General_y1+3);
    resetMouse(MouseX,MouseY);

```

```

        delay(100);
        //restore search page
        continue;
    }
}
}
if(flag_revert!=0)
{
    mousehide(MouseX,MouseY);
    Button_Darken_Search(flag_revert);
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    flag_revert=0;
}
//button 4: User
if(MouseX>UserCompetition_x0&&MouseX<UserCompetition_x1&&
MouseY>UserCompetition_y0&&MouseY<UserCompetition_y1)
{
    if(mouse_press(UserCompetition_x0,UserCompetition_y0,
UserCompetition_x1,UserCompetition_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Search(4);
            resetMouse(MouseX,MouseY);
            flag=4;
        }
        continue;
    }
    else if(mouse_press(UserCompetition_x0,UserCompetition_y0,
UserCompetition_x1,UserCompetition_y1)==1)
    {
        UserMessage_Light_Search((*people),(*age),(*zone)
,(*point),person_basic,person_competition,(*your_number),your_name);

        flag_input=0;
        place=0;
    }
}

```

```

        numorrank=0;
        flag_inputconfirm=0;
        flag_press_last=flag_press;
        flag_press=4;
        if(flag_press_last!=0)
        {
            Button_Darken_Search(flag_press_last);
        }
        //hide mouse
        Solid_Bar(UserCompetition_x0,UserCompetition_y1+1
,UserCompetition_x1,UserCompetition_y1+15,BLACK);
        Solid_Bar(UserCompetition_x1+1,UserCompetition_y0
,UserCompetition_x1+9,UserCompetition_y1+15,BLACK);
        Button_Darken_Search(3);
        Button_Darken_Search(2);
        Button_Press_Search(4);
        resetMouse(MouseX,MouseY);
        delay(100);
    }
}
//button 2: Gen
if(MouseX>GenCompetition_x0&&MouseX<GenCompetition_x1&&Mo
useY>GenCompetition_y0&&MouseY<GenCompetition_y1)
{
    if(mouse_press(GenCompetition_x0,GenCompetition_y0,Ge
nCompetition_x1,GenCompetition_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Search(2);
            resetMouse(MouseX,MouseY);
            flag=2;
        }
        continue;
    }
    else if(mouse_press(GenCompetition_x0,GenCompetition_
y0,GenCompetition_x1,GenCompetition_y1)==1)
    {
        GenMessage_Light_Search((*people),(*age),(*zone),
clock_hour_start,clock_minute_start,clock_hour_end,clock_minute_e
nd,person_basic,person_competition);
    }
}

```

```

        flag_input=0;
        place=0;
        numorrank=0;
        flag_inputconfirm=0;
        flag_press_last=flag_press;
        flag_press=2;
        if(flag_press_last!=0)
        {
            Button_Darken_Search(flag_press_last);
        }
        //hide mouse
        Solid_Bar(GenCompetition_x0,GenCompetition_y1+1,GenCompetition_x1,PerCompetition_y1-1,BLACK);
        Solid_Bar(GenCompetition_x1+1,GenCompetition_y0,GenCompetition_x1+9,GenCompetition_y1+15,BLACK);
        Button_Darken_Search(3);
        Button_Darken_Search(4);
        Button_Press_Search(2);
        resetMouse(MouseX,MouseY);
        delay(100);
    }
}
//button 3: Per
if(MouseX>PerCompetition_x0&&MouseX<PerCompetition_x1&&MouseY>PerCompetition_y0&&MouseY<PerCompetition_y1)
{
    if(mouse_press(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1)==2)
    {
        shape=3;
        if(flag==0);
        {
            mousehide(MouseX,MouseY);
            Button_Light_Search(3);
            resetMouse(MouseX,MouseY);
            flag=3;
        }
        continue;
    }
    else if(mouse_press(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y1)==1)
    {
        flag_press_last=flag_press;
    }
}

```

```

        flag_press=3;
        if(flag_press_last!=0)
        {
            Button_Darken_Search(flag_press_last);
        }
        //hide mouse
        Solid_Bar(PerCompetition_x0,PerCompetition_y1+1,PerCompetition_x1,PerCompetition_y1+15,BLACK);
        Solid_Bar(PerCompetition_x1+1,PerCompetition_y0,PerCompetition_x1+9,PerCompetition_y1+15,BLACK);
        Button_Darken_Search(2);
        Button_Darken_Search(4);
        Button_Draw(PerCompetition_x0,PerCompetition_y0,PerCompetition_x1,PerCompetition_y0+83,LIGHT_GRAY,DARK_GRAY);
        //draw general button
        Button_Draw(Num_x0-2,Num_y0-2,Rank_x1+2,Rank_y1+2,LIGHT_GRAY,DARK_GRAY);
        //Solid_Bar(Num_x0,Num_y0,Num_x1,Num_y1,DARK_GRAY);
        puthz(Num_x0+TextPer_xplus,Num_y0+TextPer_yplus,24,24,WHITE,"按序号");
        //Solid_Bar(Rank_x0,Rank_y0,Rank_x1,Rank_y1,DARK_GRAY);
        puthz(Rank_x0+TextPer_xplus,Rank_y0+TextPer_yplus,24,24,WHITE,"按名次");
        if(flag_choose!=0)
        {
            ChooseButton_Press_Per(flag_choose);
        }
        //get flag_choose to sure by number or rank
        delay(100);
        resetMouse(MouseX,MouseY);
        /* 绘制二级菜单*/
        while(1)
        {
            newxy(&MouseX,&MouseY,&press);
            //choose button 1-2 (x1-x0+1)*32
            if(MouseX>Num_x0&&MouseX<Num_x1&&MouseY>Num_y0&&MouseY<Num_y1)
            {
                if(mouse_press(Num_x0,Num_y0,Num_x1,Num_y1)==2)
                {

```

```

        shape=3;
        if(flag_per==0);
        {
            mousehide(MouseX,MouseY);
            ChooseButton_Light_Per(1);
            resetMouse(MouseX,MouseY);
            flag_per=1;
        }
        continue;
    }
    else if(mouse_press(Num_x0,Num_y0,Num_x1,
Num_y1)==1)
    {
        flag_choose=1;
        resetMouse(MouseX,MouseY);
        delay(100);
        break;
    }
}
if(MouseX>Rank_x0&&MouseX<Rank_x1&&MouseY>Ran
k_y0&&MouseY<Rank_y1)
{
    if(mouse_press(Rank_x0,Rank_y0,Rank_x1,Ra
nk_y1)==2)
    {
        shape=3;
        if(flag_per==0);
        {
            mousehide(MouseX,MouseY);
            ChooseButton_Light_Per(2);
            resetMouse(MouseX,MouseY);
            flag_per=2;
        }
        continue;
    }
    else if(mouse_press(Rank_x0,Rank_y0,Rank_
x1,Rank_y1)==1)
    {
        flag_choose=2;
        delay(100);
        break;
    }
}

```

```

        if(flag_per!=0)
        {
            if(flag_per!=flag_choose)
            {
                mousehide(MouseX,MouseY);
                ChooseButton_Darken_Per(flag_per,flag
_choose);

                resetMouse(MouseX,MouseY);
            }
            else
            {
                mousehide(MouseX,MouseY);
                ChooseButton_Press_Per(flag_per);
                resetMouse(MouseX,MouseY);
            }
            flag_per=0;
        }
        if(shape!=0)
        {
            shape=0;
        }
    }
    //refresh
    Solid_Bar(Num_x0-2,Num_y0-
2,Rank_x1+2,Rank_y1+2,BLACK);
    Button_Draw(PerCompetition_x0,PerCompetition_y0,P
erCompetition_x1,PerCompetition_y1,Hollow_Color,Solid_Color);
    puthz(PerCompetition_x0+Text_xplus,PerCompetition
_y0+Text_yplus,32,32,BLACK,"选手成绩查询");
    TriangleImage(PerCompetition_x0+311,PerCompetitio
n_y0+31,DARK_GRAY);
    //restore input button
    flag_input=0;
    place=0;
    numorrank=0;
    flag_inputconfirm=0;
    //draw search button
    PerPage_Light_Search(flag_choose);
    Button_Press_Search(3);
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}

```



```

    }
    //input button and confirm button
    if(flag_press==3)
    {
        if(flag_input)
        {
            //init
            if(place==0)
            {
                Button_Draw(InputButton_x0,InputButton_y0,InputButton_x1,InputButton_y1,LIGHT_GRAY,DARK_GRAY);
                Solid_Bar(InputButton_x0+15,InputButton_y0+24,InputButton_x0+17,InputButton_y1-24,WHITE);
            }
            //scan keyboard
            if(bioskey(1))
            {
                i=bioskey(0);
            }
            else
            {
                i=0;
            }
            //input
            if(place==0)
            {
                if(i==2864||((i>=561&&i<=2617&&((i-304)%257==0)))) //2864->0
                {
                    if(i==2864)
                    {
                        numorrank=0;
                    }
                    else
                    {
                        numorrank=(i-304)/257;
                    }
                    i=0;
                    Solid_Bar(InputButton_x0+15,InputButton_y0+24,InputButton_x0+17,InputButton_y1-24,DARK_GRAY);
                    putsz(InputButton_x0+15,InputButton_y0+24,32,32,WHITE,numorrank);
                    place=1;
                }
            }
        }
    }

```

```

        }
    }
    else
    {
        if(i==3592)        //3592->Backspace
        {
            numorrank=0;
            i=0;
            Solid_Bar(InputButton_x0+15,InputButton_y
0+24,InputButton_x0+48,InputButton_y0+57,DARK_GRAY);
            place=0;
        }
    }
    //mean in input mode
    if(flag_inputconfirm&&place&&flag_input)
    {
        putsz(InputButton_x0+15,InputButton_y0+24,32,
32,DARK_GRAY,numorrank);
        putsz(InputButton_x0+15,InputButton_y0+24,32,
32,WHITE,numorrank);
    }
}
if(flag_choose==1)
{
    if(MouseX>InputButton_x0&&MouseX<InputButton_x1&&
MouseY>InputButton_y0&&MouseY<InputButton_y1)
    {
        if(mouse_press(InputButton_x0,InputButton_y0,
InputButton_x1,InputButton_y1)==2)
        {
            shape=2;
            if(flag_num==0);
            {
                ButtonNumRank_Light_Search(1);
                flag_num=1;
            }
            continue;
        }
        else if(mouse_press(InputButton_x0,InputButto
n_y0,InputButton_x1,InputButton_y1)==1)
        {
            flag_input=1;
            continue;

```

```

        }
    }
    if(MouseX>InputConfirm_x0&&MouseX<InputConfirm_x1
&&MouseY>InputConfirm_y0&&MouseY<InputConfirm_y1)
    {
        if(mouse_press(InputConfirm_x0,InputConfirm_y
0,InputConfirm_x1,InputConfirm_y1)==2)
        {
            shape=3;
            if(flag_num==0);
            {
                mousehide(MouseX,MouseY);
                ButtonNumRank_Light_Search(2);
                resetMouse(MouseX,MouseY);
                flag_num=2;
            }
            continue;
        }
        else if(mouse_press(InputConfirm_x0,InputConf
irm_y0,InputConfirm_x1,InputConfirm_y1)==1)
        {
            shape=0;
            //darken inputconfirm
            ButtonNumRank_Darken_Search(2);
            //hide mouse
            Solid_Bar(InputConfirm_x0,InputConfirm_y1
+1,InputConfirm_x1,InputConfirm_y1+15,BLACK);
            Solid_Bar(InputConfirm_x1+1,InputConfirm_
y0,InputConfirm_x1+9,InputConfirm_y1+15,BLACK);
            if(place==0)
            {
                save_image(General_x0,General_y0,Gene
ral_x1+3,General_y1+3);

                MsgBox0NoInput_Draw_Search();
                while(1)
                {
                    if(bioskey(0)==ENTER)
                    {
                        break;
                    }
                }
                printf_image(General_x0,General_y0,Ge
neral_x1+3,General_y1+3);

```

```

        }
        else if(numorrank==0||numorrank>(*people)
)
        {
            save_image(General_x0,General_y0,General_x1+3,General_y1+3);
            MsgBox0NoSearch_Draw_Search();
            while(1)
            {
                if(bioskey(0)==ENTER)
                {
                    break;
                }
            }
            printf_image(General_x0,General_y0,General_x1+3,General_y1+3);
        }
        else
        {
            PerMessage_Light_Search(1,numorrank,(*people),(*age),(*zone),(*point),person_basic,person_competition);
        }
        flag_input=0;
        flag_inputconfirm=1;
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if( flag_num!=0)
{
    if(flag_num==1)
    {
        ButtonNumRank_Darken_Search(1);
    }
    else
    {
        mousehide(MouseX,MouseY);
        ButtonNumRank_Darken_Search(2);
        resetMouse(MouseX,MouseY);
    }
    if(flag_mouse==1)

```

```

        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        flag_num=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
if(flag_choose==2)
{
    if(MouseX>InputButton_x0&&MouseX<InputButton_x1&&
MouseY>InputButton_y0&&MouseY<InputButton_y1)
    {
        if(mouse_press(InputButton_x0,InputButton_y0,
InputButton_x1,InputButton_y1)==2)
        {
            shape=2;
            if(flag_rank==0);
            {
                ButtonNumRank_Light_Search(1);
                flag_rank=1;
            }
            continue;
        }
        else if(mouse_press(InputButton_x0,InputButto
n_y0,InputButton_x1,InputButton_y1)==1)
        {
            flag_input=1;
            continue;
        }
    }
    if(MouseX>InputConfirm_x0&&MouseX<InputConfirm_x1
&&MouseY>InputConfirm_y0&&MouseY<InputConfirm_y1)
    {
        if(mouse_press(InputConfirm_x0,InputConfirm_y
0,InputConfirm_x1,InputConfirm_y1)==2)
        {
            shape=3;
            if(flag_rank==0);
            {

```

```

        mousehide(MouseX,MouseY);
        ButtonNumRank_Light_Search(2);
        resetMouse(MouseX,MouseY);
        flag_rank=2;
    }
    continue;
}
else if(mouse_press(InputConfirm_x0,InputConf
irm_y0,InputConfirm_x1,InputConfirm_y1)==1)
{
    shape=0;
    //darken inputconfirm
    ButtonNumRank_Darken_Search(2);
    //hide mouse
    Solid_Bar(InputConfirm_x0,InputConfirm_y1
+1,InputConfirm_x1,InputConfirm_y1+15,BLACK);
    Solid_Bar(InputConfirm_x1+1,InputConfirm_
y0,InputConfirm_x1+9,InputConfirm_y1+15,BLACK);
    if(place==0)
    {
        save_image(General_x0,General_y0,Gene
ral_x1+3,General_y1+3);
        MsgBox0NoInput_Draw_Search();
        while(1)
        {
            if(bioskey(0)==ENTER)
            {
                break;
            }
        }
        printf_image(General_x0,General_y0,Ge
neral_x1+3,General_y1+3);
    }
    else if(numorrank==0||numorrank>(*people)
)
    {
        save_image(General_x0,General_y0,Gene
ral_x1+3,General_y1+3);
        MsgBox0NoSearch_Draw_Search();
        while(1)
        {
            if(bioskey(0)==ENTER)
            {

```

```

                break;
            }
        }
        printf_image(General_x0,General_y0,General_x1+3,General_y1+3);
    }
    else
    {
        PerMessage_Light_Search(2,numorrank,(*people),(*age),(*zone),(*point),person_basic,person_competition)
;
    }
    flag_input=0;
    flag_inputconfirm=1;
    getMousebk(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(flag_rank!=0)
{
    if(flag_rank==1)
    {
        ButtonNumRank_Darken_Search(1);
    }
    else
    {
        mousehide(MouseX,MouseY);
        ButtonNumRank_Darken_Search(2);
        resetMouse(MouseX,MouseY);
    }
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    flag_rank=0;
}
if(shape!=0)
{
    shape=0;
}
}

```

```

    }
    if(flag!=0)
    {
        if(flag!=flag_press)
        {
            mousehide(MouseX,MouseY);
            Button_Darken_Search(flag);
            resetMouse(MouseX,MouseY);
        }
        else
        {
            mousehide(MouseX,MouseY);
            Button_Press_Search(flag);
            resetMouse(MouseX,MouseY);
        }
        if(flag_mouse==1)
        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        flag=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}
}

```

32. setting.c

```
#include"headfile.h"
```

```

/*****
*****
* @author          ytm
* @date            2022-4-15
*****
*****/

#define Text_xplus    160
#define Text_yplus    24

//driver
#define Button1_x0     429

```



```

#define Button1_y0      70

//follower
#define Button1_x1      (Button1_x0+575)
#define Button1_y1      (Button1_y0+119)
#define Button2_x0      429
#define Button2_y0      (Button1_y0+138)
#define Button2_x1      (Button1_x0+575)
#define Button2_y1      (Button1_y0+258)
#define Button3_x0      429
#define Button3_y0      (Button1_y0+277)
#define Button3_x1      (Button1_x0+575)
#define Button3_y1      (Button1_y0+397)
#define Button4_x0      429
#define Button4_y0      (Button1_y0+416)
#define Button4_x1      (Button1_x0+575)
#define Button4_y1      (Button1_y0+536)

/*****
*****

* 函数名称      Setting
* 函数作用      设置界面
* 函数输入      page 界面地址, people 人数, age 年龄, zone 区域类型,
weather 天气类型, point 点位数量
*                point_x 点位x 值地址, point_y 点位y 值地址
*                clock_hour, clock_minute 当前时间,
clock_hour_start, clock_minute_start 比赛开始时间
*                person_basic 比赛成员基本信息结构体地址,
person_competition 比赛成员参赛信息结构体地址
* 函数输出      无
*****
*****/

void Setting(int *page,int *people, int *age, int *zone, int *weather,int *point,int* point_x,int* point_y,int* clock_hour,int* clock_minute,int *clock_hour_start,int *clock_minute_start,people_basic *person_basic,people_competition *person_competition)
{
    int flag_left,flag_press_last,flag_press,flag_start,flag_msg,
check,flag_revert,flag_people,flag_point,flag_right,flag_right3_press,flag_right3_press_last,flag_right4_press,flag_right4_press_last,flag_right5_press,flag_right5_press_last,flag_mouse,flag_mouse_light;
    /*右键菜单变量*/

```

```

    int xm1,ym1,xm2,ym2;
Refresh_Setting:
    flag_left=0,flag_press_last=0,flag_press=0,flag_start=0,flag_
msg=0,flag_revert=0,flag_people=0,flag_point=0,flag_right=0,flag_
right3_press=0,flag_right3_press_last=0,flag_right4_press=0,flag_
right4_press_last=0,flag_right5_press=0,flag_right5_press_last=0,
flag_mouse=0,flag_mouse_light=0;
    //SetSVGA64k();
    Solid_Bar(0,0,1023,767,BLACK);
    PageLeft_Draw_Setting();
    resetMouse(MouseX,MouseY);
    (*people)=1,(*age)=0,(*zone)=0,(*weather)=0,(*point)=2,People
_Init(person_basic,person_competition);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        /* 刷新界面*/
        if(mouse_press(1,1,1023,767)==3)
        {
            mousehide(MouseX,MouseY);
            if(flag_mouse==0)
            {
                flag_mouse=1;
                /* 根据鼠标位置绘制右键菜单*/
                if(MouseX>0&&MouseX<864&&MouseY>0&&MouseY<728)
                {
                    xm1=MouseX,ym1=MouseY,xm2=MouseX+160,ym2=Mous
eY+40;
                }
                else if(MouseX>863&&MouseX<1024&&MouseY>0&&MouseY
<728)
                {
                    xm1=MouseX-
160,ym1=MouseY,xm2=MouseX,ym2=MouseY+40;
                }
                else if(MouseX>0&&MouseX<864&&MouseY>727&&MouseY<
768)
                {
                    xm1=MouseX,ym1=MouseY-
40,xm2=MouseX+160,ym2=MouseY;
                }
                else
                {

```

```

        xm1=MouseX-160,ym1=MouseY-
40,xm2=MouseX,ym2=MouseY;
    }
    save_image(xm1,ym1,xm2+3,ym2+3);
    Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
    puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
}
resetMouse(MouseX,MouseY);
continue;
}
if(flag_mouse==1)
{
    if(mouse_out_press(xm1,ym1,xm2,ym2)==1)
    {
        mousehide(MouseX,MouseY);
        flag_mouse=0;
        printf_image(xm1,ym1,xm2+3,ym2+3);
        resetMouse(MouseX,MouseY);
    }
    if(mouse_press(xm1,ym1,xm2,ym2)==2)
    {
        shape=3;
        puthz(xm1+56,ym1+8,24,24,WHITE,"刷新");
        flag_mouse_light=1;
        continue;
    }
    else if(mouse_press(xm1,ym1,xm2,ym2)==1)
    {
        press=0;
        shape=0;
        //refresh page
        delay(100);
        goto Refresh_Setting;
    }
    if(flag_mouse_light!=0)
    {
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        flag_mouse_light=0;
    }
    if(shape!=0)
    {
        shape=0;
    }
}

```

