高级语言程序设计

课设报告

题　　目：赛车游戏——疯狂赛车

学　　号：22072120

姓　　名：何贵意

指导教师：蔡越江

提交日期：

成绩评价表

实验报告内容	实验报告结构	实验报告图表	界面	最终成绩
☐丰富正确 ☐基本正确 ☐设计部分少 ☐过于简单	☐完全符合要求 ☐基本符合要求 ☐有欠缺	☐符合规范 ☐基本符合规范 ☐有一些错误	☐丰富 ☐有背景图片 ☐只有背景色 ☐按钮效果好	
程序功能实现	**程序执行情况**	**问题回答情况**	**总体评价**	
☐多个扩展功能 ☐少量扩展功能 ☐完成基本功能 ☐未完成基本功能	☐流畅 ☐界面有闪动 ☐操作不灵活 ☐有小错误	☐清楚、正确 ☐基本正确 ☐回答有部分错误 ☐不能回答问题		

教师签字:_____

目录

1、需求分析.....3
1.1 功能需求.....4
1.1.1 基本功能.....4
1.1.2 扩展功能.....5
1.2 数据需求.....8
1.3 界面需求.....8
1.3.1 一级主菜单界面.....8
1.3.2 二级选车界面.....9
1.3.3 三级游戏界面.....10
1.4 开发环境.....12
2 、游戏程序总体设计.....13
2.2 程序模块设计.....13
2.1.1 初始化模块(init).....13
2.1.2 按钮控制模块(bc).....14
2.1.3 用户管理模块(um).....15
2.1.3 游戏管理模块(gm).....16
2.1.4 排行榜模块(top).....17
2.2 程序动态流程设计.....19
2.3 主要数据结构.....20
2.3.1 按钮结构.....20
2.3.2 玩家车辆结构.....20
2.3.3 石头障碍物结构.....20
2.3.4 地图结构.....21
2.3.5 排行榜结构.....21
2.3.6 排行榜链表结构.....21
2.3.7 文件.....22
2.3.8 数据结构杂项.....22
3 、游戏实现技术点分析.....25
3.1 静态画面设计.....25
3.2 游戏动画设计.....26
3.3 游戏交互设计.....27
3.3 其他技术点分析.....28
3.3.1 图片显示.....28
3.3.2 音乐播放.....28
3.3.2 去除闪烁.....28
4、测试.....28
4.1 用户输入空格(或空输入)测试.....28
4.2 已注册用户密码输入错误测试.....29
4.3 暂停界面时停功能测试.....30
4.4 游戏通关过线画面测试.....31
4.5 排行榜及胜利界面结算测试.....32
5、用户手册.....33

6 、总结提高.....34
6.1 课设设计总结.....34
6.2 对课程的一些建议.....34
7、附件：程序源代码.....35

所有内容均已做开源处理，供学弟学妹学习，也欢迎大家随时进行订正，在此感谢所有帮助过我的老师以及同学们，谢谢大家！

Git:https://github.com/ShiyuBanzhou/CrazyRacingGame#crazyracinggame

cppaste:https://www.cppaste.com/p/tOG6Zijm

1、需求分析

利用基础的 C 语言以及额外引用的针对于 C 语言以及 C++语言的 EasyX 图形库(包含在<graphics.h>头文件中)制作一款简单的赛车训练游戏。

程序运行后，玩家可以输入用户名以及密码。玩家点击“开始游戏”按钮，即可开始游戏，进入选车画面，选车画面将提供三种不同类型，不同颜色的车辆。玩家可以根据自己的喜好选择车辆开始游戏，选车完成后，将提示选车成功并播放醒目的提示音。之后将正式进入游戏操作界面，将在信息显示区显示血量、导弹数量、里程数、车辆速度、剩余时间、玩家分数、车辆是否处于碰撞保护状态等信息，玩家需要通过键盘进行控制，‘W’、‘A’、‘S’、‘D’、‘↑’、‘←’、‘↓’、‘→’、‘Space’（空格）、‘Esc’等按键进行移动与发射导弹以及打开暂停界面等操作。游戏将在游戏开始进程 3s 后随机刷出石头路障，玩家需要通过控制自己的车辆来规避石头路障或是通过发射导弹来击毁石头路障为自己开路(这将提高游戏最后结束时结算的分数，每个被击碎的石头路障算作一百分)。触碰石头路障，将减少玩家血量一点。若在游戏运行时间归零之前，玩家血量归零，则进入游戏失败界面。待游戏运行时间归零时，若玩家血量未归零则进入游戏获胜的结算界面，并根据玩家获得的分数，获得相应的星星数量。此外，游戏将具有暂停菜单界面，可以提供时停功能。