```

    }
    //basic button 1
    if(MouseX>Start_x0&&MouseX<Start_x1&&MouseY>Start_y0&&MouseY<Start_y1)
    {
        if(mouse_press(Start_x0,Start_y0,Start_x1,Start_y1)==
2)
        {
            shape=3;
            if(flag_start==0);
            {
                mousehide(MouseX,MouseY);
                ButtonLeft_Light_Setting(flag_start);
                resetMouse(MouseX,MouseY);
                flag_start=1;
            }
            continue;
        }
        else if(mouse_press(Start_x0,Start_y0,Start_x1,Start_
y1)==1)
        {
            if((*age)==0||(*zone)==0||(*weather)==0)
            {
                ButtonLeft_Press_Setting(1);
                //hide mouse
                Solid_Bar(Start_x0,Start_y1+1,Start_x1,Start_
y1+15,BLACK);
                Solid_Bar(Start_x1+1,Start_y0,Start_x1+9,Star
t_y1+15,BLACK);
                ButtonLeft_Darken_Setting(2);
                ButtonLeft_Darken_Setting(3);
                ButtonLeft_Darken_Setting(4);
                ButtonLeft_Darken_Setting(5);
                ButtonLeft_Darken_Setting(6);
                //display setting error page
                MsgBox2_Draw_Setting();
                resetMouse(MouseX,MouseY);
                while(1)
                {
                    newxy(&MouseX,&MouseY,&press);
                    if(MouseX>Confirm_x0&&MouseX<Confirm_x1&&
MouseY>Confirm_y0&&MouseY<Confirm_y1)
                    {

```

```

        if(mouse_press(Confirm_x0,Confirm_y0,
Confirm_x1,Confirm_y1)==2)
        {
            shape=3;
            if(flag_msg==0);
            {
                MsgBox2Button_Light(1);
                flag_msg=1;
            }
            continue;
        }
        else if(mouse_press(Confirm_x0,Confir
m_y0, Confirm_x1,Confirm_y1)==1)
        {
            check=1;
            break;
        }
    }
    if(MouseX>Cancel_x0&&MouseX<Cancel_x1&& M
ouseY>Cancel_y0&&MouseY<Cancel_y1)
    {
        if(mouse_press(Cancel_x0,Cancel_y0,Ca
ncel_x1,Cancel_y1)==2)
        {
            shape=3;
            if(flag_msg==0);
            {
                MsgBox2Button_Light(2);
                flag_msg=2;
            }
            continue;
        }
        else if(mouse_press(Cancel_x0,Cancel_
y0,Cancel_x1,Cancel_y1)==1)
        {
            check=0;
            break;
        }
    }
    if(flag_msg!=0)
    {
        MsgBox2Button_Darken(flag_msg);
        flag_msg=0;
    }

```

```

        }
        if(shape!=0)
        {
            shape=0;
        }
    }
    if(check==1)
    {
        if((*age)==0)
        {
            (*age)=1;
        }
        if((*zone)==0)
        {
            (*zone)=1;
        }
        if((*weather)==0)
        {
            (*weather)=1;
        }
        //SetSVGA64k();
        Solid_Bar(0,0,1023,767,BLACK);
        Loading_Play(512,372,(*people),(*age),(*zone),(*weather),(*point),point_x,point_y,clock_hour,clock_minute,
        clock_hour_start,clock_minute_start,person_basic,person_competition);

        //loading 1.6s
        *page = 3;
        shape=0;
        delay(100);
        //go to play_simulating page
        break;
    }
    else
    {
        {
            //msgbox close
            Solid_Bar(344,230,680,610,BLACK);

            //boundaryBar refresh: 410-
            Boundary_Draw(410,53,4,DARK_GRAY);

```

413 DARK_GRAY

```

//button 2-
3 refresh: 384*86 DARK_GRAY + LIGHT_GRAY
        Button_Draw(People_x0,People_y0,Peopl
e_x1,People_y1,Hollow_Color,Solid_Color);
        puthz(People_x0+Text_xplus,People_y0+
Text_yplus,32,32,BLACK,"人数");

        Button_Draw(Age_x0,Age_y0,Age_x1,Age_
y1,Hollow_Color,Solid_Color);
        puthz(Age_x0+Text_xplus,Age_y0+Text_y
plus,32,32,BLACK,"年龄");

//button 4-
6 refresh: 384*80 DARK_GRAY + LIGHT_GRAY

        Button_Draw(Zone_x0,Zone_y0,Zone_x1,Z
one_y1,Hollow_Color,Solid_Color);
        puthz(Zone_x0+Text_xplus,Zone_y0+Text
_yplus,32,32,BLACK,"区域");

        Button_Draw(Weather_x0,Weather_y0,Wea
ther_x1,Weather_y1,Hollow_Color,Solid_Color);
        puthz(Weather_x0+Text_xplus,Weather_y
0+Text_yplus,32,32,BLACK,"天气");

//set num GRAY
if(flag_press==2)
{
    PeopleNum_Draw_Setting1((*people)
,GRAY);
}
else if(flag_press==3)
{
    ButtonRight_Draw_Setting345(3);
    if((*age)!=0)
    {
        ButtonRight3_Press_Setting345
(*age);
    }
}
else if(flag_press==4)
{
    ButtonRight_Draw_Setting345(4);

```

```

        if((*zone)!=0)
        {
            ButtonRight3_Press_Setting345
(*zone);
        }
    }
    else if(flag_press==5)
    {
        ButtonRight_Draw_Setting345(5);
        if((*weather)!=0)
        {
            ButtonRight3_Press_Setting345
(*weather);
        }
    }
    else if(flag_press==6)
    {
        PointNum_Draw_Setting6((*point),G
RAY);
    }
}
resetMouse(MouseX,MouseY);
shape=0;
delay(100);
//refresh setting page
continue;
}
}
//SetSVGA64k();
Solid_Bar(0,0,1023,767,BLACK);
Loading_Play(512,372,(*people),(*age),(*zone),(*w
eather),(*point),point_x,point_y,clock_hour,clock_minute,clock_ho
ur_start,clock_minute_start,person_basic,person_competition);
//Loading 1.6s
*page = 3;
shape=0;
delay(100);
//go to play_simulating page
break;
}
}
if(flag_start!=0)
{

```



```

        mousehide(MouseX,MouseY);
        ButtonLeft_Darken_Setting(flag_start);
        if(flag_mouse==1)
        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        resetMouse(MouseX,MouseY);
        flag_start=0;
    }
    //special button 2
    if(MouseX>People_x0&&MouseX<People_x1&&MouseY>People_y0&&
MouseY<People_y1)
    {
        if(mouse_press(People_x0,People_y0,People_x1,People_y
1)==2)
        {
            shape=3;
            if(flag_left==0);
            {
                mousehide(MouseX,MouseY);
                ButtonLeft_Light_Setting(2);
                resetMouse(MouseX,MouseY);
                flag_left=2;
            }
            continue;
        }
        else if(mouse_press(People_x0,People_y0,People_x1,Peo
ple_y1)==1)
        {
            flag_press_last=flag_press;
            flag_press=2;
            if(flag_press_last!=0)
            {
                ButtonLeft_Darken_Setting(flag_press_last);
            }
            ButtonLeft_Press_Setting(2);
            //hide mouse
            Solid_Bar(People_x0,People_y1+1,People_x1,Age_y1-
1,BLACK);
            Solid_Bar(People_x1+1,People_y0,People_x1+9,Peopl
e_y1+15,BLACK);
            ButtonLeft_Darken_Setting(3);

```

```

        ButtonLeft_Darken_Setting(4);
        ButtonLeft_Darken_Setting(5);
        ButtonLeft_Darken_Setting(6);
        ButtonRight_Draw_Setting1(people);
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(flag_press==2)
{
    if(MouseX>NumMinus_x0&&MouseX<NumMinus_x1&&MouseY>Num
Minus_y0&&MouseY<NumMinus_y1)
    {
        if(mouse_press(NumMinus_x0,NumMinus_y0,NumMinus_x
1,NumMinus_y1)==2)
        {
            shape=3;
            if(flag_people==0);
            {
                mousehide(MouseX,MouseY);
                ButtonPeopleNum_Light_Setting1(1);
                resetMouse(MouseX,MouseY);
                flag_people=1;
            }
            continue;
        }
        else if(mouse_press(NumMinus_x0,NumMinus_y0,NumMi
nus_x1,NumMinus_y1)==1)
        {
            shape=0;
            if((*people)!=1)
            {
                (*people)--;
                PeopleNum_Draw_Setting1((*people),WHITE);
                press=2;
                delay(300);
                //in order to prevent mouse is always cli
cking
            }
            ButtonPeopleNum_Darken_Setting1(2);
            ButtonPeopleNum_Darken_Setting1(3);
            continue;
        }
    }
}

```

```

        }
    }
    if(MouseX>PeopleNum_x0&&MouseX<PeopleNum_x1&&MouseY>P
eopleNum_y0&&MouseY<PeopleNum_y1)
    {
        if(mouse_press(PeopleNum_x0,PeopleNum_y0,PeopleNu
m_x1,PeopleNum_y1)==2)
        {
            shape=1;
            if(flag_people==0);
            {
                mousehide(MouseX,MouseY);
                ButtonPeopleNum_Light_Setting1(2);
                resetMouse(MouseX,MouseY);
                flag_people=2;
            }
            continue;
        }
    }
    if(MouseX>NumPlus_x0&&MouseX<NumPlus_x1&&MouseY>NumPl
us_y0&&MouseY<NumPlus_y1)
    {
        if(mouse_press(NumPlus_x0,NumPlus_y0,NumPlus_x1,N
umPlus_y1)==2)
        {
            shape=3;
            if(flag_people==0);
            {
                mousehide(MouseX,MouseY);
                ButtonPeopleNum_Light_Setting1(3);
                resetMouse(MouseX,MouseY);
                flag_people=3;
            }
            continue;
        }
        else if(mouse_press(NumPlus_x0,NumPlus_y0,NumPlus
_x1,NumPlus_y1)==1)
        {
            shape=0;
            if((*people)!=6)
            {
                (*people)++;
                PeopleNum_Draw_Setting1((*people),WHITE);

```

```

        press=2;
        delay(300);
        //in order to prevent mouse is always cli
cking
    }
    ButtonPeopleNum_Darken_Setting1(1);
    ButtonPeopleNum_Darken_Setting1(2);
    continue;
}
}
if(flag_people!=0)
{
    mousehide(MouseX,MouseY);
    ButtonPeopleNum_Darken_Setting1(flag_people);
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    flag_people=0;
}
if(shape!=0)
{
    shape=0;
}
}
//special button 3
if(MouseX>Age_x0&&MouseX<Age_x1&&MouseY>Age_y0&&MouseY<Age_y1)
{
    if(mouse_press(Age_x0,Age_y0,Age_x1,Age_y1)==2)
    {
        shape=3;
        if(flag_left==0);
        {
            mousehide(MouseX,MouseY);
            ButtonLeft_Light_Setting(3);
            resetMouse(MouseX,MouseY);
            flag_left=3;
        }
        continue;
    }
}

```

```

else if(mouse_press(Age_x0,Age_y0,Age_x1,Age_y1)==1)
{
    flag_press_last=flag_press;
    flag_press=3;
    if(flag_press_last!=0)
    {
        ButtonLeft_Darken_Setting(flag_press_last);
    }
    ButtonLeft_Press_Setting(3);
    //hide mouse
    Solid_Bar(Age_x0,Age_y1+1,Age_x1,Age_y1+15,BLACK)
;
    Solid_Bar(Age_x1+1,Age_y0,Age_x1+9,Age_y1+15,BLAC
K);

    ButtonLeft_Darken_Setting(2);
    ButtonLeft_Darken_Setting(4);
    ButtonLeft_Darken_Setting(5);
    ButtonLeft_Darken_Setting(6);
    //hide mouse
    ButtonRight_Draw_Setting345(3);
    if((*age)!=0)
    {
        ButtonRight3_Press_Setting345(*age);
    }
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(flag_press==3)
{
    if(MouseX>Button1_x0&&MouseX<Button1_x1&&MouseY>Butto
n1_y0&&MouseY<Button1_y1)
    {
        if(mouse_press(Button1_x0,Button1_y0,Button1_x1,B
utton1_y1)==2)
        {
            shape=3;
            if(flag_right==0);
            {
                mousehide(MouseX,MouseY);
                ButtonRight3_Light_Setting345(1);
                resetMouse(MouseX,MouseY);

```

```

        flag_right=1;
    }
    continue;
}
else if(mouse_press(Button1_x0,Button1_y0,Button1
_x1,Button1_y1)==1)
{
    flag_right3_press_last=flag_right3_press;
    flag_right3_press=1;
    if(flag_right3_press_last!=0)
    {
        ButtonRight3_Darken_Setting345(flag_right
3_press_last);
    }
    ButtonRight3_Darken_Setting345(2);
    ButtonRight3_Darken_Setting345(3);
    ButtonRight3_Darken_Setting345(4);
    ButtonRight3_Press_Setting345(1);
    Solid_Bar(429,190,1004,207,BLACK);
    Solid_Bar(1005,70,1013,204,BLACK);
    //hide mouse: button 1
    (*age)=1;
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(MouseX>Button2_x0&&MouseX<Button2_x1&&MouseY>Butto
n2_y0&&MouseY<Button2_y1)
{
    if(mouse_press(Button2_x0,Button2_y0,Button2_x1,B
utton2_y1)==2)
    {
        shape=3;
        if(flag_right==0);
        {
            mousehide(MouseX,MouseY);
            ButtonRight3_Light_Setting345(2);
            resetMouse(MouseX,MouseY);
            flag_right=2;
        }
        continue;
    }
}

```

```

else if(mouse_press(Button2_x0,Button2_y0,Button2
_x1,Button2_y1)==1)
{
    flag_right3_press_last=flag_right3_press;
    flag_right3_press=2;
    if(flag_right3_press_last!=0)
    {
        ButtonRight3_Darken_Setting345(flag_right
3_press_last);
    }
    ButtonRight3_Darken_Setting345(1);
    ButtonRight3_Darken_Setting345(3);
    ButtonRight3_Darken_Setting345(4);
    ButtonRight3_Press_Setting345(2);
    Solid_Bar(429,329,1004,346,BLACK);
    Solid_Bar(1005,208,1013,343,BLACK);
    //hide mouse: button 2
    (*age)=2;
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(MouseX>Button3_x0&&MouseX<Button3_x1&&MouseY>Butto
n3_y0&&MouseY<Button3_y1)
{
    if(mouse_press(Button3_x0,Button3_y0,Button3_x1,B
utton3_y1)==2)
    {
        shape=3;
        if(flag_right==0);
        {
            mousehide(MouseX,MouseY);
            ButtonRight3_Light_Setting345(3);
            resetMouse(MouseX,MouseY);
            flag_right=3;
        }
        continue;
    }
    else if(mouse_press(Button3_x0,Button3_y0,Button3
_x1,Button3_y1)==1)
    {
        flag_right3_press_last=flag_right3_press;

```

```

        flag_right3_press=3;
        if(flag_right3_press_last!=0)
        {
            ButtonRight3_Darken_Setting345(flag_right
3_press_last);
        }
        ButtonRight3_Darken_Setting345(1);
        ButtonRight3_Darken_Setting345(2);
        ButtonRight3_Darken_Setting345(4);
        ButtonRight3_Press_Setting345(3);
        Solid_Bar(429,468,1004,485,BLACK);
        Solid_Bar(1005,347,1013,482,BLACK);
        //hide mouse: button 3
        (*age)=3;
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(MouseX>Button4_x0&&MouseX<Button4_x1&&MouseY>Butto
n4_y0&&MouseY<Button4_y1)
{
    if(mouse_press(Button4_x0,Button4_y0,Button4_x1,B
utton4_y1)==2)
    {
        shape=3;
        if(flag_right==0);
        {
            mousehide(MouseX,MouseY);
            ButtonRight3_Light_Setting345(4);
            resetMouse(MouseX,MouseY);
            flag_right=4;
        }
        continue;
    }
    else if(mouse_press(Button4_x0,Button4_y0,Button4
_x1,Button4_y1)==1)
    {
        flag_right3_press_last=flag_right3_press;
        flag_right3_press=4;
        if(flag_right3_press_last!=0)
        {
            ButtonRight3_Darken_Setting345(flag_right

```



```

3_press_last);
    }
    ButtonRight3_Darken_Setting345(1);
    ButtonRight3_Darken_Setting345(2);
    ButtonRight3_Darken_Setting345(3);
    ButtonRight3_Press_Setting345(4);
    Solid_Bar(429,607,1004,624,BLACK);
    Solid_Bar(1005,486,1013,621,BLACK);
    //hide mouse: button 4
    (*age)=4;
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(flag_right!=0)
{
    if(flag_right!=flag_right3_press)
    {
        mousehide(MouseX,MouseY);
        ButtonRight3_Darken_Setting345(flag_right);
        resetMouse(MouseX,MouseY);
    }
    else
    {
        mousehide(MouseX,MouseY);
        ButtonRight3_Press_Setting345(flag_right);
        resetMouse(MouseX,MouseY);
    }
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    flag_right=0;
}
if(shape!=0)
{
    shape=0;
}
}
//special button 4
if(MouseX>Zone_x0&&MouseX<Zone_x1&&MouseY>Zone_y0&&MouseY

```

```

<Zone_y1)
{
    if(mouse_press(Zone_x0,Zone_y0,Zone_x1,Zone_y1)==2)
    {
        shape=3;
        if(flag_left==0);
        {
            mousehide(MouseX,MouseY);
            ButtonLeft_Light_Setting(4);
            resetMouse(MouseX,MouseY);
            flag_left=4;
        }
        continue;
    }
    else if(mouse_press(Zone_x0,Zone_y0,Zone_x1,Zone_y1)=
=1)
    {
        flag_press_last=flag_press;
        flag_press=4;
        if(flag_press_last!=0)
        {
            ButtonLeft_Darken_Setting(flag_press_last);
        }
        ButtonLeft_Press_Setting(4);
        //hide mouse
        Solid_Bar(Zone_x0,Zone_y1+1,Zone_x1,Weather_y1-
1,BLACK);
        Solid_Bar(Zone_x1+1,Zone_y0,Zone_x1+9,Zone_y1+15,
BLACK);

        ButtonLeft_Darken_Setting(2);
        ButtonLeft_Darken_Setting(3);
        ButtonLeft_Darken_Setting(5);
        ButtonLeft_Darken_Setting(6);
        ButtonRight_Draw_Setting345(4);
        if((*zone)!=0)
        {
            ButtonRight4_Press_Setting345(*zone);
        }
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}

```

```

        if(flag_press==4)
        {
            if(MouseX>Button1_x0&&MouseX<Button1_x1&&MouseY>Button1_y0&&MouseY<Button1_y1)
            {
                if(mouse_press(Button1_x0,Button1_y0,Button1_x1,Button1_y1)==2)
                {
                    shape=3;
                    if(flag_right==0);
                    {
                        mousehide(MouseX,MouseY);
                        ButtonRight4_Light_Setting345(1);
                        resetMouse(MouseX,MouseY);
                        flag_right=1;
                    }
                    continue;
                }
                else if(mouse_press(Button1_x0,Button1_y0,Button1_x1,Button1_y1)==1)
                {
                    flag_right4_press_last=flag_right4_press;
                    flag_right4_press=1;
                    if(flag_right4_press_last!=0)
                    {
                        ButtonRight4_Darken_Setting345(flag_right4_press_last);
                    }
                    ButtonRight4_Darken_Setting345(2);
                    ButtonRight4_Darken_Setting345(3);
                    ButtonRight4_Press_Setting345(1);
                    Solid_Bar(429,190,1004,207,BLACK);
                    Solid_Bar(1005,70,1013,204,BLACK);
                    //hide mouse: button 1
                    (*zone)=1;
                    resetMouse(MouseX,MouseY);
                    delay(100);
                    continue;
                }
            }
            if(MouseX>Button2_x0&&MouseX<Button2_x1&&MouseY>Button2_y0&&MouseY<Button2_y1)
            {

```

```

        if(mouse_press(Button2_x0,Button2_y0,Button2_x1,B
utton2_y1)==2)
        {
            shape=3;
            if(flag_right==0);
            {
                mousehide(MouseX,MouseY);
                ButtonRight4_Light_Setting345(2);
                resetMouse(MouseX,MouseY);
                flag_right=2;
            }
            continue;
        }
        else if(mouse_press(Button2_x0,Button2_y0,Button2
_x1,Button2_y1)==1)
        {
            flag_right4_press_last=flag_right4_press;
            flag_right4_press=2;
            if(flag_right4_press_last!=0)
            {
                ButtonRight4_Darken_Setting345(flag_right
4_press_last);
            }
            ButtonRight4_Darken_Setting345(1);
            ButtonRight4_Darken_Setting345(3);
            ButtonRight4_Press_Setting345(2);
            Solid_Bar(429,329,1004,346,BLACK);
            Solid_Bar(1005,208,1013,343,BLACK);
            //hide mouse: button 2
            (*zone)=2;
            resetMouse(MouseX,MouseY);
            delay(100);
            continue;
        }
    }
    if(MouseX>Button3_x0&&MouseX<Button3_x1&&MouseY>Butto
n3_y0&&MouseY<Button3_y1)
    {
        if(mouse_press(Button3_x0,Button3_y0,Button3_x1,B
utton3_y1)==2)
        {
            shape=3;
            if(flag_right==0);

```

```

        {
            mousehide(MouseX,MouseY);
            ButtonRight4_Light_Setting345(3);
            resetMouse(MouseX,MouseY);
            flag_right=3;
        }
        continue;
    }
    else if(mouse_press(Button3_x0,Button3_y0,Button3
_x1,Button3_y1)==1)
    {
        flag_right4_press_last=flag_right4_press;
        flag_right4_press=3;
        if(flag_right4_press_last!=0)
        {
            ButtonRight4_Darken_Setting345(flag_right
4_press_last);
        }
        ButtonRight4_Darken_Setting345(1);
        ButtonRight4_Darken_Setting345(2);
        ButtonRight4_Press_Setting345(3);
        Solid_Bar(429,468,1004,485,BLACK);
        Solid_Bar(1005,347,1013,482,BLACK);
        //hide mouse: button 3
        (*zone)=3;
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(flag_right!=0)
{
    if(flag_right!=flag_right4_press)
    {
        mousehide(MouseX,MouseY);
        ButtonRight4_Darken_Setting345(flag_right);
        resetMouse(MouseX,MouseY);
    }
    else
    {
        mousehide(MouseX,MouseY);
        ButtonRight4_Press_Setting345(flag_right);
        resetMouse(MouseX,MouseY);
    }
}

```

```

    }
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    flag_right=0;
}
if(shape!=0)
{
    shape=0;
}
}
//special button 5
if(MouseX>Weather_x0&&MouseX<Weather_x1&&MouseY>Weather_y
0&&MouseY<Weather_y1)
{
    if(mouse_press(Weather_x0,Weather_y0,Weather_x1,Weath
er_y1)==2)
    {
        shape=3;
        if(flag_left==0);
        {
            mousehide(MouseX,MouseY);
            ButtonLeft_Light_Setting(5);
            resetMouse(MouseX,MouseY);
            flag_left=5;
        }
        continue;
    }
    else if(mouse_press(Weather_x0,Weather_y0,Weather_x1,
Weather_y1)==1)
    {
        flag_press_last=flag_press;
        flag_press=5;
        if(flag_press_last!=0)
        {
            ButtonLeft_Darken_Setting(flag_press_last);
        }
        ButtonLeft_Press_Setting(5);
        //hide mouse
        Solid_Bar(Weather_x0,Weather_y1+1,Weather_x1,Poin
t_y1-1,BLACK);
    }
}

```

```

        Solid_Bar(Weather_x1+1,Weather_y0,Weather_x1+9,Weather_y1+15,BLACK);
        ButtonLeft_Darken_Setting(2);
        ButtonLeft_Darken_Setting(3);
        ButtonLeft_Darken_Setting(4);
        ButtonLeft_Darken_Setting(6);
        ButtonRight_Draw_Setting345(5);
        if((*weather)!=0)
        {
            ButtonRight5_Press_Setting345(*weather);
        }
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(flag_press==5)
{
    if(MouseX>Button1_x0&&MouseX<Button1_x1&&MouseY>Button1_y0&&MouseY<Button1_y1)
    {
        if(mouse_press(Button1_x0,Button1_y0,Button1_x1,Button1_y1)==2)
        {
            shape=3;
            if(flag_right==0);
            {
                mousehide(MouseX,MouseY);
                ButtonRight5_Light_Setting345(1);
                resetMouse(MouseX,MouseY);
                flag_right=1;
            }
            continue;
        }
        else if(mouse_press(Button1_x0,Button1_y0,Button1_x1,Button1_y1)==1)
        {
            flag_right5_press_last=flag_right5_press;
            flag_right5_press=1;
            if(flag_right5_press_last!=0)
            {
                ButtonRight5_Darken_Setting345(flag_right5_press_last);
            }
        }
    }
}

```

```

    }
    ButtonRight5_Darken_Setting345(2);
    ButtonRight5_Darken_Setting345(3);
    ButtonRight5_Press_Setting345(1);
    Solid_Bar(429,190,1004,207,BLACK);
    Solid_Bar(1005,70,1013,204,BLACK);
    //hide mouse: button 1
    (*weather)=1;
    resetMouse(MouseX,MouseY);
    delay(100);
    continue;
}
}
if(MouseX>Button2_x0&&MouseX<Button2_x1&&MouseY>Button2_y0&&MouseY<Button2_y1)
{
    if(mouse_press(Button2_x0,Button2_y0,Button2_x1,Button2_y1)==2)
    {
        shape=3;
        if(flag_right==0);
        {
            mousehide(MouseX,MouseY);
            ButtonRight5_Light_Setting345(2);
            resetMouse(MouseX,MouseY);
            flag_right=2;
        }
        continue;
    }
    else if(mouse_press(Button2_x0,Button2_y0,Button2_x1,Button2_y1)==1)
    {
        flag_right5_press_last=flag_right5_press;
        flag_right5_press=2;
        if(flag_right5_press_last!=0)
        {
            ButtonRight5_Darken_Setting345(flag_right5_press_last);
        }
        ButtonRight5_Darken_Setting345(1);
        ButtonRight5_Darken_Setting345(3);
        ButtonRight5_Press_Setting345(2);
        Solid_Bar(429,329,1004,346,BLACK);
    }
}

```



```

        Solid_Bar(1005,208,1013,343,BLACK);
        //hide mouse: button 2
        (*weather)=2;
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(MouseX>Button3_x0&&MouseX<Button3_x1&&MouseY>Button3_y0&&MouseY<Button3_y1)
{
    if(mouse_press(Button3_x0,Button3_y0,Button3_x1,Button3_y1)==2)
    {
        shape=3;
        if(flag_right==0);
        {
            mousehide(MouseX,MouseY);
            ButtonRight5_Light_Setting345(3);
            resetMouse(MouseX,MouseY);
            flag_right=3;
        }
        continue;
    }
    else if(mouse_press(Button3_x0,Button3_y0,Button3_x1,Button3_y1)==1)
    {
        flag_right5_press_last=flag_right5_press;
        flag_right5_press=3;
        if(flag_right5_press_last!=0)
        {
            ButtonRight5_Darken_Setting345(flag_right5_press_last);
        }
        ButtonRight5_Darken_Setting345(1);
        ButtonRight5_Darken_Setting345(2);
        ButtonRight5_Press_Setting345(3);
        Solid_Bar(429,468,1004,485,BLACK);
        Solid_Bar(1005,347,1013,482,BLACK);
        //hide mouse: button 3
        (*weather)=3;
        resetMouse(MouseX,MouseY);
        delay(100);
    }
}

```

```

        continue;
    }
}
if(flag_right!=0)
{
    if(flag_right!=flag_right5_press)
    {
        mousehide(MouseX,MouseY);
        ButtonRight5_Darken_Setting345(flag_right);
        resetMouse(MouseX,MouseY);
    }
    else
    {
        mousehide(MouseX,MouseY);
        ButtonRight5_Press_Setting345(flag_right);
        resetMouse(MouseX,MouseY);
    }
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    flag_right=0;
}
if(shape!=0)
{
    shape=0;
}
}
//special button 6
if(MouseX>Point_x0&&MouseX<Point_x1&&MouseY>Point_y0&&MouseY<Point_y1)
{
    if(mouse_press(Point_x0,Point_y0,Point_x1,Point_y1)==
2)
    {
        shape=3;
        if(flag_left==0);
        {
            mousehide(MouseX,MouseY);
            ButtonLeft_Light_Setting(6);
            resetMouse(MouseX,MouseY);
            flag_left=6;

```

```

        }
        continue;
    }
    else if(mouse_press(Point_x0,Point_y0,Point_x1,Point_
y1)==1)
    {
        flag_press_last=flag_press;
        flag_press=6;
        if(flag_press_last!=0)
        {
            ButtonLeft_Darken_Setting(flag_press_last);
        }
        ButtonLeft_Press_Setting(6);
        //hide mouse
        Solid_Bar(Point_x0,Point_y1+1,Point_x1,Point_y1+1
5,BLACK);
        Solid_Bar(Point_x1+1,Point_y0,Point_x1+9,Point_y1
+15,BLACK);
        ButtonLeft_Darken_Setting(2);
        ButtonLeft_Darken_Setting(3);
        ButtonLeft_Darken_Setting(4);
        ButtonLeft_Darken_Setting(5);
        //hide mouse
        ButtonRight_Draw_Setting6(point);
        resetMouse(MouseX,MouseY);
        delay(100);
        continue;
    }
}
if(flag_press==6)
{
    if(MouseX>Point_x0&&MouseX<PointMinus_x1&&MouseY>Poin
tMinus_y0&&MouseY<PointMinus_y1)
    {
        if(mouse_press(PointMinus_x0,PointMinus_y0,PointM
inus_x1,PointMinus_y1)==2)
        {
            shape=3;
            if(flag_point==0);
            {
                mousehide(MouseX,MouseY);
                ButtonPointNum_Light_Setting6(1);
                resetMouse(MouseX,MouseY);
            }
        }
    }
}

```

```

        flag_point=1;
    }
    continue;
}
else if(mouse_press(PointMinus_x0,PointMinus_y0,P
ointMinus_x1,PointMinus_y1)==1)
{
    shape=0;
    if((*point)!=2)
    {
        (*point)--;
        PointNum_Draw_Setting6((*point),WHITE);
        press=2;
        delay(300);
        //in order to prevent mouse is always cli
cking
    }
    ButtonPointNum_Darken_Setting6(2);
    ButtonPointNum_Darken_Setting6(3);
    continue;
}
}
if(MouseX>PointNum_x0&&MouseX<PointNum_x1&&MouseY>Poi
ntNum_y0&&MouseY<PointNum_y1)
{
    if(mouse_press(PointNum_x0,PointNum_y0,PointNum_x
1,PointNum_y1)==2)
    {
        shape=1;
        if(flag_point==0);
        {
            mousehide(MouseX,MouseY);
            ButtonPointNum_Light_Setting6(2);
            resetMouse(MouseX,MouseY);
            flag_point=2;
        }
        continue;
    }
}
if(MouseX>PointPlus_x0&&MouseX<PointPlus_x1&&MouseY>P
ointPlus_y0&&MouseY<PointPlus_y1)
{
    if(mouse_press(PointPlus_x0,PointPlus_y0,PointPlu

```

```

s_x1,PointPlus_y1)==2)
{
    shape=3;
    if(flag_point==0);
    {
        mousehide(MouseX,MouseY);
        ButtonPointNum_Light_Setting6(3);
        resetMouse(MouseX,MouseY);
        flag_point=3;
    }
    continue;
}
else if(mouse_press(PointPlus_x0,PointPlus_y0,Poi
ntPlus_x1,PointPlus_y1)==1)
{
    shape=0;
    if((*point)!=4)
    {
        (*point)++;
        PointNum_Draw_Setting6((*point),WHITE);
        press=2;
        delay(300);
        //in order to prevent mouse is always cli
cking
    }
    ButtonPointNum_Darken_Setting6(1);
    ButtonPointNum_Darken_Setting6(2);
    continue;
}
}
if(flag_point!=0)
{
    mousehide(MouseX,MouseY);
    ButtonPointNum_Darken_Setting6(flag_point);
    if(flag_mouse==1)
    {
        Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
        puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
    }
    resetMouse(MouseX,MouseY);
    flag_point=0;
}
if(shape!=0)

```

```

        {
            shape=0;
        }
    }
    //darken button support for special button 2-6
    if(flag_left!=0)
    {
        if(flag_left!=flag_press)
        {
            mousehide(MouseX,MouseY);
            ButtonLeft_Darken_Setting(flag_left);
            resetMouse(MouseX,MouseY);
        }
        else
        {
            mousehide(MouseX,MouseY);
            ButtonLeft_Press_Setting(flag_left);
            resetMouse(MouseX,MouseY);
        }
        if(flag_mouse==1)
        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        flag_left=0;
    }
    //basic button 7
    if(MouseX>Revert_x0&&MouseX<Revert_x1&&MouseY>Revert_y0&&
    MouseY<Revert_y1)
    {
        if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Revert_y
1)==2)
        {
            shape=3;
            if(flag_revert==0);
            {
                mousehide(MouseX,MouseY);
                ButtonLeft_Light_Setting(flag_revert);
                resetMouse(MouseX,MouseY);
                flag_revert=7;
            }
            continue;
        }
    }

```

```

        else if(mouse_press(Revert_x0,Revert_y0,Revert_x1,Revert_y1)==1)
        {
            *page = 0;
            //go to menu page
            delay(100);
            break;
        }
    }
    if(flag_revert!=0)
    {
        mousehide(MouseX,MouseY);
        ButtonLeft_Darken_Setting(flag_revert);
        if(flag_mouse==1)
        {
            Solid_Bar(xm1,ym1,xm2,ym2,DARK_GRAY);
            puthz(xm1+56,ym1+8,24,24,GRAY,"刷新");
        }
        resetMouse(MouseX,MouseY);
        flag_revert=0;
    }
    //reset mouse
    if(shape!=0)
    {
        shape=0;
    }
}
}

```

```

/*****
*****/

```

```

* 函数名称      PageLeft_Draw_Setting
* 函数作用      绘制分界线左边界面
* 函数输入      people 人数
* 函数输出      无

```

```

*****/
*****/

```

```

void PageLeft_Draw_Setting()
{
    //upperBar: 1024*50 DARK_GRAY + LIGHT_GRAY
    UpperBar_Draw(50,4,LIGHT_GRAY,DARK_GRAY);

    //BoundaryBar: 410-413 DARK_GRAY

```

```

Boundary_Draw(410,53,4,DARK_GRAY);

//button 1: 384*86 DARK_GRAY + LIGHT_GRAY
Button_Draw(Start_x0,Start_y0,Start_x1,Start_y1,Hollow_Color,
Solid_Color);
puthz(Start_x0+Text_xplus-
32,Start_y0+Text_yplus,32,32,BLACK,"开始模拟");

//button 2-3: 384*86 DARK_GRAY + LIGHT_GRAY
puthz(Text1_x0,Text1_y0,32,32,WHITE,"参赛人员设置");

Button_Draw(People_x0,People_y0,People_x1,People_y1,Hollow_Color,Solid_Color);
puthz(People_x0+Text_xplus,People_y0+Text_yplus,32,32,BLACK,"
人数");

Button_Draw(Age_x0,Age_y0,Age_x1,Age_y1,Hollow_Color,Solid_Color);
puthz(Age_x0+Text_xplus,Age_y0+Text_yplus,32,32,BLACK,"年龄
");

//button 4-6: 384*80 DARK_GRAY + LIGHT_GRAY
puthz(Text2_x0,Text2_y0,32,32,WHITE,"比赛场地设置");

Button_Draw(Zone_x0,Zone_y0,Zone_x1,Zone_y1,Hollow_Color,Solid_Color);
puthz(Zone_x0+Text_xplus,Zone_y0+Text_yplus,32,32,BLACK,"区域
");

Button_Draw(Weather_x0,Weather_y0,Weather_x1,Weather_y1,Hollow_Color,Solid_Color);
puthz(Weather_x0+Text_xplus,Weather_y0+Text_yplus,32,32,BLACK,"天气");

Button_Draw(Point_x0,Point_y0,Point_x1,Point_y1,Hollow_Color,Solid_Color);
puthz(Point_x0+Text_xplus,Point_y0+Text_yplus,32,32,BLACK,"点位");

//buttons 7: 40*40 BLACK + LIGHT_GRAY
RevertImage_Draw(25,25,BLACK);
puthz(50,9,32,32,BLACK,"模拟");
}

```



```

/*****
*****
* 函数名称      ButtonLeft_Light_Setting
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonLeft_Light_Setting(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Start_x0,Start_y0,Start_x1,Start_y1,WHITE);
            Solid_Bar(Start_x0,Start_y0,Start_x1,Start_y1,GREEN);
            puthz(Start_x0+Text_xplus-32,Start_y0+Text_yplus,32,32,GRAY,"开始模拟");
            break;
        }
        case 2:
        {
            Button_Edge(People_x0,People_y0,People_x1,People_y1,WHITE);
            Solid_Bar(People_x0,People_y0,People_x1,People_y1,GREEN);
            puthz(People_x0+Text_xplus,People_y0+Text_yplus,32,32,GRAY,"人数");
            break;
        }
        case 3:
        {
            Button_Edge(Age_x0,Age_y0,Age_x1,Age_y1,WHITE);
            Solid_Bar(Age_x0,Age_y0,Age_x1,Age_y1,GREEN);
            puthz(Age_x0+Text_xplus,Age_y0+Text_yplus,32,32,GRAY,"年龄");
            break;
        }
        case 4:
        {

```

```

        Button_Edge(Zone_x0,Zone_y0,Zone_x1,Zone_y1,WHITE
);
        Solid_Bar(Zone_x0,Zone_y0,Zone_x1,Zone_y1,GREEN);
        puthz(Zone_x0+Text_xplus,Zone_y0+Text_yplus,32,32
,GRAY,"区域");
        break;
    }
    case 5:
    {
        Button_Edge(Weather_x0,Weather_y0,Weather_x1,Weather_y1,WHITE);
        Solid_Bar(Weather_x0,Weather_y0,Weather_x1,Weather_y1,GREEN);
        puthz(Weather_x0+Text_xplus,Weather_y0+Text_yplus
,32,32,GRAY,"天气");
        break;
    }
    case 6:
    {
        Button_Edge(Point_x0,Point_y0,Point_x1,Point_y1,WHITE);
        Solid_Bar(Point_x0,Point_y0,Point_x1,Point_y1,GREEN);
        puthz(Point_x0+Text_xplus,Point_y0+Text_yplus,32,
32,GRAY,"点位");
        break;
    }
    case 7:
    {
        RevertButton_Draw(25,25,DARK_GRAY);
        RevertImage_Draw(25,25,WHITE);
        //revert button on
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      ButtonLeft_Darken_Setting
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无

```

```

*****
*****/
void ButtonLeft_Darken_Setting(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Start_x0,Start_y0,Start_x1,Start_y1,B
LACK);
            Button_Draw(Start_x0,Start_y0,Start_x1,Start_y1,H
ollow_Color,Solid_Color);
            puthz(Start_x0+Text_xplus-
32,Start_y0+Text_yplus,32,32,BLACK,"开始模拟");
            break;
        }
        case 2:
        {
            Button_Edge(People_x0,People_y0,People_x1,People_
y1,BLACK);
            Button_Draw(People_x0,People_y0,People_x1,People_
y1,Hollow_Color,Solid_Color);
            puthz(People_x0+Text_xplus,People_y0+Text_yplus,3
2,32,BLACK,"人数");
            break;
        }
        case 3:
        {
            Button_Edge(Age_x0,Age_y0,Age_x1,Age_y1,BLACK);
            Button_Draw(Age_x0,Age_y0,Age_x1,Age_y1,Hollow_Co
lor,Solid_Color);
            puthz(Age_x0+Text_xplus,Age_y0+Text_yplus,32,32,B
LACK,"年龄");
            break;
        }
        case 4:
        {
            Button_Edge(Zone_x0,Zone_y0,Zone_x1,Zone_y1,BLACK
);
            Button_Draw(Zone_x0,Zone_y0,Zone_x1,Zone_y1,Hollo
w_Color,Solid_Color);
            puthz(Zone_x0+Text_xplus,Zone_y0+Text_yplus,32,32
,BLACK,"区域");

```

```

        break;
    }
    case 5:
    {
        Button_Edge(Weather_x0,Weather_y0,Weather_x1,Weather_y1,BLACK);
        Button_Draw(Weather_x0,Weather_y0,Weather_x1,Weather_y1,Hollow_Color,Solid_Color);
        puthz(Weather_x0+Text_xplus,Weather_y0+Text_yplus,32,32,BLACK,"天气");
        break;
    }
    case 6:
    {
        Button_Edge(Point_x0,Point_y0,Point_x1,Point_y1,BLACK);
        Button_Draw(Point_x0,Point_y0,Point_x1,Point_y1,Hollow_Color,Solid_Color);
        puthz(Point_x0+Text_xplus,Point_y0+Text_yplus,32,32,BLACK,"点位");
        break;
    }
    case 7:
    {
        RevertButton_Draw(25,25,LIGHT_GRAY);
        RevertImage_Draw(25,25,BLACK);
        //revert button off
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      ButtonLeft_Press_Setting
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数注意      上方状态栏中心 x 值是 719，故上方状态栏右边文字左上角
坐标是(687,9)(2 个字)
* 函数输出      无
*****
*****/
void ButtonLeft_Press_Setting(int flag)

```

```

{
    Solid_Bar(687,9,751,41,LIGHT_GRAY);
    switch(flag)
    {
        case 1:
            {
                Button_Edge(Start_x0,Start_y0,Start_x1,Start_y1,BLACK
);
                Button_Draw(Start_x0,Start_y0,Start_x1,Start_y1,LIGHT
_GRAY,GRAY);
                puthz(Start_x0+Text_xplus-
32,Start_y0+Text_yplus,32,32,BLACK,"开始模拟");
                break;
            }
        case 2:
            {
                Button_Edge(People_x0,People_y0,People_x1,People_y1,B
LACK);
                Button_Draw(People_x0,People_y0,People_x1,People_y1,L
IGHT_GRAY,GRAY);
                puthz(People_x0+Text_xplus,People_y0+Text_yplus,32,32
,BLACK,"人数");
                puthz(687,9,32,32,BLACK,"人数");
                break;
            }
        case 3:
            {
                Button_Edge(Age_x0,Age_y0,Age_x1,Age_y1,BLACK);
                Button_Draw(Age_x0,Age_y0,Age_x1,Age_y1,LIGHT_GRAY,GR
AY);
                puthz(Age_x0+Text_xplus,Age_y0+Text_yplus,32,32,BLACK
,"年龄");
                puthz(687,9,32,32,BLACK,"年龄");
                break;
            }
        case 4:
            {
                Button_Edge(Zone_x0,Zone_y0,Zone_x1,Zone_y1,BLACK);
                Button_Draw(Zone_x0,Zone_y0,Zone_x1,Zone_y1,LIGHT_GRA
Y,GRAY);
                puthz(Zone_x0+Text_xplus,Zone_y0+Text_yplus,32,32,BLA
CK,"区域");
                puthz(687,9,32,32,BLACK,"区域");
            }
    }
}

```

```

        break;
    }
    case 5:
    {
        Button_Edge(Weather_x0,Weather_y0,Weather_x1,Weather_
y1,BLACK);
        Button_Draw(Weather_x0,Weather_y0,Weather_x1,Weather_
y1,LIGHT_GRAY,GRAY);
        puthz(Weather_x0+Text_xplus,Weather_y0+Text_yplus,32,
32,BLACK,"天气");
        puthz(687,9,32,32,BLACK,"天气");
        break;
    }
    case 6:
    {
        Button_Edge(Point_x0,Point_y0,Point_x1,Point_y1,BLACK
);
        Button_Draw(Point_x0,Point_y0,Point_x1,Point_y1,LIGHT
_GRAY,GRAY);
        puthz(Point_x0+Text_xplus,Point_y0+Text_yplus,32,32,B
LACK,"点位");
        puthz(687,9,32,32,BLACK,"点位");
        break;
    }
}
}
33. setting1 ,c
#include"headfile.h"

/*****
*****

* @author          ytm
* @date            2022-4-4
*****
*****/

//driver
#define NumMinus_x0      429
#define NumMinus_y0      70
#define Text_xplus       62
#define Text_yplus       27

//follower

```

```

#define NumMinus_x1      (NumMinus_x0+187)
#define NumMinus_y1      (NumMinus_y0+85)
#define PeopleNum_x0     (NumMinus_x0+193)
#define PeopleNum_y0     70
#define PeopleNum_x1     (NumMinus_x0+381)
#define PeopleNum_y1     (NumMinus_y0+85)
#define NumPlus_x0       (NumMinus_x0+387)
#define NumPlus_y0       70
#define NumPlus_x1       (NumMinus_x0+575)
#define NumPlus_y1       (NumMinus_y0+85)

/*****
*****
* 函数名称      ButtonRight_Draw_Setting1
* 函数作用      绘制分界线右边界面
* 函数输入      people 人数
* 函数注意      左上角坐标(414,54)
* 函数输出      无
*****
*****/
void ButtonRight_Draw_Setting1(int *people)
{
    //right page refresh
    Solid_Bar(414,54,1023,767,BLACK);

    //button 1: 188*86
    Button_Draw(NumMinus_x0,NumMinus_y0,NumMinus_x1,NumMinus_y1,Hollow_Color,Solid_Color);
    puthz(NumMinus_x0+Text_xplus,NumMinus_y0+Text_yplus,32,32,BLACK,"减少");

    //button 2: 188*86 display people number
    Button_Draw(PeopleNum_x0,PeopleNum_y0,PeopleNum_x1,PeopleNum_y1,GRAY,DARK_GRAY);
    PeopleNum_Draw_Setting1((*people),GRAY);

    //button 3: 188*86
    Button_Draw(NumPlus_x0,NumPlus_y0,NumPlus_x1,NumPlus_y1,Hollow_Color,Solid_Color);
    puthz(NumPlus_x0+Text_xplus,NumPlus_y0+Text_yplus,32,32,BLACK,"增加");
}

```

```

/*****
*****

* 函数名称      ButtonPeopleNum_Light_Setting1
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonPeopleNum_Light_Setting1(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(NumMinus_x0,NumMinus_y0,NumMinus_x1,NumMi
            nus_y1,WHITE);
            Solid_Bar(NumMinus_x0,NumMinus_y0,NumMinus_x1,NumMinu
            s_y1,GREEN);
            puthz(NumMinus_x0+Text_xplus,NumMinus_y0+Text_yplus,3
            2,32,GRAY,"减少");
            break;
        }
        case 2:
        {
            Button_Edge(PeopleNum_x0,PeopleNum_y0,PeopleNum_x1,Pe
            opleNum_y1,WHITE);
            break;
        }
        case 3:
        {
            Button_Edge(NumPlus_x0,NumPlus_y0,NumPlus_x1,NumPlus_
            y1,WHITE);
            Solid_Bar(NumPlus_x0,NumPlus_y0,NumPlus_x1,NumPlus_y1
            ,GREEN);
            puthz(NumPlus_x0+Text_xplus,NumPlus_y0+Text_yplus,32,
            32,GRAY,"增加");
            break;
        }
    }
}

/*****
*****

```



```

* 函数名称      ButtonPeopleNum_Darken_Setting1
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonPeopleNum_Darken_Setting1(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(NumMinus_x0,NumMinus_y0,NumMinus_x1,NumMi
nus_y1,BLACK);
            Button_Draw(NumMinus_x0,NumMinus_y0,NumMinus_x1,NumMi
nus_y1,Hollow_Color,Solid_Color);
            puthz(NumMinus_x0+Text_xplus,NumMinus_y0+Text_yplus,3
2,32,BLACK,"减少");
            break;
        }
        case 2:
        {
            Button_Edge(PeopleNum_x0,PeopleNum_y0,PeopleNum_x1,Pe
opleNum_y1,BLACK);
            break;
        }
        case 3:
        {
            Button_Edge(NumPlus_x0,NumPlus_y0,NumPlus_x1,NumPlus_
y1,BLACK);
            Button_Draw(NumPlus_x0,NumPlus_y0,NumPlus_x1,NumPlus_
y1,Hollow_Color,Solid_Color);
            puthz(NumPlus_x0+Text_xplus,NumPlus_y0+Text_yplus,32,
32,BLACK,"增加");
            break;
        }
    }
}

/*****
*****
* 函数名称      PeopleNum_Draw_Setting1
* 函数作用      显示人数

```

```

* 函数输入      people 人数, color 显示颜色
* 函数输出      无
*****
*****/
void PeopleNum_Draw_Setting1(int people,int color)
{
    //button refresh
    Solid_Bar(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus,Peo
pleNum_x0+126,PeopleNum_y0+59,DARK_GRAY);

    switch(people)
    {
        case 1:
        {
            puthz(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus
,32,32,color,"一人");
            break;
        }
        case 2:
        {
            puthz(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus
,32,32,color,"二人");
            break;
        }
        case 3:
        {
            puthz(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus
,32,32,color,"三人");
            break;
        }
        case 4:
        {
            puthz(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus
,32,32,color,"四人");
            break;
        }
        case 5:
        {
            puthz(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus
,32,32,color,"五人");
            break;
        }
        case 6:

```

```

        {
            puthz(PeopleNum_x0+Text_xplus,PeopleNum_y0+Text_yplus
,32,32,color,"六人");
            break;
        }
    }
}

```

34. setting2.c

```

#include"headfile.h"

/*****
*****

* @author          ytm
* @date            2022-4-4
*****
*****/

//driver
#define Button1_x0      429
#define Button1_y0      70

#define Title_xplus     15
#define Title_yplus     15
#define Detail1_xplus   15
#define Detail1_yplus   56
#define Detail2_xplus   15
#define Detail2_yplus   86

//follower
#define Button1_x1      (Button1_x0+575)
#define Button1_y1      (Button1_y0+119)
#define Button2_x0      429
#define Button2_y0      (Button1_y0+138)
#define Button2_x1      (Button1_x0+575)
#define Button2_y1      (Button1_y0+258)
#define Button3_x0      429
#define Button3_y0      (Button1_y0+277)
#define Button3_x1      (Button1_x0+575)
#define Button3_y1      (Button1_y0+397)
#define Button4_x0      429
#define Button4_y0      (Button1_y0+416)
#define Button4_x1      (Button1_x0+575)
#define Button4_y1      (Button1_y0+536)

```

```

/*****
*****
* 函数名称      ButtonRight_Draw_Setting345
* 函数作用      绘制分界线右边界面
* 函数输入      flag 绘制序号
* 函数注意      左下角坐标(414,54)
* 函数输出      无
*****
*****/
void ButtonRight_Draw_Setting345(int flag)
{
    //right page refresh
    Solid_Bar(414,54,1023,767,BLACK);

    switch(flag)
    {
        case 3:
            {
                //button 1-4: 576*130 DARK_GRAY + LIGHT_GRAY
                Button_Draw(Button1_x0,Button1_y0,Button1_x1,Button1_y1,Hollow_Color,Solid_Color);
                puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,BLACK,"少年");
                puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快，体质不差，体力恢复不");
                puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"快，可以支撑适量距离的跑步");

                Button_Draw(Button2_x0,Button2_y0,Button2_x1,Button2_y1,Hollow_Color,Solid_Color);
                puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,BLACK,"青年");
                puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度很快，体质很好，体力恢复很");
                puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2_yplus,24,24,DARK_GRAY,"快，可以支撑较远距离的跑步");

                Button_Draw(Button3_x0,Button3_y0,Button3_x1,Button3_y1,Hollow_Color,Solid_Color);
                puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,BLACK,"中年");
                puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度很快，体质很好，体力恢复很");
                puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2_yplus,24,24,DARK_GRAY,"快，可以支撑较远距离的跑步");
            }
    }
}

```

```

us,32,32,BLACK,"中年");
    puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1
_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度中等，体质中等，体力恢复
中");
    puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2
_yplus,24,24,DARK_GRAY,"等，可以支撑中等距离的跑步");

    Button_Draw(Button4_x0,Button4_y0,Button4_x1,Butt
on4_y1,Hollow_Color,Solid_Color);
    puthz(Button4_x0+Title_xplus,Button4_y0+Title_ypl
us,32,32,BLACK,"老年");
    puthz(Button4_x0+Detail1_xplus,Button4_y0+Detail1
_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快，体质较差，体力恢复
较");
    puthz(Button4_x0+Detail2_xplus,Button4_y0+Detail2
_yplus,24,24,DARK_GRAY,"慢，但仍可以支撑少量距离的跑步");
    break;
}
case 4:
{
    //button 1-3: 576*130 DARK_GRAY + LIGHT_GRAY
    Button_Draw(Button1_x0,Button1_y0,Button1_x1,Butt
on1_y1,Hollow_Color,Solid_Color);
    puthz(Button1_x0+Title_xplus,Button1_y0+Title_ypl
us,32,32,BLACK,"城区");
    puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1
_yplus,24,24,DARK_GRAY,"以公路和建筑为主，还有不少的树木，有时会有河
流");
    puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2
_yplus,24,24,DARK_GRAY,"和湖泊");

    Button_Draw(Button2_x0,Button2_y0,Button2_x1,Butt
on2_y1,Hollow_Color,Solid_Color);
    puthz(Button2_x0+Title_xplus,Button2_y0+Title_ypl
us,32,32,BLACK,"郊区");
    puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1
_yplus,24,23,DARK_GRAY,"以草地和树木为主，还有不少的田地、小径和建筑
物，");
    puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2
_yplus,24,24,DARK_GRAY,"有时会有河流和公路");

    Button_Draw(Button3_x0,Button3_y0,Button3_x1,Butt
on3_y1,Hollow_Color,Solid_Color);

```

```

        puthz(Button3_x0+Title_xplus,Button3_y0+Title_ypl
us,32,32,BLACK,"山区");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1
_yplus,24,24,DARK_GRAY,"以山、树木和草地为主，还有不少的河流和湖泊，
有");
        puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2
_yplus,24,24,DARK_GRAY,"时会有沼泽和小屋");
        break;
    }
    case 5:
    {
        //button 1-3: 576*130 DARK_GRAY + LIGHT_GRAY
        Button_Draw(Button1_x0,Button1_y0,Button1_x1,Butt
on1_y1,Hollow_Color,Solid_Color);
        puthz(Button1_x0+Title_xplus,Button1_y0+Title_ypl
us,32,32,BLACK,"晴天");
        puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1
_yplus,24,24,DARK_GRAY,"天气较为炎热，走路速度减少五分之一跑步、爬
山，");
        puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2
_yplus,24,24,DARK_GRAY,"速度减半");

        Button_Draw(Button2_x0,Button2_y0,Button2_x1,Butt
on2_y1,Hollow_Color,Solid_Color);
        puthz(Button2_x0+Title_xplus,Button2_y0+Title_ypl
us,32,32,BLACK,"多云");
        puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1
_yplus,24,24,DARK_GRAY,"天气较为舒适，走路、跑步、爬山速度不受影响
");

        Button_Draw(Button3_x0,Button3_y0,Button3_x1,Butt
on3_y1,Hollow_Color,Solid_Color);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_ypl
us,32,32,BLACK,"雨天");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1
_yplus,24,24,DARK_GRAY,"天气较为糟糕，走路、跑步、爬山速度减少二分之
一");
        break;
    }
}
}

```

```

/*****

```

```

*****
* 函数名称      ButtonRight3_Light_Setting345
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonRight3_Light_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,WHITE);
            Solid_Bar(Button1_x0,Button1_y0,Button1_x1,Button1_y1,GREEN);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,WHITE,"少年");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快，体质不差，体力恢复不");
            puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"快，可以支撑适量距离的跑步");
            break;
        }
        case 2:
        {
            Button_Edge(Button2_x0,Button2_y0,Button2_x1,Button2_y1,WHITE);
            Solid_Bar(Button2_x0,Button2_y0,Button2_x1,Button2_y1,GREEN);
            puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,WHITE,"青年");
            puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度很快，体质很好，体力恢复很");
            puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2_yplus,24,24,DARK_GRAY,"快，可以支撑较远距离的跑步");
            break;
        }
        case 3:
        {

```

```

        Button_Edge(Button3_x0,Button3_y0,Button3_x1,Button3_y1,WHITE);
        Solid_Bar(Button3_x0,Button3_y0,Button3_x1,Button3_y1,GREEN);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,WHITE,"中年");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度中等，体质中等，体力恢复中");
        puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2_yplus,24,24,DARK_GRAY,"等，可以支撑中等距离的跑步");
        break;
    }
    case 4:
    {
        Button_Edge(Button4_x0,Button4_y0,Button4_x1,Button4_y1,WHITE);
        Solid_Bar(Button4_x0,Button4_y0,Button4_x1,Button4_y1,GREEN);
        puthz(Button4_x0+Title_xplus,Button4_y0+Title_yplus,32,32,WHITE,"老年");
        puthz(Button4_x0+Detail1_xplus,Button4_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快，体质较差，体力恢复较");
        puthz(Button4_x0+Detail2_xplus,Button4_y0+Detail2_yplus,24,24,DARK_GRAY,"慢，但仍可以支撑少量距离的跑步");
        break;
    }
}
}
}

```

```

/*****
*****
* 函数名称      ButtonRight3_Darken_Setting345
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void ButtonRight3_Darken_Setting345(int flag)
{
    switch(flag)
    {

```



```

        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,BLACK);
            Button_Draw(Button1_x0,Button1_y0,Button1_x1,Button1_y1,Hollow_Color,Solid_Color);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,BLACK,"少年");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快, 体质不差, 体力恢复不");
            puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"快, 可以支撑适量距离的跑步");
            break;
        }
        case 2:
        {
            Button_Edge(Button2_x0,Button2_y0,Button2_x1,Button2_y1,BLACK);
            Button_Draw(Button2_x0,Button2_y0,Button2_x1,Button2_y1,Hollow_Color,Solid_Color);
            puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,BLACK,"青年");
            puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度很快, 体质很好, 体力恢复很");
            puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2_yplus,24,24,DARK_GRAY,"快, 可以支撑较远距离的跑步");
            break;
        }
        case 3:
        {
            Button_Edge(Button3_x0,Button3_y0,Button3_x1,Button3_y1,BLACK);
            Button_Draw(Button3_x0,Button3_y0,Button3_x1,Button3_y1,Hollow_Color,Solid_Color);
            puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,BLACK,"中年");
            puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度中等, 体质中等, 体力恢复中");
            puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2_yplus,24,24,DARK_GRAY,"等, 可以支撑中等距离的跑步");

```

```

        break;
    }
    case 4:
    {
        Button_Edge(Button4_x0,Button4_y0,Button4_x1,Button4_y1,BLACK);
        Button_Draw(Button4_x0,Button4_y0,Button4_x1,Button4_y1,Hollow_Color,Solid_Color);
        puthz(Button4_x0+Title_xplus,Button4_y0+Title_yplus,32,32,BLACK,"老年");
        puthz(Button4_x0+Detail1_xplus,Button4_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快, 体质较差, 体力恢复较");
        puthz(Button4_x0+Detail2_xplus,Button4_y0+Detail2_yplus,24,24,DARK_GRAY,"慢, 但仍可以支撑少量距离的跑步");
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      ButtonRight3_Press_Setting345
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void ButtonRight3_Press_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,BLACK);
            Button_Draw(Button1_x0,Button1_y0,Button1_x1,Button1_y1,LIGHT_GRAY,GRAY);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,BLACK,"少年");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快, 体质不差, 体力恢复不");

```

```

        puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2
        _yplus,24,24,DARK_GRAY,"快，可以支撑适量距离的跑步");
        break;
    }
    case 2:
    {
        Button_Edge(Button2_x0,Button2_y0,Button2_x1,Butt
        on2_y1,BLACK);
        Button_Draw(Button2_x0,Button2_y0,Button2_x1,Butt
        on2_y1,LIGHT_GRAY,GRAY);
        puthz(Button2_x0+Title_xplus,Button2_y0+Title_ypl
        us,32,32,BLACK,"青年");
        puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1
        _yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度很快，体质很好，体力恢复
        很");
        puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2
        _yplus,24,24,DARK_GRAY,"快，可以支撑较远距离的跑步");
        break;
    }
    case 3:
    {
        Button_Edge(Button3_x0,Button3_y0,Button3_x1,Butt
        on3_y1,BLACK);
        Button_Draw(Button3_x0,Button3_y0,Button3_x1,Butt
        on3_y1,LIGHT_GRAY,GRAY);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_ypl
        us,32,32,BLACK,"中年");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1
        _yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度中等，体质中等，体力恢复
        中");
        puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2
        _yplus,24,24,DARK_GRAY,"等，可以支撑中等距离的跑步");
        break;
    }
    case 4:
    {
        Button_Edge(Button4_x0,Button4_y0,Button4_x1,Butt
        on4_y1,BLACK);
        Button_Draw(Button4_x0,Button4_y0,Button4_x1,Butt
        on4_y1,LIGHT_GRAY,GRAY);
        puthz(Button4_x0+Title_xplus,Button4_y0+Title_ypl
        us,32,32,BLACK,"老年");
        puthz(Button4_x0+Detail1_xplus,Button4_y0+Detail1

```

```

_yplus,24,24,DARK_GRAY,"走路、跑步、爬山速度不快, 体质较差, 体力恢复
较");
        puthz(Button4_x0+Detail2_xplus,Button4_y0+Detail2
_yplus,24,24,DARK_GRAY,"慢, 但仍可以支撑少量距离的跑步");
        break;
    }
}
}

/*****
*****
* 函数名称      ButtonRight4_Light_Setting345
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonRight4_Light_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Butt
on1_y1,WHITE);
            Solid_Bar(Button1_x0,Button1_y0,Button1_x1,Button
1_y1,GREEN);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_ypl
us,32,32,WHITE,"城区");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1
_yplus,24,24,DARK_GRAY,"以公路和建筑为主, 还有不少的树木, 有时会有河
流");
            puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2
_yplus,24,24,DARK_GRAY,"和湖泊");
            break;
        }
        case 2:
        {
            Button_Edge(Button2_x0,Button2_y0,Button2_x1,Butt
on2_y1,WHITE);
            Solid_Bar(Button2_x0,Button2_y0,Button2_x1,Button
2_y1,GREEN);
            puthz(Button2_x0+Title_xplus,Button2_y0+Title_ypl

```

```

us,32,32,WHITE,"郊区");
        puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1
_yplus,24,23,DARK_GRAY,"以草地和树木为主，还有不少的田地、小径和建筑
物，");
        puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2
_yplus,24,24,DARK_GRAY,"有时会有河流和公路");
        break;
    }
    case 3:
    {
        Button_Edge(Button3_x0,Button3_y0,Button3_x1,Butt
on3_y1,WHITE);
        Solid_Bar(Button3_x0,Button3_y0,Button3_x1,Button
3_y1,GREEN);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_ypl
us,32,32,WHITE,"山区");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1
_yplus,24,24,DARK_GRAY,"以山、树木和草地为主，还有不少的河流和湖泊，
有");
        puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2
_yplus,24,24,DARK_GRAY,"时会有沼泽和小屋");
        break;
    }
}
}
}

```

```

/*****
*****
* 函数名称      ButtonRight4_Darken_Setting345
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void ButtonRight4_Darken_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Butt
on1_y1,BLACK);
            Button_Draw(Button1_x0,Button1_y0,Button1_x1,Butt

```

```

on1_y1,Hollow_Color,Solid_Color);
        puthz(Button1_x0+Title_xplus,Button1_y0+Title_ypl
us,32,32,BLACK,"城区");
        puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1
_yplus,24,24,DARK_GRAY,"以公路和建筑为主，还有不少的树木，有时会有河
流");
        puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2
_yplus,24,24,DARK_GRAY,"和湖泊");
        break;
    }
    case 2:
    {
        Button_Edge(Button2_x0,Button2_y0,Button2_x1,Butt
on2_y1,BLACK);
        Button_Draw(Button2_x0,Button2_y0,Button2_x1,Butt
on2_y1,Hollow_Color,Solid_Color);
        puthz(Button2_x0+Title_xplus,Button2_y0+Title_ypl
us,32,32,BLACK,"郊区");
        puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1
_yplus,24,23,DARK_GRAY,"以草地和树木为主，还有不少的田地、小径和建筑
物，");
        puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2
_yplus,24,24,DARK_GRAY,"有时会有河流和公路");
        break;
    }
    case 3:
    {
        Button_Edge(Button3_x0,Button3_y0,Button3_x1,Butt
on3_y1,BLACK);
        Button_Draw(Button3_x0,Button3_y0,Button3_x1,Butt
on3_y1,Hollow_Color,Solid_Color);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_ypl
us,32,32,BLACK,"山区");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1
_yplus,24,24,DARK_GRAY,"以山、树木和草地为主，还有不少的河流和湖泊，
有");
        puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2
_yplus,24,24,DARK_GRAY,"时会有沼泽和小屋");
        break;
    }
}
}

```

```

/*****
*****

* 函数名称      ButtonRight4_Press_Setting345
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonRight4_Press_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,BLACK);
            Button_Draw(Button1_x0,Button1_y0,Button1_x1,Button1_y1,LIGHT_GRAY,GRAY);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,BLACK,"城区");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"以公路和建筑为主，还有不少的树木，有时会有河流");
            puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"和湖泊");
            break;
        }
        case 2:
        {
            Button_Edge(Button2_x0,Button2_y0,Button2_x1,Button2_y1,BLACK);
            Button_Draw(Button2_x0,Button2_y0,Button2_x1,Button2_y1,LIGHT_GRAY,GRAY);
            puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,BLACK,"郊区");
            puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,23,DARK_GRAY,"以草地和树木为主，还有不少的田地、小径和建筑物，");
            puthz(Button2_x0+Detail2_xplus,Button2_y0+Detail2_yplus,24,24,DARK_GRAY,"有时会有河流和公路");
            break;
        }
        case 3:

```

```

        {
            Button_Edge(Button3_x0,Button3_y0,Button3_x1,Button3_y1,BLACK);
            Button_Draw(Button3_x0,Button3_y0,Button3_x1,Button3_y1,LIGHT_GRAY,GRAY);
            puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,BLACK,"山区");
            puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"以山、树木和草地为主，还有不少的河流和湖泊，有");
            puthz(Button3_x0+Detail2_xplus,Button3_y0+Detail2_yplus,24,24,DARK_GRAY,"时会有沼泽和小屋");
            break;
        }
    }
}

```

```

/*****
*****
* 函数名称      ButtonRight5_Light_Setting345
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void ButtonRight5_Light_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,WHITE);
            Solid_Bar(Button1_x0,Button1_y0,Button1_x1,Button1_y1,GREEN);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,WHITE,"晴天");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为炎热，走路速度减少四分之一，跑步、爬山");
            puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"速度减半");
            break;
        }
    }
}

```



```

        }
        case 2:
        {
            Button_Edge(Button2_x0,Button2_y0,Button2_x1,Button2_y1,WHITE);
            Solid_Bar(Button2_x0,Button2_y0,Button2_x1,Button2_y1,GREEN);
            puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,WHITE,"多云");
            puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为舒适，走路、跑步、爬山速度不受影响");
            break;
        }
        case 3:
        {
            Button_Edge(Button3_x0,Button3_y0,Button3_x1,Button3_y1,WHITE);
            Solid_Bar(Button3_x0,Button3_y0,Button3_x1,Button3_y1,GREEN);
            puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,WHITE,"雨天");
            puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为糟糕，走路、跑步、爬山速度减少二分之一");
            break;
        }
    }
}

```

```

/*****
*****

```

```

* 函数名称      ButtonRight5_Darken_Setting345
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无

```

```

*****
*****/

```

```

void ButtonRight5_Darken_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {

```

```

        Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,BLACK);
        Button_Draw(Button1_x0,Button1_y0,Button1_x1,Button1_y1,Hollow_Color,Solid_Color);
        puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,BLACK,"晴天");
        puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为炎热，走路速度减少四分之一，跑步、爬山");
        puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"速度减半");
        break;
    }
    case 2:
    {
        Button_Edge(Button2_x0,Button2_y0,Button2_x1,Button2_y1,BLACK);
        Button_Draw(Button2_x0,Button2_y0,Button2_x1,Button2_y1,Hollow_Color,Solid_Color);
        puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,BLACK,"多云");
        puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为舒适，走路、跑步、爬山速度不受影响");
        break;
    }
    case 3:
    {
        Button_Edge(Button3_x0,Button3_y0,Button3_x1,Button3_y1,BLACK);
        Button_Draw(Button3_x0,Button3_y0,Button3_x1,Button3_y1,Hollow_Color,Solid_Color);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,BLACK,"雨天");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为糟糕，走路、跑步、爬山速度减少二分之一");
        break;
    }
}
}

```

```

/*****

```

```

*****
* 函数名称      ButtonRight5_Press_Setting345
* 函数作用      按下按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonRight5_Press_Setting345(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(Button1_x0,Button1_y0,Button1_x1,Button1_y1,BLACK);
            Button_Draw(Button1_x0,Button1_y0,Button1_x1,Button1_y1,LIGHT_GRAY,GRAY);
            puthz(Button1_x0+Title_xplus,Button1_y0+Title_yplus,32,32,BLACK,"晴天");
            puthz(Button1_x0+Detail1_xplus,Button1_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为炎热，走路速度减少四分之一，跑步、爬山");
            puthz(Button1_x0+Detail2_xplus,Button1_y0+Detail2_yplus,24,24,DARK_GRAY,"速度减半");
            break;
        }
        case 2:
        {
            Button_Edge(Button2_x0,Button2_y0,Button2_x1,Button2_y1,BLACK);
            Button_Draw(Button2_x0,Button2_y0,Button2_x1,Button2_y1,LIGHT_GRAY,GRAY);
            puthz(Button2_x0+Title_xplus,Button2_y0+Title_yplus,32,32,BLACK,"多云");
            puthz(Button2_x0+Detail1_xplus,Button2_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为舒适，走路、跑步、爬山速度不受影响");
            break;
        }
        case 3:
        {
            Button_Edge(Button3_x0,Button3_y0,Button3_x1,Button3_y1,BLACK);

```

```

        Button_Draw(Button3_x0,Button3_y0,Button3_x1,Button3_y1,LIGHT_GRAY,GRAY);
        puthz(Button3_x0+Title_xplus,Button3_y0+Title_yplus,32,32,BLACK,"雨天");
        puthz(Button3_x0+Detail1_xplus,Button3_y0+Detail1_yplus,24,24,DARK_GRAY,"天气较为糟糕，走路、跑步、爬山速度减少二分之一");
        break;
    }
}

```

35. setting3.c

```
#include"headfile.h"
```

```

/*****
*****

* @author          ytm
* @date            2022-4-4
*****
*****/

//driver
#define PointMinus_x0      429
#define PointMinus_y0      70
#define Text_xplus        62
#define Text_yplus        27

//follower
#define PointMinus_x1      (PointMinus_x0+187)
#define PointMinus_y1      (PointMinus_y0+85)
#define PointNum_x0        (PointMinus_x0+193)
#define PointNum_y0        70
#define PointNum_x1        (PointMinus_x0+381)
#define PointNum_y1        (PointMinus_y0+85)
#define PointPlus_x0        (PointMinus_x0+387)
#define PointPlus_y0        70
#define PointPlus_x1        (PointMinus_x0+575)
#define PointPlus_y1        (PointMinus_y0+85)

/*****
*****

* 函数名称          ButtonRight_Draw_Setting6
* 函数作用          绘制分界线右边界面

```

```

* 函数输入      point 点位
* 函数注意      左上角坐标(414,54)
* 函数输出      无
*****
*****/
void ButtonRight_Draw_Setting6(int *point)
{
    //right page refresh
    Solid_Bar(414,54,1023,767,BLACK);

    //button 1: 188*86
    Button_Draw(PointMinus_x0,PointMinus_y0,PointMinus_x1,PointMinus_y1,Hollow_Color,Solid_Color);
    puthz(PointMinus_x0+Text_xplus,PointMinus_y0+Text_yplus,32,32,BLACK,"减少");

    //button 2: 188*86 display Point number
    Button_Draw(PointNum_x0,PointNum_y0,PointNum_x1,PointNum_y1,GRAY,DARK_GRAY);
    PointNum_Draw_Setting6((*point),GRAY);

    //button 3: 188*86
    Button_Draw(PointPlus_x0,PointPlus_y0,PointPlus_x1,PointPlus_y1,Hollow_Color,Solid_Color);
    puthz(PointPlus_x0+Text_xplus,PointPlus_y0+Text_yplus,32,32,BLACK,"增加");
}

/*****
*****
* 函数名称      ButtonPointNum_Light_Setting6
* 函数作用      点亮按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/
void ButtonPointNum_Light_Setting6(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(PointMinus_x0,PointMinus_y0,PointMinus_x1

```

```

,PointMinus_y1,WHITE);
    Solid_Bar(PointMinus_x0,PointMinus_y0,PointMinus_x1,PointMinus_y1,GREEN);
    puthz(PointMinus_x0+Text_xplus,PointMinus_y0+Text_yplus,32,32,GRAY,"减少");
    break;
}
case 2:
{
    Button_Edge(PointNum_x0,PointNum_y0,PointNum_x1,PointNum_y1,WHITE);
    break;
}
case 3:
{
    Button_Edge(PointPlus_x0,PointPlus_y0,PointPlus_x1,PointPlus_y1,WHITE);
    Solid_Bar(PointPlus_x0,PointPlus_y0,PointPlus_x1,PointPlus_y1,GREEN);
    puthz(PointPlus_x0+Text_xplus,PointPlus_y0+Text_yplus,32,32,GRAY,"增加");
    break;
}
}
}

```

```

/*****
*****
* 函数名称      ButtonPointNum_Darken_Setting6
* 函数作用      熄灭按钮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void ButtonPointNum_Darken_Setting6(int flag)
{
    switch(flag)
    {
    case 1:
    {
        Button_Edge(PointMinus_x0,PointMinus_y0,PointMinus_x1,PointMinus_y1,BLACK);
        Button_Draw(PointMinus_x0,PointMinus_y0,PointMinus_x1

```

```
,PointMinus_y1,Hollow_Color,Solid_Color);
    puthz(PointMinus_x0+Text_xplus,PointMinus_y0+Text_yplus,32,32,BLACK,"减少");
    break;
}
case 2:
{
    Button_Edge(PointNum_x0,PointNum_y0,PointNum_x1,PointNum_y1,BLACK);
    break;
}
case 3:
{
    Button_Edge(PointPlus_x0,PointPlus_y0,PointPlus_x1,PointPlus_y1,BLACK);
    Button_Draw(PointPlus_x0,PointPlus_y0,PointPlus_x1,PointPlus_y1,Hollow_Color,Solid_Color);
    puthz(PointPlus_x0+Text_xplus,PointPlus_y0+Text_yplus,32,32,BLACK,"增加");
    break;
}
}
}
```

```

/*****
*****
* 函数名称      PointNum_Draw_Setting6
* 函数作用      显示点位数量
* 函数输入      point 点位数量, color 显示颜色
* 函数输出      无
*****
*****/

```

```
void PointNum_Draw_Setting6(int point,int color)
{
    //button refresh
    Solid_Bar(PointNum_x0+Text_xplus,PointNum_y0+Text_yplus,PointNum_x0+126,PointNum_y0+59,DARK_GRAY);

    switch(point)
    {
    case 2:
        {
            puthz(PointNum_x0+Text_xplus,PointNum_y0+Text_yplus,3
```

```

2,32,color,"二个");
        break;
    }
    case 3:
    {
        puthz(PointNum_x0+Text_xplus,PointNum_y0+Text_yplus,3
2,32,color,"三个");
        break;
    }
    case 4:
    {
        puthz(PointNum_x0+Text_xplus,PointNum_y0+Text_yplus,3
2,32,color,"四个");
        break;
    }
}
}

```

36. sign.c

```

#include"headfile.h"

/*****
*****

* @author          ytm
* @date            2022-4-15
*****
*****/

//driver
#define Text_xplus    142
#define Text_yplus    4

//follower
#define City_x0        x0+2
#define City_y0        (y0+2)
#define City_x1        x1-2
#define City_y1        (y0+41)
#define Suburb_x0      x0+2
#define Suburb_y0      (y0+42)
#define Suburb_x1      x1-2
#define Suburb_y1      (y0+81)
#define Mountain_x0    x0+2
#define Mountain_y0    (y0+82)
#define Mountain_x1    x1-2

```



```

#define Mountain_y1    (y0+121)
#define Animation_x0    x0+2
#define Animation_y0    (y0+122)
#define Animation_x1    x1-2
#define Animation_y1    (y0+161)

/*****
*****
* 函数名称      ChooseButtonSign
* 函数作用      选择显示图例地区
* 函数输入      x0, y0 按钮绘制左上角, x1, y1 按钮绘制右下角,
flag_choose_sign 选择地区类型地址
* 函数输出      无
*****
*****/
void ChooseButtonSign(int x0,int y0,int x1,int y1,int *flag_choose_sign)
{
    int flag_sign=0;
    ChooseButtonSign_Draw(x0,y0,x1,y1,(*flag_choose_sign));
    resetMouse(MouseX,MouseY);
    while(1)
    {
        newxy(&MouseX,&MouseY,&press);
        //choose button 1-3 (x1-x0+1)*32
        if(MouseX>City_x0&&MouseX<City_x1&&MouseY>City_y0&&MouseY
<City_y1)
        {
            if(mouse_press(City_x0,City_y0,City_x1,City_y1)==2)
            {
                shape=3;
                if(flag_sign==0);
                {
                    mousehide(MouseX,MouseY);
                    ChooseButton_Light_Sign(x0,y0,x1,y1,1);
                    resetMouse(MouseX,MouseY);
                    flag_sign=1;
                }
                continue;
            }
            else if(mouse_press(City_x0,City_y0,City_x1,City_y1)=
=1)
            {

```

```

        (*flag_choose_sign)=1;
        resetMouse(MouseX,MouseY);
        delay(100);
        break;
    }
}
if(MouseX>Suburb_x0&&MouseX<Suburb_x1&&MouseY>Suburb_y0&&
MouseY<Suburb_y1)
{
    if(mouse_press(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y
1)==2)
    {
        shape=3;
        if(flag_sign==0);
        {
            mousehide(MouseX,MouseY);
            ChooseButton_Light_Sign(x0,y0,x1,y1,2);
            resetMouse(MouseX,MouseY);
            flag_sign=2;
        }
        continue;
    }
    else if(mouse_press(Suburb_x0,Suburb_y0,Suburb_x1,Sub
urb_y1)==1)
    {
        (*flag_choose_sign)=2;
        resetMouse(MouseX,MouseY);
        delay(100);
        break;
    }
}
if(MouseX>Mountain_x0&&MouseX<Mountain_x1&&MouseY>Mountai
n_y0&&MouseY<Mountain_y1)
{
    if(mouse_press(Mountain_x0,Mountain_y0,Mountain_x1,Mo
untain_y1)==2)
    {
        shape=3;
        if(flag_sign==0);
        {
            mousehide(MouseX,MouseY);
            ChooseButton_Light_Sign(x0,y0,x1,y1,3);
            resetMouse(MouseX,MouseY);

```

```

        flag_sign=3;
    }
    continue;
}
else if(mouse_press(Mountain_x0,Mountain_y0,Mountain_
x1,Mountain_y1)==1)
{
    (*flag_choose_sign)=3;
    resetMouse(MouseX,MouseY);
    delay(100);
    break;
}
}
if(MouseX>Animation_x0&&MouseX<Animation_x1&&MouseY>Anima
tion_y0&&MouseY<Animation_y1)
{
    if(mouse_press(Animation_x0,Animation_y0,Animation_x1
,Animation_y1)==2)
    {
        shape=3;
        if(flag_sign==0);
        {
            mousehide(MouseX,MouseY);
            ChooseButton_Light_Sign(x0,y0,x1,y1,4);
            resetMouse(MouseX,MouseY);
            flag_sign=4;
        }
        continue;
    }
    else if(mouse_press(Animation_x0,Animation_y0,Animati
on_x1,Animation_y1)==1)
    {
        (*flag_choose_sign)=4;
        //hide mouse
        Solid_Bar(12,408,395,422,BLACK);
        resetMouse(MouseX,MouseY);
        delay(100);
        break;
    }
}
if(flag_sign!=0)
{
    if(flag_sign!=(*flag_choose_sign))

```

```

        {
            mousehide(MouseX,MouseY);
            ChooseButton_Darken_Sign(x0,y0,x1,y1,flag_sign,(*
flag_choose_sign));
            resetMouse(MouseX,MouseY);
        }
        else
        {
            mousehide(MouseX,MouseY);
            ChooseButton_Press_Sign(x0,y0,x1,y1,flag_sign);
            resetMouse(MouseX,MouseY);
        }
        flag_sign=0;
    }
    //reset mouse
    if(shape!=0)
    {
        shape=0;
    }
}

}

/*****
*****
* 函数名称      ChooseButtonSign_Draw
* 函数作用      绘制按钮
* 函数输入      x0, y0 按钮绘制左上角, x1, y1 按钮绘制右下角,
flag_choose_sign 选择地区类型
* 函数输出      无
*****
*****/
void ChooseButtonSign_Draw(int x0,int y0,int x1,int y1,int flag_choose_sign)
{
    //draw general button
    Button_Draw(x0,y0,x1,y0+161,LIGHT_GRAY,DARK_GRAY);

    //Solid_Bar(City_x0,City_y0,City_x1,City_y1,DARK_GRAY);
    puthz(City_x0+Text_xplus,City_y0+Text_yplus,24,24,WHITE,"城区图例");

    //Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y1,DARK_GRAY);
}

```

```
    puthz(Suburb_x0+Text_xplus,Suburb_y0+Text_yplus,24,24,WHITE,"
郊区图例");
```

```
    //Solid_Bar(Mountain_x0,Mountain_y0,Mountain_x1,Mountain_y1,D
ARK_GRAY);
```

```
    puthz(Mountain_x0+Text_xplus,Mountain_y0+Text_yplus,24,24,WHI
TE,"山区图例");
```

```
    //Solid_Bar(Animation_x0,Animation_y0,Animation_x1,Animation_
y1,DARK_GRAY);
```

```
    puthz(Animation_x0+Text_xplus,Animation_y0+Text_yplus,24,24,W
HITE,"人物动画");
```

```
    if(flag_choose_sign!=0)
    {
        ChooseButton_Press_Sign(x0,y0,x1,y1,flag_choose_sign);
    }
}
```

```

/*****
*****
```

```
* 函数名称      ChooseButton_Light_Sign
* 函数作用      点亮按钮
* 函数输入      x0, y0 按钮绘制左上角, x1, y1 按钮绘制右下角, flag
按钮序号
* 函数输出      无
```

```
*****
*****/
```

```
void ChooseButton_Light_Sign(int x0,int y0,int x1,int y1,int flag
)
{
```

```
    switch(flag)
    {
    case 1:
        {
            Button_Edge(City_x0,City_y0,City_x1,City_y1,WHITE);
            Solid_Bar(City_x0,City_y0,City_x1,City_y1,YELLOW);
            puthz(City_x0+Text_xplus,City_y0+Text_yplus,24,24,BLA
CK,"城区图例");
            break;
        }
    case 2:
        {
```

```

        Button_Edge(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y1,W
HITE);
        Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y1,YEL
LOW);
        puthz(Suburb_x0+Text_xplus,Suburb_y0+Text_yplus,24,24
,BLACK,"郊区图例");
        break;
    }
    case 3:
    {
        Button_Edge(Mountain_x0,Mountain_y0,Mountain_x1,Mount
ain_y1,WHITE);
        Solid_Bar(Mountain_x0,Mountain_y0,Mountain_x1,Mountai
n_y1,YELLOW);
        puthz(Mountain_x0+Text_xplus,Mountain_y0+Text_yplus,2
4,24,BLACK,"山区图例");
        break;
    }
    case 4:
    {
        Button_Edge(Animation_x0,Animation_y0,Animation_x1,Ani
mation_y1,WHITE);
        Solid_Bar(Animation_x0,Animation_y0,Animation_x1,Anim
ation_y1,YELLOW);
        puthz(Animation_x0+Text_xplus,Animation_y0+Text_yplus
,24,24,BLACK,"人物动画");
        break;
    }
}
}

```

```

/*****
*****/

```

```

* 函数名称      ChooseButton_Darken_Sign
* 函数作用      熄灭按钮
* 函数输入      x0, y0 按钮绘制左上角, x1, y1 按钮绘制右下角, flag
按钮序号
* 函数输出      无

```

```

*****/

```

```

void ChooseButton_Darken_Sign(int x0,int y0,int x1,int y1,int fla
g,int flag_choose_sign)
{

```

```

switch(flag)
{
case 1:
{
Button_Edge(City_x0,City_y0,City_x1,City_y1,LIGHT_GRA
Y);
if(flag_choose_sign==2)
{
Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y0
+1,GRAY);
}
else
{
Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y0
+1,DARK_GRAY);
}
Solid_Bar(City_x0,City_y0,City_x1,City_y1,DARK_GRAY);
puthz(City_x0+Text_xplus,City_y0+Text_yplus,24,24,WHI
TE,"城区图例");
break;
}
case 2:
{
Button_Edge(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y1,L
IGHT_GRAY);
if(flag_choose_sign==1)
{
Solid_Bar(Suburb_x0,Suburb_y0-
2,Suburb_x1,Suburb_y0-1,GRAY);
Solid_Bar(Suburb_x0,Suburb_y1+1,Suburb_x1,Suburb_
y1+2,DARK_GRAY);
}
else if(flag_choose_sign==3)
{
Solid_Bar(Suburb_x0,Suburb_y0-
2,Suburb_x1,Suburb_y0-1,DARK_GRAY);
Solid_Bar(Suburb_x0,Suburb_y1+1,Suburb_x1,Suburb_
y1+2,GRAY);
}
else
{
Solid_Bar(Suburb_x0,Suburb_y0-
2,Suburb_x1,Suburb_y0-1,DARK_GRAY);

```

```

        Solid_Bar(Suburb_x0,Suburb_y1+1,Suburb_x1,Suburb_
y1+2,DARK_GRAY);
    }
    Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y1,DAR
K_GRAY);
    puthz(Suburb_x0+Text_xplus,Suburb_y0+Text_yplus,24,24
,WHITE,"郊区图例");
    break;
}
case 3:
{
    Button_Edge(Mountain_x0,Mountain_y0,Mountain_x1,Mount
ain_y1,LIGHT_GRAY);
    if(flag_choose_sign==2)
    {
        Solid_Bar(Mountain_x0,Mountain_y0-
2,Mountain_x1,Mountain_y0-1,GRAY);
        Solid_Bar(Mountain_x0,Mountain_y1+1,Mountain_x1,M
ountain_y1+2,DARK_GRAY);
    }
    else if(flag_choose_sign==4)
    {
        Solid_Bar(Mountain_x0,Mountain_y0-
2,Mountain_x1,Mountain_y0-1,DARK_GRAY);
        Solid_Bar(Mountain_x0,Mountain_y1+1,Mountain_x1,M
ountain_y1+2,GRAY);
    }
    else
    {
        Solid_Bar(Mountain_x0,Mountain_y0-
2,Mountain_x1,Mountain_y0-1,DARK_GRAY);
        Solid_Bar(Mountain_x0,Mountain_y1+1,Mountain_x1,M
ountain_y1+2,DARK_GRAY);
    }
    Solid_Bar(Mountain_x0,Mountain_y0,Mountain_x1,Mountai
n_y1,DARK_GRAY);
    puthz(Mountain_x0+Text_xplus,Mountain_y0+Text_yplus,2
4,24,WHITE,"山区图例");
    break;
}
case 4:
{
    Button_Edge(Animation_x0,Animation_y0,Animation_x1,An

```



```

        imation_y1,LIGHT_GRAY);
        if(flag_choose_sign==3)
        {
            Solid_Bar(Mountain_x0,Mountain_y1-
1,Mountain_x1,Mountain_y1,GRAY);
        }
        else
        {
            Solid_Bar(Mountain_x0,Mountain_y1-
1,Mountain_x1,Mountain_y1,DARK_GRAY);
        }
        Solid_Bar(Animation_x0,Animation_y0,Animation_x1,Anim
ation_y1,DARK_GRAY);
        puthz(Animation_x0+Text_xplus,Animation_y0+Text_yplus
,24,24,WHITE,"人物动画");
        break;
    }
}
}

```

```

/*****
*****

```

```

* 函数名称      ChooseButton_Press_Sign
* 函数作用      按下按钮
* 函数输入      x0, y0 按钮绘制左上角, x1, y1 按钮绘制右下角, flag
按钮序号
* 函数输出      无

```

```

*****
*****/

```

```

void ChooseButton_Press_Sign(int x0,int y0,int x1,int y1,int flag
)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(City_x0,City_y0,City_x1,City_y1,LIGHT
_GRAY);
            Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y0
+1,DARK_GRAY);
            Solid_Bar(City_x0,City_y0,City_x1,City_y1,GRAY);
            puthz(City_x0+Text_xplus,City_y0+Text_yplus,24,24
,WHITE,"城区图例");

```

```

        break;
    }
    case 2:
    {
        Button_Edge(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_
y1,LIGHT_GRAY);
        Solid_Bar(Suburb_x0,Suburb_y0-
2,Suburb_x1,Suburb_y0-1,DARK_GRAY);
        Solid_Bar(Suburb_x0,Suburb_y1+1,Suburb_x1,Suburb_
y1+2,DARK_GRAY);
        Solid_Bar(Suburb_x0,Suburb_y0,Suburb_x1,Suburb_y1
,GRAY);
        puthz(Suburb_x0+Text_xplus,Suburb_y0+Text_yplus,2
4,24,WHITE,"郊区图例");
        break;
    }
    case 3:
    {
        Button_Edge(Mountain_x0,Mountain_y0,Mountain_x1,M
ountain_y1,LIGHT_GRAY);
        Solid_Bar(Mountain_x0,Mountain_y0-
2,Mountain_x1,Mountain_y0-1,DARK_GRAY);
        Solid_Bar(Mountain_x0,Mountain_y1+1,Mountain_x1,M
ountain_y1+2,DARK_GRAY);
        Solid_Bar(Mountain_x0,Mountain_y0,Mountain_x1,Mou
untain_y1,GRAY);
        puthz(Mountain_x0+Text_xplus,Mountain_y0+Text_ypl
us,24,24,WHITE,"山区图例");
        break;
    }
    case 4:
    {
        Button_Edge(Animation_x0,Animation_y0,Animation_x
1,Animation_y1,LIGHT_GRAY);
        Solid_Bar(Mountain_x0,Mountain_y1-
1,Mountain_x1,Mountain_y1,DARK_GRAY);
        Solid_Bar(Animation_x0,Animation_y0,Animation_x1,
Animation_y1,GRAY);
        puthz(Animation_x0+Text_xplus,Animation_y0+Text_y
plus,24,24,WHITE,"人物动画");
        break;
    }

```

```

    }
}

```

37. surface.c

```

#include "headfile.h"

#define Buttom_x_left 256
#define Buttom_x_right 767
#define Buttom_y_up 192-8-32
#define Buttom_y_down 575+8+32
#define Message_x_left (Buttom_x_left+8)
#define Message_x_right (Buttom_x_right-8)
#define Message_y_up (Buttom_y_up+16+64+16)
#define Message_y_down Message_y_up+8+32+8+32+8+32+8+32+16+8+32+8
+32

#define register_x0 Buttom_x_left+8
#define register_y0 Message_y_down+8
#define register_x1 512-4
#define register_y1 Buttom_y_down-8
#define out_x0 512+4
#define out_y0 Message_y_down+8
#define out_x1 Buttom_x_right-8
#define out_y1 Buttom_y_down-8
#define loginword_x0 Buttom_x_left+100
#define loginword_y0 Message_y_down+38
#define outword_x0 608
#define outword_y0 Message_y_down+8+30
#define input1_x0 Message_x_left+8+8
#define input1_y0 Message_y_up+8+32+8
#define input1_x1 Message_x_right-8-8
#define input1_y1 Message_y_up+8+32+8+32
#define input2_x0 Message_x_left+8+8
#define input2_y0 Message_y_up+8+32+8+32+8+32+8
#define input2_x1 Message_x_right-8-8
#define input2_y1 Message_y_up+8+32+8+32+8+32+8+32
#define input3_x0 Message_x_left+8+8
#define input3_y0 Message_y_up+8+32+8+32+8+32+8+32+8+32+8
#define input3_x1 Message_x_right-8-8
#define input3_y1 Message_y_up+8+32+8+32+8+32+8+32+8+32+8+32

#define msg_x0 312
#define msg_y0 309

```

```

#define msg_x1 712
#define msg_y1 459

/*****
*****
* 函数名称      Draw_Register
* 函数作用      绘制注册界面
* 函数输入      无
* 函数输出      无
*****
*****/

void Draw_Register()
{
    Button_Draw(Button_x_left,Button_y_up,Button_x_right,Button_y
_down,DARK_GRAY,GRAY);
    puthz(464,Button_y_up+24,48,48,BLACK,"注册");
    Button_Draw(Message_x_left,Message_y_up,Message_x_right,Messa
ge_y_down,Hollow_Color,BLACK);
    puthz(Message_x_left+8,Message_y_up+8,32,32,WHITE,"用户名:
");
    Button_Draw(input1_x0,input1_y0,input1_x1,input1_y1,DARK_GRAY
,GRAY);
    puthz(Message_x_left+8,Message_y_up+8+32+8+32+8,32,32,WHITE,"
密码: ");
    Button_Draw(input2_x0,input2_y0,input2_x1,input2_y1,DARK_GRAY
,GRAY);
    puthz(Message_x_left+8,Message_y_up+8+32+8+32+8+32+8+32+8,32,
32,WHITE,"确认密码: ");
    Button_Draw(input3_x0,input3_y0,input3_x1,input3_y1,DARK_GRAY
,GRAY);

    Button_Draw(register_x0,register_y0,register_x1,register_y1,H
ollow_Color,Solid_Color);
    puthz(loginword_x0,loginword_y0,32,32,BLACK,"注册");

    Button_Draw(out_x0,out_y0,out_x1,out_y1,Hollow_Color,Solid_Co
lor);
    puthz(outword_x0,outword_y0,32,32,BLACK,"返回");
}

```

```

/*****
*****
* 函数名称      Darken_Register
* 函数作用      注册界面关闭
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Darken_Register()
{
    Solid_Bar(256,192-8-32,767,575+8+32,BLACK);
}

```

```

/*****
*****
* 函数名称      Button_Light_Register
* 函数作用      注册按钮高亮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void Button_Light_Register(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(register_x0,register_y0,register_x1,register_y1,WHITE);
            Solid_Bar(register_x0,register_y0,register_x1,register_y1,GREEN);
            puthz(loginword_x0,loginword_y0,32,32,GRAY,"注册");
            break;
        }
        case 2:
        {
            Button_Edge(out_x0,out_y0,out_x1,out_y1,WHITE);
            Solid_Bar(out_x0,out_y0,out_x1,out_y1,RED);
            puthz(outword_x0,outword_y0,32,32,GRAY,"返回");
            break;
        }
    }
}

```

```

    }
}

/*****
*****
* 函数名称      Button_Darken_Register
* 函数作用      注册按钮熄灭
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

void Button_Darken_Register(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(register_x0,register_y0,register_x1,register_y1,GRAY);
            Button_Draw(register_x0,register_y0,register_x1,register_y1,Hollow_Color,Solid_Color);
            puthz(loginword_x0,loginword_y0,32,32,BLACK,"注册");
            break;
        }
        case 2:
        {
            Button_Edge(out_x0,out_y0,out_x1,out_y1,GRAY);
            Button_Draw(out_x0,out_y0,out_x1,out_y1,Hollow_Color,Solid_Color);
            puthz(outword_x0,outword_y0,32,32,BLACK,"返回");
            break;
        }
    }
}

/*****
*****
* 函数名称      Reg_Input_Username
* 函数作用      注册输入用户名
* 函数输入      user_buffer 用户名缓存
* 函数输出      无
*****
*****/

```

*****/

```
void Reg_Input_Username(char* user_buffer)
{
    char user_temp[11]={0};
    char* p=user_temp;
    char i=0,l=0;
    mousehide(MouseX,MouseY);
    while(1)
    {
        i=bioskey(0);

        if(l<=10)
        {
            if(i!=033&&i!='\b'&&i!='\n'&&i!='\r')
            {
                l++;
                *p=i;
                p++;
                *p=0;
            }
            else if(i==033)
            {
                break;
            }
            else if(i=='\b'&&l>=0)
            {
                *p=0;
                *(p-1)=0;
                p--;
                l--;
            }
            else if(i=='\n' || i=='\r')
            {
                break;
            }
        }
        if(l>10)
        {
            break;
        }
    }
}
```

```

        Reg_String_Display(input1_x0+8,input1_y0+8,1,user_temp);
    }
    strcpy(user_buffer,user_temp);
}

```

```

/*****
*****
* 函数名称      Reg_Input_Password
* 函数作用      注册输入密码
* 函数输入      pass_buffer 密码缓存
* 函数输出      无
*****
*****/

```

```

void Reg_Input_Password(char* pass_buffer)
{
    char pass_temp[11]={0};
    char* p=pass_temp;
    char i=0,l=0;
    char visualess[11]={0};
    char* v=visualess;
    mousehide(MouseX,MouseY);
    while(1)
    {
        i=bioskey(0);
        if(l<=10)
        {
            if(i!=033&&i!='\b'&&i!='\n'&&i!='\r')
            {
                l++;
                *p=i;
                p++;
                *p=0;
                *v='*';
                v++;
                *v=0;
            }
            else if(i==033)
            {
                break;
            }
            else if(i=='\b'&&l>=0)
            {

```



```

        *p=0;
        *(p-1)=0;
        p--;
        *v=0;
        *(v-1)=0;
        v--;
        l--;
    }
    else if(i=='\n' || i=='\r')
    {
        break;
    }
}
if(l>10)
{
    break;
}
Reg_String_Display(input2_x0+8,input2_y0+8,2,visualess);
}
strcpy(pass_buffer,pass_temp);
}

```

```

/*****
*****
* 函数名称      Reg_Confirm_Password
* 函数作用      注册确认密码
* 函数输入      confirm_buffer 确认密码缓存
* 函数输出      无
*****
*****/

```

```

void Reg_Confirm_Password(char* confirm_buffer)
{
    char pass_temp[11]={0};
    char* p=pass_temp;
    char i=0,l=0;
    char visualess[11]={0};
    char* v=visualess;
    mousehide(MouseX,MouseY);
    while(1)
    {
        i=bioskey(0);
        if(l<=10)

```

```

{
    if(i!=033&&i!='\b'&&i!='\n'&&i!='\r')
    {
        l++;
        *p=i;
        p++;
        *p=0;
        *v='*';
        v++;
        *v=0;
    }
    else if(i==033)
    {
        break;
    }
    else if(i=='\b'&&l>=0)
    {
        *p=0;
        *(p-1)=0;
        p--;
        *v=0;
        *(v-1)=0;
        v--;
        l--;
    }
    else if(i=='\n' || i=='\r')
    {
        break;
    }
}
if(l>10)
{
    break;
}
Reg_String_Display(input3_x0+8,input3_y0+8,3,visualess);
}
strcpy(confirm_buffer,pass_temp);
}

```

```

/*****
*****

```

```

* 函数名称      Reg_String_Display
* 函数作用      注册界面字符串显示

```

```

* 函数输入      x,y 显示位置, flag 输入框序号, p 字符串地址
* 函数输出      无
*****
*****/

```

```

void Reg_String_Display(int x,int y,int flag,char* p)
{
    switch(flag)
    {
        case 1:
            Button_Draw(input1_x0,input1_y0,input1_x1,input1_y1,D
ARK_GRAY,DIMGRAY);
            putzm(x,y,16,16,WHITE,p);
            break;
        case 2:
            Button_Draw(input2_x0,input2_y0,input2_x1,input2_y1,D
ARK_GRAY,DIMGRAY);
            putzm(x,y,16,16,WHITE,p);
            break;
        case 3:
            Button_Draw(input3_x0,input3_y0,input3_x1,input3_y1,D
ARK_GRAY,DIMGRAY);
            putzm(x,y,16,16,WHITE,p);
            break;
    }
}

```

```

/*****
*****
* 函数名称      Uni_Msgbox
* 函数作用      通用消息提示框
* 函数输入      s 提示消息字符串
* 函数输出      无
*****
*****/

```

```

void Uni_Msgbox(char* s)
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);

```

```

        puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,s);
        delay(2000);
        Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
    }

/*****
*****
* 函数名称      Uni_Msgbox_1
* 函数作用      界面提示框1
* 函数输入      无
* 函数输出      无
*****
*****/

void Uni_Msgbox_1()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"请输入用户名!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

/*****
*****
* 函数名称      Uni_Msgbox_2
* 函数作用      界面提示框2
* 函数输入      无
* 函数输出      无
*****
*****/

void Uni_Msgbox_2()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"请输入密码!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

```

```

}

/*****
*****
* 函数名称      Uni_Msgbox_3
* 函数作用      界面提示框3
* 函数输入      无
* 函数输出      无
*****
*****/

void Uni_Msgbox_3()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"请确认密码!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

/*****
*****
* 函数名称      Uni_Msgbox_4
* 函数作用      界面提示框4
* 函数输入      无
* 函数输出      无
*****
*****/

void Uni_Msgbox_4()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"用户名重复!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

/*****
*****

```

```

*****
* 函数名称      Uni_Msgbox_5
* 函数作用      界面提示框5
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Uni_Msgbox_5()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"确认密码不匹配!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

```

```

/*****
*****
* 函数名称      Uni_Msgbox_6
* 函数作用      界面提示框6
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Uni_Msgbox_6()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"注册成功!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

```

38. svga.c

```
#include"headfile.h"
```

```

/*****
*****

```

```

* @author          unknown
* @editor          ytm
* @date            2022-4-16
* @notice          此版本为SVGA 64k 版本! 其中默认模式为
1024*768 64k !
*                  2022-1-27: 删除了1024*768 256 显示模式相关函数
和异或画点函数
*                  2022-3-26: 加入了水平、垂直线的显存写入函数
*                  2022-3-27: 加入了斜线的显存写入函数
*                  2022-4-4: 优化了程序的注释, 便于阅读
*                  2022-4-16: 优化了存取图像函数的运行效率, 新增了
存取标题艺术字的函数
*                  修复了bc 中可能出现的无法读取存储文
件的问题
* @reference       https://www.jianshu.com/p/8dfeed75f75b
*****
*****/

/*****
*****
* 函数名称       SetSVGA64k
* 函数作用       初始化 SVGA 模式
* 函数输入       无
* 函数输出       无
*****
*****/

void SetSVGA64k()
{
    union REGS in,out;
    in.x.ax = 0x4f02;
    in.x.bx = 0x117; /*对应的模式: 默认模式为1024*768 64k*/
    /*****
    获取SVGA 显示模式号bx。摘录64k 的模式号如下:
    模式号      分辨率      颜色数
    0x111      640*480      64K
    0x114      800*600      64K
    0x117      1024*768      64K
    *****/
    int86(0x10,&in,&out);
    if(out.x.ax!=0x004f)//若调用成功则返回0x004f
    {
        ReturnMode();
        printf("Error in setting SVGA64k!\n");
    }
}

```

```

        getch();
        exit(1);
    }
}

/*****
*****
* 函数名称      ReturnMode
* 函数作用      SVGA 切换失败后返回文本显示模式
* 函数输入      无
* 函数输出      无
*****
*****/
void ReturnMode()
{
    union REGS in;
    in.h.ah=0;
    in.h.al=(char)0x03;    //返回文本模式
    int86(0x10,&in,&in);
}

/*****
*****
* 函数名称      GetSVGA
* 函数作用      获得当前 svga 显示模式的信息
* 函数输入      无
* 函数输出      返回显示模式号
*****
*****/
unsigned int GetSVGA()
{
    union REGS out;
    out.x.ax = 0x4f03;
    int86(0x10,&out,&out);
    if(out.x.ax!=0x004f)
    {
        ReturnMode();
        printf("Error: is not SVGA!\n");
        getch();
        exit(1);
    }
    return(out.x.bx);
}

```



```

/*****
    获取 SVGA 显示模式号 bx。摘录 64k 的模式号如下：
    模式号          分辨率          颜色数
    0x111          640*480          64K
    0x114          800*600          64K
    0x117          1024*768        64K
    *****/

/*****
    * 函数名称      SelectPage
    * 函数作用      显存换页
    * 函数输入      page 页面号
    * 函数输出      切换成功输出为 0
    *****/

unsigned int SelectPage(unsigned char page)
{
    union REGS r;
    static unsigned char old_page = 0;
    static unsigned char flag = 0;
    r.x.ax = 0x4f05;
    r.x.bx = 0;
    if (flag == 0)
    {
        old_page = page;
        r.x.dx = page;
        int86(0x10, &r, &r);
        flag++;
    }
    if (page != old_page)
    {
        old_page = page;
        r.x.dx = page;
        int86(0x10, &r, &r);
    }

    return 0;
}

/*****
    * 函数名称      Putpixel64k

```

```

* 函数作用          用 64k 的模式画点
* 函数输入          x, y 画点的位置, color 画点颜色
* 函数输出          无
*****
*****/
void Putpixel64k(int x, int y, int color)
{
    if ( x>=0 && x<=1024 && y>=0 && y<=768 )
    {
        /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
        unsigned int far * const video_buffer = (unsigned int far
        *)0xa0000000L;

        /*要切换的页面号*/
        unsigned char newpage=0;

        /*对应显存地址偏移量*/
        unsigned long int page;

        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)y << 10) + x;
        newpage = page >> 15;    /*32k 个点一换页, 除以 32k 的替代算
法*/

        SelectPage(newpage);

        /*向显存写入颜色, 对应点画出*/
        *(video_buffer + page) = color;
    }
}

/*****
*****
* 函数名称          Getpixel64k
* 函数作用          用 64k 的模式读取点的颜色
* 函数输入          x, y 读取点的位置
* 函数输出          点的颜色
*****
*****/
unsigned int Getpixel64k(int x, int y)
{
    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0

```

```

xa0000000L;

/*要切换的页面号*/
unsigned char new_page;

/*对应显存地址偏移量*/
unsigned long int page;

/*判断点是否在屏幕范围内，不在就退出*/
if (x < 0 || x > (1024 - 1) || y < 0 || y > (768 - 1))
{
    return 0;
}

/*计算显存地址偏移量和对应的页面号，做换页操作*/
page = ((unsigned long int)y << 10) + x;
new_page = page >> 15;    /*32k 个点一换页，除以 32k 的替代算法*/
SelectPage(new_page);

/*返回颜色*/
return *(video_buffer + page);
}

/*****
*****
* 函数名称      Horizline
* 函数作用      画水平线函数
* 函数输入      int x                起始点横坐标，从左到右增
加，0 为最小值（屏幕参考系）
                  int y                起始点纵坐标，从上到下增
加，0 为最小值（屏幕参考系）
                  int width            水平长度，为正向右延伸，为
负向左延伸
                  unsigned char color  颜色
* 函数注意      可以接收超出屏幕范围的数据，只画出在屏幕内部分
*               因为没有防止整型变量溢出的判断，画超出屏幕的线时应防
止输入特大数据
* 函数输出      无
*****
*****/
void Horizline(int x, int y, int width, int color)
{
    /*显存指针常量，指向显存首地址，指针本身不允许修改*/

```

```
    unsigned int far * const video_buffer = (unsigned int far *  
)0xa0000000L;
```

```
    /*要切换的页面号*/
```

```
    unsigned char new_page = 0;
```

```
    //    unsigned char old_page = 0;
```

```
    /*对应显存地址偏移量*/
```

```
    unsigned long int page;
```

```
    /*i 是 x 的临时变量, 后作循环变量*/
```

```
    int i;
```

```
    /*判断延伸方向, 让起始点靠左*/
```

```
    if (width < 0)
```

```
    {
```

```
        x = x + width;
```

```
        width = -width;
```

```
    }
```

```
    i = x;
```

```
    /*省略超出屏幕左边部分*/
```

```
    if (x < 0)
```

```
    {
```

```
        x = 0;
```

```
        width += i;
```

```
    }
```

```
    /*整条线在屏幕外时退出*/
```

```
    if (x >= 1024)
```

```
    {
```

```
        return;
```

```
    }
```

```
    /*整条线在屏幕外时退出*/
```

```
    if (y < 0 || y >= 768)
```

```
    {
```

```
        return;
```

```
    }
```

```
    /*省略超出屏幕右边部分*/
```

```
    if (x + width > 1024)
```

```

{
    width = 1024 - x;
}

/* 计算显存地址偏移量和对应的页面号，做换页操作*/
page = ((unsigned long int)y << 10) + x;
new_page = page >> 15;

SelectPage(new_page);

/* 向显存写入颜色，水平线画出*/
for (i = 0; i < width; i++)
{
    *(video_buffer + page + i) = color;
}
}

/*****
*****
* 函数名称      Vertiline
* 函数作用      画竖直线函数
* 函数输入      int x                起始点横坐标，从左到右增
加，0 为最小值（屏幕参考系）
                  int y                起始点纵坐标，从上到下增
加，0 为最小值（屏幕参考系）
                  int width            竖直长度，为正下右延伸，为
负向上延伸
                  unsigned char color  颜色
* 函数注意      可以接收超出屏幕范围的数据，只画出在屏幕内部分
*              因为没有防止整型变量溢出的判断，画超出屏幕的线时应防
止输入特大数据
* 函数输出      无
*****
*****/
void Vertiline(int x, int y, int width, int color)
{
    /* 显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /* 要切换的页面号*/

```

```

unsigned char new_page = 0;

//    unsigned char old_page = 0;
/*对应显存地址偏移量*/
unsigned long int page;

/*i 是 y 的临时变量, 后作循环变量; j 是 y 的循环变量*/
int i,temp;

/*判断延伸方向, 让起始点靠上*/
if (width < 0)
{
    y = y + width;
    width = -width;
}

i = y;

/*省略超出屏幕上边部分*/
if (y < 0)
{
    y = 0;
    width += i;
}

/*整条线在屏幕外时退出*/
if (y >= 768)
{
    return;
}

/*整条线在屏幕外时退出*/
if (x < 0 || x >= 1024)
{
    return;
}

/*省略超出屏幕下边部分*/
if (y + width > 768)
{
    width = 768 - y;
}

```

```

/* 循环变量赋初值*/
temp = y;

/* 向显存写入颜色，竖直线画出*/
for (i = 0; i < width; i++ ,temp ++ )
{
    /* 计算显存地址偏移量和对应的页面号，做换页操作*/
    page = ((unsigned long int)temp << 10) + x;
    new_page = page >> 15;

    SelectPage(new_page);

    *(video_buffer + page ) = color;
}
}

/*****
*****
* 函数名称      Obliqline
* 函数作用      画斜线函数
* 函数输入      int x0                起始点横坐标，从左到右增
加，0 为最小值，1024 为最大值
                  int y0                起始点纵坐标，从上到下增
加，0 为最小值，1024 为最大值
                  int x1                结束点横坐标，从左到右增
加，0 为最小值，1024 为最大值
                  int y1                结束点纵坐标，从上到下增
加，0 为最小值，1024 为最大值
                  unsigned char color    颜色
* 函数注意      不可以接收超出屏幕范围的数据
*              因为没有防止整型变量溢出的判断，画超出屏幕的线时应防
止输入特大数据
* 函数输出      无
*****
*****/
void Obliqline(int x0, int y0, int x1, int y1, int color)
{
    /* 显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *
)0xa0000000L;

    /* 要切换的页面号*/
    unsigned char new_page = 0;

```

```

//    unsigned char old_page = 0;
/*对应显存地址偏移量*/
unsigned long int page;

/*i 是临时变量, 后作 x 的循环变量; j 是 y 的循环变量*/
int i,j;

/*用于斜线的浮点运算*/
double k,b;

/*****
注意: 并没有超出屏幕范围的数据的判断和处理
*****/

/*确保 x0,y0 在 x1,y1 左侧, 保证循环正常进行*/
if(x0 > x1)
{
    i = x0;
    x0 = x1;
    x1 = i;

    i = y0;
    y0 = y1;
    y1 = i;
}

/*计算直线相关参数*/
k = (double)(y1 - y0) / (double)(x1 - x0);
b = (double)(y0) - k * (double)(x0);

/*循环变量赋初值*/
j = y0;

/*向显存写入颜色, 斜线画出*/
for( i = x0 ; i <= x1 ; i++ )
{
    /*以 0.5 为分界点*/
    j = (floor)(k * i + b + 0.5);

    /*计算显存地址偏移量和对应的页面号, 做换页操作*/
    page = ((unsigned long int)j << 10) + i;
    new_page = page >> 15;
}

```



```

        SelectPage(new_page);

        *(video_buffer + page ) = color;
    }
}

/*****
*****
* 函数名称      Putbmp64k
* 函数作用      读取 24 位图片
* 函数输入      参数 x,y 为图片位置, name 为路径
* 函数输出      0 成功, 其余失败
*****
*****/
int Putbmp64k(int x,int y,const char *path)
{
    FILE *fpbmp;
    WESHEN *buffer;//行像素缓存指针
    long int width,height,linebytes;//一行像素所占字节数(含补齐空
字节)
    /*循环变量*/
    int i, j;

    /*图片位深*/
    unsigned int bit;

    /*压缩类型数*/
    unsigned long int compression;

    /*打开文件*/
    if ((fpbmp = fopen(path, "rb")) == NULL)
    {
        return 1;
    }

    /*读取位深*/
    fseek(fpbmp, 28L, 0);
    fread(&bit, 2, 1, fpbmp);

    /*非 24 位图则退出*/
    if (bit != 24U)
    {

```

```

        return 2;
    }

    /* 读取压缩类型*/
    fseek(fpbmp, 30L, 0);
    fread(&compression, 4, 1, fpbmp);

    /* 采用压缩算法则退出*/
    if (compression != 0UL)
    {
        return 3;
    }

    /* 读取宽度、高度*/
    fseek(fpbmp, 18L, 0);
    fread(&width, 4, 1, fpbmp);
    fread(&height, 4, 1, fpbmp);

    /* 计算一行像素占字节数，包括补齐的空字节*/
    linebytes = (3 * width) % 4;

    if(!linebytes)
    {
        linebytes = 3 * width;
    }
    else
    {
        linebytes = 3 * width + 4 - linebytes;
    }

    /* 开辟行像素数据动态储存空间*/
    if ((buffer = (WESHEN *)malloc(linebytes)) == 0)
    {
        return 4;
    }

    /* 行扫描形式读取图片数据并显示*/
    fseek(fpbmp, 54L, 0);
    for (i = height - 1; i > -1; i--)
    {
        fread(buffer, linebytes, 1, fpbmp);    /* 读取一行像素数据
*/

```

```

        /*一行像素的数据处理和画出*/
        for (j = 0; j < width; j++)
        {
            /*0x117 模式下，原图红绿蓝各 8 位分别近似为 5 位、6 位、5 位
            */
            buffer[j].R >>= 3;
            buffer[j].G >>= 2;
            buffer[j].B >>= 3;
            Putpixel64k(j + x, i + y,
                (((unsigned int)buffer[j].R) << 11)
                | (((unsigned int)buffer[j].G) << 5)
                | (((unsigned int)buffer[j].B)));    /* 计算最终颜色，红
            绿蓝从高位到低位排列*/
        }
    }

    free(buffer);
    fclose(fpbmp);
    return 0;
}

/*****
*****
* 函数名称      get_image
* 函数作用      得到一片区域的图像信息
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标, save 指向
                  存储信息的指针
* 函数输出      无
*****
*****/
void get_image(int x0,int y0,int x1,int y1,unsigned int far *save
)
{
    int i=0;                                /*循环变量*/
    int j=0;
    int wide=x1-x0;
    int high=y1-y0;

    /*显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int far *VideoBuffer=(unsigned int far *)0xa0000000L
;

```

```

    long int newpage=0;

    unsigned long pos;

    for(i=0;i<high;i++)
    {
        for(j=0;j<x1-x0;j++)
        {
            pos=((unsigned long)(y0+i)<<10)+x0+j;

            newpage=pos>>15 ;                                /*除以32k
的替代算法*/

            SelectPage(newpage);

            save[i * wide + j]= *(VideoBuffer + pos);
        }
    }
}

/*****
*****
* 函数名称      put_image
* 函数作用      显示出存储空间的图像信息
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标, save 指向
存储信息的指针
* 函数输出      无
*****
*****/
void put_image(int x0,int y0,int x1,int y1,unsigned int far *save
)
{
    int i=0;                                                /*循环变量*/
    int j=0;

    int wide=x1-x0;
    int high=y1-y0;

    unsigned int far *VideoBuffer=(unsigned int far *)0xa000000L
;

    long int newpage=0;

```

```

    unsigned long pos;

    for(i=0;i<high;i++)
    {
        for(j=0;j<x1-x0;j++)
        {

            pos=((unsigned long)(y0+i)<<10)+x0+j;

            newpage = pos>>15 ;                                /*除以32k
的替代算法*/

            SelectPage(newpage);

            *(VideoBuffer + pos)=save[i*wide+j];
        }
    }
}

/*****
*****
* 函数名称      save_image
* 函数作用      保存存储空间的图像信息
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标
* 函数注意      将图像信息存到固定文件里面
*               必须确保范围在屏幕内, 否则运行效率将不高
* 函数输出      无
*****
*****/
void save_image(int x0, int y0, int x1, int y1)
{
    /*定义文件指针*/
    FILE * fp;

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

```

```

/*循环变量*/
int i = 0;
int j = 0;
int wide = x1 - x0;
int high = y1 - y0;
int save = 0;

/*保存背景的代码*/
fp = fopen(".\\PICTURE\\SAVED_I", "wb");
if (fp == NULL)
{
    printf("the file cant creat\n");
    exit(1);
}

for (i = 0; i < high; i++)
{
    /*计算显存地址偏移量和对应的页面号,做换页操作*/
    page = ((unsigned long int)(y0 + i) << 10) + x0;

    new_page = page >> 15;    /*32k 个点一换页,除以 32k 的替代算
法*/

    SelectPage(new_page);

    for (j = 0; j < wide; j++)
    {
        save = *(video_buffer + page + j);

        fwrite(&save,sizeof(int),1,fp);
    }
}
fclose(fp);
}

/*****
*****
* 函数名称      printf_image
* 函数作用      显示出存储空间的图像信息
* 函数输入      x0, y0 左上角起始坐标, X1, Y1 右下角坐标
* 函数注意      从固定文件中读取这个背景像素,将图像信息覆盖到界面上
* 函数输出      无

```

```

*****
*****/
void printf_image(int x0, int y0, int x1, int y1)
{
    /*定义文件指针*/
    FILE * fpo;

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环变量*/
    int i = 0;
    int j = 0;
    int wide = x1 - x0;
    int high = y1 - y0;
    unsigned int save = 0;

    fpo = fopen(".\\PICTURE\\SAVED_I", "rb");
    if (fpo == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i < high; i++)
    {
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 + i) << 10) + x0;

        new_page = page >> 15;    /*32k 个点一换页, 除以 32k 的替代算
法*/

        SelectPage(new_page);

        for (j = 0; j < wide; j++)
        {

```

```

        fread(&save,sizeof(int),1,fpo);

        *(video_buffer + page + j) = save;
    }
}
fclose(fpo);
}

/*****
*****
* 函数名称      save_title
* 函数作用      保存存储空间的图像信息
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标
* 函数注意      将图像信息存到固定文件里面
*               必须确保范围在屏幕内, 否则运行效率将不高
* 函数输出      无
*****
*****/
void save_title(int x0, int y0, int x1, int y1)
{
    /*定义文件指针*/
    FILE * fp;

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    /*循环变量*/
    int i = 0;
    int j = 0;
    int wide = x1 - x0;
    int high = y1 - y0;
    int save = 0;

    /*保存标题艺术字*/
    fp = fopen(".\\TITLE\\SAVED_T", "wb");
    if (fp == NULL)

```



```

    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i < high; i++)
    {
        /* 计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 + i) << 10) + x0;

        new_page = page >> 15;    /*32k 个点一换页, 除以 32k 的替代算
法*/

        SelectPage(new_page);

        for (j = 0; j < wide; j++)
        {
            save = *(video_buffer + page + j);

            fwrite(&save, sizeof(int), 1, fp);
        }
    }
    fclose(fp);
}

/*****
*****
* 函数名称      printf_title
* 函数作用      显示出存储空间的图像信息
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标
* 函数注意      从固定文件中读取这个背景像素, 将图像信息覆盖到界面上
* 函数输出      无
*****
*****/
void printf_title(int x0, int y0, int x1, int y1)
{
    /* 定义文件指针*/
    FILE * fpo;

    /* 显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

```

```

/*要切换的页面号*/
unsigned char new_page;

/*对应显存地址偏移量*/
unsigned long int page;

/*循环变量*/
int i = 0;
int j = 0;
int wide = x1 - x0;
int high = y1 - y0;
unsigned int save = 0;

fpo = fopen(".\\TITLE\\SAVED_T", "rb");
if (fpo == NULL)
{
    printf("the file cant creat\n");
    exit(1);
}

for (i = 0; i < high; i++)
{
    /*计算显存地址偏移量和对应的页面号, 做换页操作*/
    page = ((unsigned long int)(y0 + i) << 10) + x0;

    new_page = page >> 15;    /*32k 个点一换页, 除以 32k 的替代算
法*/

    SelectPage(new_page);

    for (j = 0; j < wide; j++)
    {
        fread(&save, sizeof(int), 1, fpo);

        /*仅输出非黑色部分*/
        if(save!=BLACK)
        {
            *(video_buffer + page + j) = save;
        }
    }
}
fclose(fpo);
}

```

```

/*****
*****

* 函数名称      save_image0
* 函数作用      显示出存储空间的背景像素
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标
* 函数输出      无
*****
*****/
void save_image0(int x0, int y0, int x1, int y1)
{
    FILE * fp;                                /*定义文件指针*/

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    int i = 0;                                /*循环变量*/
    int j = 0;
    int wide = x1 - x0;
    int high = y1 - y0;
    int save = 0;

    char fg[20] = "saveim0.dat";
    fp = fopen(fg, "wb+");                    /*建立保存背景的文件*/
    if (fp == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<high; i++)
    {
        /*计算显存地址偏移量和对应的页面号, 做换页操作*/
        page = ((unsigned long int)(y0 + i) << 10) + x0;

        new_page = page >> 15;    /*32k 个点一换页, 除以 32k 的替代算

```

法*/

```
SelectPage(new_page);

for (j = 0; j<wide; j++)
{
    save = *(video_buffer + page + j);

    fwrite(&save,sizeof(unsigned int),1,fp);
}
}
fclose(fp);
}

/*****
*****
* 函数名称      printf_image0
* 函数作用      从文件中读取这个背景像素
* 函数输入      x0, y0 左上角起始坐标, x1, y1 右下角坐标
* 函数输出      无
*****
*****/
void printf_image0(int x0, int y0, int x1, int y1)
{
    FILE * fpo;                                /*定义文件指针*/

    /*显存指针常量, 指向显存首地址, 指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0
xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    int i = 0;                                /*循环变量*/
    int j = 0;
    int wide = x1 - x0;
    int high = y1 - y0;

    unsigned int save = 0;
```

```

    char fas[20]="saveim0.dat";
    fpo = fopen(fas, "rb+");           /*建立保存背景的文件
*/
    if (fpo == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<high; i++)
    {
        /*计算显存地址偏移量和对应的页面号，做换页操作*/
        page = ((unsigned long int)(y0 + i) << 10) + x0;

        new_page = page >> 15;      /*32k 个点一换页，除以 32k 的替代算
法*/

        SelectPage(new_page);

        for (j = 0; j<wide; j++)
        {
            fread(&save,sizeof(unsigned int),1,fpo);

            *(video_buffer + page + j) = save;
        }
    }
    fclose(fpo);
}

```

39. universe.c

```

#include "headfile.h"

#define GUEST 0
#define USER 1
#define ADMIN 2

#define Buttom_x_left 256
#define Buttom_x_right 767
#define Buttom_y_up 192
#define Buttom_y_down 575
#define Message_x_left (Buttom_x_left+8)
#define Message_x_right (Buttom_x_right-8)
#define Message_y_up (Buttom_y_up+16+64+16)
#define Message_y_down Message_y_up+8+32+8+32+8+32+8+32+16

```

```

#define login_x0 Buttom_x_left+8
#define login_y0 Message_y_down+8
#define login_x1 512-4
#define login_y1 Buttom_y_down-8
#define out_x0 512+4
#define out_y0 Message_y_down+8
#define out_x1 Buttom_x_right-8
#define out_y1 Buttom_y_down-8
#define loginword_x0 Buttom_x_left+100
#define loginword_y0 Message_y_down+38
#define outword_x0 608
#define outword_y0 Message_y_down+8+30
#define input1_x0 Message_x_left+8+8
#define input1_y0 Message_y_up+8+32+8
#define input1_x1 Message_x_right-8-8
#define input1_y1 Message_y_up+8+32+8+32
#define input2_x0 Message_x_left+8+8
#define input2_y0 Message_y_up+8+32+8+32+8+32+8
#define input2_x1 Message_x_right-8-8
#define input2_y1 Message_y_up+8+32+8+32+8+32+8+32

#define register_x0 Buttom_x_right-8-100
#define register_y0 Buttom_y_up+8
#define register_x1 Buttom_x_right-8
#define register_y1 Message_y_up-8

#define regiword_x0 Buttom_x_right-8-100+50-32
#define regiword_y0 Buttom_y_up+32

#define msg_x0 312
#define msg_y0 309
#define msg_x1 712
#define msg_y1 459

/*****
*****
* 函数名称      Draw_Login
* 函数作用      登录界面绘制
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Draw_Login()
{
    Button_Draw(Button_x_left,Button_y_up,Button_x_right,Button_y_down,DARK_GRAY,GRAY);
    puthz(464,Button_y_up+24,48,48,BLACK,"登录");
    Button_Draw(Message_x_left,Message_y_up,Message_x_right,Message_y_down,Hollow_Color,BLACK);
    puthz(Message_x_left+8,Message_y_up+8,32,32,WHITE,"用户名:");
    Button_Draw(input1_x0,input1_y0,input1_x1,input1_y1,DARK_GRAY,GRAY);
    puthz(Message_x_left+8,Message_y_up+8+32+8+32+8,32,32,WHITE,"密码:");
    Button_Draw(input2_x0,input2_y0,input2_x1,input2_y1,DARK_GRAY,GRAY);

    Button_Draw(login_x0,login_y0,login_x1,login_y1,Hollow_Color,Solid_Color);
    puthz(loginword_x0,loginword_y0,32,32,BLACK,"登录");

    Button_Draw(out_x0,out_y0,out_x1,out_y1,Hollow_Color,Solid_Color);
    puthz(outword_x0,outword_y0,32,32,BLACK,"返回");

    Button_Draw(register_x0,register_y0,register_x1,register_y1,Hollow_Color,Solid_Color);
    puthz(regiword_x0,regiword_y0,32,32,BLACK,"注册");
}

```

```

/*****
*****
* 函数名称      Button_Light_Login
* 函数作用      按钮高亮
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void Button_Light_Login(int flag)
{
    switch(flag)
    {

```

```

        case 1:
        {
            Button_Edge(login_x0,login_y0,login_x1,login_y1,WHITE
);
            Solid_Bar(login_x0,login_y0,login_x1,login_y1,GREEN);
            puthz(loginword_x0,loginword_y0,32,32,BLACK,"登录");
            break;
        }
        case 2:
        {
            Button_Edge(out_x0,out_y0,out_x1,out_y1,WHITE);
            Solid_Bar(out_x0,out_y0,out_x1,out_y1,RED);
            puthz(outword_x0,outword_y0,32,32,BLACK,"返回");
            break;
        }
        case 3:
        {
            Button_Edge(register_x0,register_y0,register_x1,regis
ter_y1,WHITE);
            Solid_Bar(register_x0,register_y0,register_x1,registe
r_y1,YELLOW);
            puthz(regiword_x0,regiword_y0,32,32,BLACK,"注册");
            break;
        }
    }
}

```

```

/*****
*****
* 函数名称      Button_Darken_Login
* 函数作用      按钮熄灭
* 函数输入      flag 按钮序号
* 函数输出      无
*****
*****/

```

```

void Button_Darken_Login(int flag)
{
    switch(flag)
    {
        case 1:
        {
            Button_Edge(login_x0,login_y0,login_x1,login_y1,GRAY)

```



```

;
        Button_Draw(login_x0,login_y0,login_x1,login_y1,Hollow_Color,Solid_Color);
        puthz(loginword_x0,loginword_y0,32,32,BLACK,"登录");
        break;
    }
    case 2:
    {
        Button_Edge(out_x0,out_y0,out_x1,out_y1,GRAY);
        Button_Draw(out_x0,out_y0,out_x1,out_y1,Hollow_Color,Solid_Color);
        puthz(outword_x0,outword_y0,32,32,BLACK,"返回");
        break;
    }
    case 3:
    {
        Button_Edge(register_x0,register_y0,register_x1,register_y1,GRAY);
        Button_Draw(register_x0,register_y0,register_x1,register_y1,Hollow_Color,Solid_Color);
        puthz(regiword_x0,regiword_y0,32,32,BLACK,"注册");
        break;
    }
}
}

```

```

/*****
*****
* 函数名称      Input_Username
* 函数作用      用户名输入
* 函数输入      user_buffer 用户名缓存
* 函数输出      无
*****
*****/

```

```

void Input_Username(char* user_buffer)
{
    char user_temp[11]={0};
    char* p=user_temp;
    char i=0,l=0;
    mousehide(MouseX,MouseY);
    while(1)
    {

```

```

i=bioskey(0);

if(l<=10)
{
    if(i!=033&&i!='\b'&&i!='\n'&&i!='\r')
    {
        l++;
        *p=i;
        p++;
        *p=0;
    }
    else if(i==033)
    {
        break;
    }
    else if(i=='\b'&&l>=0)
    {
        *p=0;
        *(p-1)=0;
        p--;
        l--;
    }
    else if(i=='\n' || i=='\r')
    {
        break;
    }
}
if(l>10)
{
    break;
}
String_Display(Message_x_left+8+8+8,Message_y_up+8+32+8+8
,1,user_temp);
}
strcpy(user_buffer,user_temp);
}

```

```

/*****
*****

```

```

* 函数名称      Input_Password
* 函数作用      密码输入

```

```

* 函数输入      pass_buffer 密码缓存
* 函数输出      无
*****
*****/

```

```

void Input_Password(char* pass_buffer)
{
    char pass_temp[11]={0};
    char* p=pass_temp;
    char i=0,l=0;
    char visualess[11]={0};
    char* v=visualess;
    mousehide(MouseX,MouseY);
    while(1)
    {
        i=bioskey(0);
        if(l<=10)
        {
            if(i!=033&&i!='\b'&&i!='\n'&&i!='\r')
            {
                l++;
                *p=i;
                p++;
                *p=0;
                *v='*';
                v++;
                *v=0;
            }
            else if(i==033)
            {
                break;
            }
            else if(i=='\b'&&l>=0)
            {
                *p=0;
                *(p-1)=0;
                p--;
                *v=0;
                *(v-1)=0;
                v--;
                l--;
            }
            else if(i=='\n' || i=='\r')

```

```

        {
            break;
        }
    }
    if(l>10)
    {
        break;
    }
    String_Display(input2_x0+8,input2_y0+8,2,visualess);
}
strcpy(pass_buffer,pass_temp);
}

/*****
*****

* 函数名称      String_Display
* 函数作用      字符串显示
* 函数输入      x,y 显示位置坐标, flag 显示序号, p 显示字符串
* 函数输出      无
*****
*****/

void String_Display(int x,int y,int flag,char* p)
{
    switch(flag)
    {
        case 1:
            Button_Draw(Message_x_left+8+8,Message_y_up+8+32+8,Message_x_right-8-8,Message_y_up+8+32+8+32,DARK_GRAY,DIMGRAY);
            putzm(x,y,16,16,WHITE,p);
            break;
        case 2:
            Button_Draw(input2_x0,input2_y0,input2_x1,input2_y1,DARK_GRAY,DIMGRAY);
            putzm(x,y,16,16,WHITE,p);
            break;
    }
}

/*****
*****

* 函数名称      Darken_Login
* 函数作用      登陆界面关闭

```

```

* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Darken_Login()
{
    Solid_Bar(256,192,767,575,BLACK);
}

```

```

/*****
*****
* 函数名称      Uni_Msgbox_7
* 函数作用      界面提示框7
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Uni_Msgbox_7()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
    Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
    puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"登录成功!");
    delay(2000);
    Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
}

```

```

/*****
*****
* 函数名称      Uni_Msgbox_8
* 函数作用      界面提示框8
* 函数输入      无
* 函数输出      无
*****
*****/

```

```

void Uni_Msgbox_8()
{
    Button_Draw(msg_x0,msg_y0,msg_x1,msg_y1,DARK_GRAY,GRAY);
    puthz(480,msg_y0+8+8,32,32,BLACK,"提示");
}

```

```

        Button_Draw(msg_x0+8,msg_y0+64,msg_x1-8,msg_y1-
8,Hollow_Color,BLACK);
        puthz(msg_x0+8+8,msg_y0+8+64,32,32,WHITE,"请检查用户名和密码!
");
        delay(2000);
        Solid_Bar(msg_x0,msg_y0,msg_x1,msg_y1,BLACK);
    }
40. weather.c
#include "headfile.h"

#define SUNNY_SKY SKY_BLUE
#define CLOUDY_SKY BABY_BLUE
#define RAINY_SKY GRAY

/*****
*****
* 函数名称      Sky
* 函数作用      绘制天空
* 函数输入      weather 天气序号, cLock_hour 时间
* 函数输出      无
*****
*****/

void Sky(int weather,int clock_hour)
{
    const int x=63,y=54;
    float Sun_Place=0.0;
    switch(weather)
    {
        case SUNNY:
            Solid_Bar(x,y,x+960,y+40,SUNNY_SKY);
            Sun_Place=(clock_hour-5.0)/13.0*960.0;
            Draw_Sun(x+(int)Sun_Place,y,YELLOW);
            Draw_Sun(x-51,15,YELLOW);
            //putsz(x,y,16,16,BLACK,cLock_hour);
            break;
        case CLOUDY:
            Solid_Bar(x,y,x+960,y+40,CLOUDY_SKY);
            Draw_Cloud(x-51,15,WHITE);
            break;
        case RAINY:
            Solid_Bar(x,y,x+960,y+40,RAINY_SKY);
            Draw_Rainy(x-51,15,GRAY);
    }
}

```

```

        break;
    }
}

/*****
*****
* 函数名称      Cloud_Setup
* 函数作用      初始化云朵
* 函数输入      weather 天气序号, cloud 云朵位置数组, speed 云朵速
度, forward 云朵云朵方向
* 函数输出      无
*****
*****/

void Cloud_Setup(int weather,int *cloud,int *speed,int *forward)
{
    int i;
    int cloud_num;
    switch(weather)
    {
        case SUNNY:
            cloud_num=2;
            break;
        case CLOUDY:
            cloud_num=8;
            break;
        case RAINY:
            cloud_num=10;
            break;
    }
    for(i=0;i<cloud_num;i++)
    {
        cloud[i]=random(936);
    }
    *speed=random(5)+1;
    *forward=random(2)+1;
}

/*****
*****
* 函数名称      Animate_Cloud
* 函数作用      云朵动画绘制
* 函数输入      weather 天气序号, cloud 云朵位置数组, speed 云朵速

```

度, *forward* 云朵云朵方向

* 函数输出 无

```
*****  
*****/
```

```
void Animate_Cloud(int weather,int *cloud,int speed,int forward)
{
    int i;
    const int x=63,y=54;
    int cloud_num;
    int cloud_color;
    switch(weather)
    {
        case SUNNY:
            cloud_num=2;
            cloud_color=WHITE;
            break;
        case CLOUDY:
            cloud_num=8;
            cloud_color=LIGHT_GRAY;
            break;
        case RAINY:
            cloud_num=10;
            cloud_color=DIMGRAY;
            break;
    }
    for(i=0;i<cloud_num;i++)
    {
        Draw_Cloud(x+cloud[i],y,cloud_color);
        if(forward==1)
        {
            cloud[i]-=speed;
            if(cloud[i]<0)
            {
                cloud[i]=935;
            }
        }
        if(forward==2)
        {
            cloud[i]+=speed;
            if(cloud[i]>=936)
            {
                cloud[i]=0;
            }
        }
    }
}
```



```

    }
}
41. welcome.c
#include "headfile.h"

/*****
*****

* @author          ytm
* @date            2022-4-16
*****
*****/

/*****
*****

* 函数名称          Welcome
* 函数作用          欢迎界面
* 函数输入          无
* 函数输出          无
*****
*****/

void Welcome()
{
    printf_title(300,215,762,281);
    /*原有文字输出备份*/
    /*
    puthz(306,221,64,76,DARK_GRAY,"定向越野模拟");
    puthz(305,220,64,76,GRAY,"定向越野模拟");
    puthz(304,219,64,76,GRAY,"定向越野模拟");
    puthz(303,218,64,76,GRAY,"定向越野模拟");
    puthz(302,217,64,76,LIGHT_GRAY,"定向越野模拟");
    puthz(301,216,64,76,LIGHT_GRAY,"定向越野模拟");
    puthz(300,215,64,76,WHITE,"定向越野模拟");
    */
    delay(100);
    /*等待系统初始化*/
    Loading_Welcome(512,473,3200);
}

```

第八部分 总结

一、组员叶庭茂总结

刚进入自动化学院，就听说了挂科率极高的 C 语言课程设计考试。老师再三强调要认真对待，在 C 课设一开始，我们就开始了 C 语言课程设计的工作。先是完成了需求分析，之后便抱着一本厚厚的 C 语言教程硬啃。

一开始，我们学习如何建立工程，学习鼠标的使用方法，之后，我们学习电脑内存地址的知识，了解 SVGA 函数的编写。这些知识对于我们来说并不是很好理解，毕竟我们对计算机系统的底层使之很难有一个非常确切的认识，所幸的是我们选择了一道去年的题目，能够得到学长的一些建议。于是，在问了一些学长的一些建议之后，我们正式开始了编写代码的过程。

代码过程的编写是一个极为耗时的过程，由于菜单界面大量的代码重复，往往一个下午只能完成一小部分。但是，我认为这磨练了我的意志力，让我明白了持之以恒的重要性。

虽然这是一个旧题目，但我仍然希望创新出一些新的功能。首先，去年的很多模拟功能是确定的地图，不能随机生成，但我希望地图可以随机生成，于是，我便花了一整天了解了不同区域的元素分布情况，编写了能根据地区不同元素分布进行随机生成的代码。完成代码的调试和运行之时，我能感受到一种成就感。解决了一个感觉无法解决的问题，真的是一件很棒的事情。这也让我深切地明白了创新的难度与重要性。

编写代码时候，我们还遇到许多的 bug，由于自己的代码很难在一个看不见参数的界面中调试，我为那些容易出 bug 的代码区域编写了能够导出变量数据的 debug 调试功能和能够实时将代码显示于屏幕之上的 gui 功能，这大大增加了我对代码的可视性，在 debug 中帮助了我很多。

即使过程比较坎坷，但是 C 课设教会了我很多。首先是解决问题方面，在一开始拿到题目时的一脸茫然，但后面分模块步步推进，每天投入一定的时间完成一两个功能，不知不觉三个多月过去了，重新审视自己的代码，会发现当初那庞大的任务，已经完成的差不多了。接着是责任心，起初遇到 bug，我总是会把责任归咎于 bc，但每次把 bug 找出后，其实都是自己的粗心大意，比如有一次我发现有一部分数组总会出错，一直认为是 bc 的内存管理问题，后来才发现是我使用 while 函数访问到不该访问的地址，并对这些地址的变量赋值导致的错误。慢慢的我在后期写代码时逐渐谨慎，在遇到 bug 时也能冷静分析，从自身出发找问题，而不是将原因归咎于其他因素。

在此非常感谢指导老师提供的帮助，帮我们在课设过程中明确方向。非常感谢学长们在课设过程中给我们提供的指导，在课设初期让我们少走了很多弯路。同时非常感谢我的搭档梁忻健，帮助我发现了了很多问题和 bug。

二、组员梁忻健总结

在我学习 C 语言课程时，平时作业写的代码都是几十行左右，我很难想象居然能用这些简单的代码写出如此复杂的程序。在选题时，由于选题顺序靠前，选择也比较多，而比起信息管理类系统，我更希望能够做一些画面精美、功能强大的软件，因此，我果断选择了定向越野模拟系统。而在刚开始时，我完全没有头绪，但经过和队友叶庭茂的讨论，完成了需求分析，也掌握了一些最基本的 SVGA 绘图函数。而在寒假的时候，我就开始进行绘制图像的工作。由于我们打算做一个能够基于规则下的随机地图生成，并且能发挥 SVGA——高清分辨率与色彩丰富的特定的软件，是不能使用贴图的。因此，我打算使用代码来完成地图的绘制。首先是地图图标绘制，这需要先使用 PS 在像素精度下对图标进行精细的绘制，并且再使用代码在 BC 中绘制。这需要大量的时间绘制，但尽管工作量不小，但代码量比较少，因此我也有一点茫然。但看着一个个精细的图标在 BC 中绘制出来，我也逐渐感受到了 C 语言的魅力，也有很大的成就感。

由于使用了 SVGA 模式，因此标准库文件 `graph.h` 便不能使用了。因此也需要我自己编写一些相关的通用函数，如——任意实心三角形、四边形、椭圆等图形的绘制，这也让我感受到了库文件的底层函数是如何编写的。

而到了整体地图的生成时，如何在有限的信息下由程序随机生成一幅完美的地图是一个很大的难题，一开始我只是把我绘制的图标往上贴，然而生成出来的效果很差，有如元素之间不协调、地图与实际情况不符等等。我只能不断进行调整，不断增加扫描算法，以增加地图生成的规则。在无数次的调整之后，我终于能够让程序绘制出一幅整体协调的地图。那一刻的成就感记忆犹新。

动画技术是我工作中最大的难点，首先是 SVGA 的运算量比 VGA 高不少，导致动画的刷新帧率很低，有时候还会出现“逐行扫描”的现象，而且 BC 提供的内存也比较少，因此导致我生成的地图缓存无法存放到内存中。在往届中也很少有使用 SVGA 模式的学长学姐，制作动画者就更少了。因此我只能自己从零钻研，而我参考到了鼠标动画的实现，其是只刷新鼠标所在的区域的背景。因此，这给了我一个思路。然而由于地图太大，不能存到内存。最终，我选择了把整幅地图切分成 56 小块保存到二进制文件中，在播放动画时，读取人物位置，刷新对应的地图区块，这样既避免了类似鼠标动画的频繁读写，又解决了内存不足的问题。该动画技术只需要进行读取文件操作，速度较快。因此，我也解决了 SVGA 的动画问题。我也懂得了，在学习编程时，应当参考他人的代码，但是不是简单的照搬套用，而是思考他人如何实现，进而思考还有没有适合自己程序的最佳算法，有没有改进的空间，这才是最重要的。

在这三个月的时间里，C 课设让我学到了很多，也必将留下一段难忘的经历。它让我从真正意义上体会到了一个实际软件的编写，也教会了我如何去进行团队协作。同时，也实现了我编写一个界面酷炫的程序的梦想。

最后，由衷感谢所有指导老师。他们在我编写程序时候有耐心的讲解，也有丰富的建议，给了我很大的帮助。感谢帮助过我的学长学姐们，为我提供了方向。也非常感谢我的队友叶庭茂，帮助我发现并解决了很多 BUG，提供了很大的支持。

“惟其痛苦，才有欢乐。” --贝多芬

“精诚所至，金石为开。” --《论衡·感虚篇》

三、代码分工和统计

主要文件名称	完成者	包含内容	代码行数
about.c	叶庭茂	关于界面	192
advance.c		高级图形库	367
clock.c		时间函数	108
find.c		寻路函数	1619
find1.c		寻路算法	1845
graph.c		图形库	197
help.c		帮助界面	716
loading.c		加载动画	157
main.c		主函数	63
map.c		地图随机函数	2062
menu.c		菜单界面	336
msgbox.c		消息框绘制	183
people.c		参赛人员数据处理	193
play.c		模拟比赛界面	898
quit.c		退出界面	135
search.c		搜索界面	1202
setting.c		设置界面	1296
setting1.c		设置界面子菜单 1	112
setting2.c		设置界面子菜单 2	398
setting3.c		设置界面子菜单 3	97
sign.c		图例显示界面	304
welcome.c		欢迎界面	7

animate.c	梁忻健	模拟动画实现	562
back.c		主界面背景	130
create.c		整幅地图生成	1123
gins.c		高级绘图函数	259
icon.c		区块图像绘制	1226
info.c		人物状态栏显示	114
login.c		登录界面的绘制及实现	432
moveofc.c		少年动画绘制	467
moveofm.c		中年动画绘制	492
moveofo.c		老年动画绘制	519
moveoft.c		青年动画绘制	568
register.c		注册的实现	557
surface.c		输入框、提示框绘制	320
universe.c		基础界面绘制	240
weather.c		天气绘制系统	112
help.c		动画帮助界面	262
hz.c	引用和编辑 学长的 代码	汉字输出	285
mouse.c		鼠标	358
asc.c		字符输出	336
svga.c		SVGA 显示函数	472

叶庭茂完成代码行数为 12487 行，梁忻健完成代码行数为 6902 行，合 19389 行。引用和编辑学长代码 1457 行。