该游戏应具有具体如下功能：

1、 程序成功运行后，将弹出信息窗口提示用户输入用户名信息，该信息框获取用户输入的信息后可以对用户名进行合理性判断（用户名中不能存在空格以及不能为空），并在用户输入不合理的用户名后弹出信息框进行提示，并供用户再次重新输入用户名。成功输入用户名后，将弹框提示用户输入密码，同时进行合理性判断（密码中不能存在空格），并在用户输入不合理的密码后弹出信息框进行提示，并供用户再次重新输入密码（注：如果用户非第一次游玩本游戏，即该玩家已注册过账号，本游戏将提供本地账号与密码的对照检测，对于不完全相同的密码将弹框提示用户剩余密码尝试错误，并在尝试次数结束后直接关闭游戏程序，避免他人暴力破解账号与密码）。

2、 成功登录游戏之后，将弹出欢迎框，并判断该用户是否第一次游玩本游戏。如果是第一次游玩本游戏，将弹框欢迎玩家第一次游玩 CrazzyHe（游戏制作者昵称）的游戏，如果是非第一次游玩本游戏，将弹框“您是否准备好打破记录了”，为玩家做好良好的游戏引导，并播放游戏的背景音乐。

3、 游戏欢迎界面结束后，将进入游戏主界面，包括三个主按钮：“开始游戏”、“玩法说明”、“退出游戏”。

4、 点击“开始游戏”按钮后将进入选车画面，提供三辆不同类型、不同品牌、不同外观的车辆供玩家选择，玩家可以通过点击对应车辆下方的选择按钮进行选择。

5、 成功选择车辆后，将正式开始游戏，游戏的目标为在规定的游戏时间内存活更长的时间以及争取获得更多的分数。游戏可以通过方向键以及 wasd 按键进行车辆位移的控制，空格键进行导弹的发射。若玩家触碰石头路障将减少一滴血量，若玩家血量归零，则提前结束游戏。玩家可以通过发射导弹击碎路障石头，并获得 100 分数。游戏提供信息显示区，将显示车辆的相关信息，如速度，公里数等以及游戏的相关信息，如剩余时间，导弹数量，血量，分数等。

6、 游戏失败与成功将进入相对应的游戏结算界面。游戏还应提供暂停界面，并通过按键 Esc 进行触发。

1.1 功能需求

1.1.1 基本功能

1、开始新游戏

程序成功运行后，将弹出信息窗口提示用户输入用户名信息，该信息框获取用户输入的信息后可以对用户名进行合理性判断（用户名中不能存在空格以及不能为空）， 并在用户输入不合理的用户名后弹出信息框进行提示，并供用户再次重新输入用户名。成功输入用户名后，将弹框提示用户输入密码，同时进行合理性判断（密码中不能存在空格），并在用户输入不合理的密码后弹出信息框进行提示，并供用户再次重新输入密码（注：如果用户非第一次游玩本游戏，即该玩家已注册过账号，本游戏将提供本地账号与密码的对照检测，对于不完全相同的密码将弹框提示用户剩余密码尝试错误，并在尝试次数结束后直接关闭游戏程序，避免他人暴力破解账号与密码）。

2、欢迎界面

成功登录游戏之后，将弹出欢迎框，并判断该用户是否第一次游玩本游戏。如果是第一次游玩本游戏，将弹框欢迎玩家第一次游玩 CrazzyHe（游戏制作者昵称）的游戏，如果是非第一次游玩本游戏，将弹框“您是否准备好打破记录了”，为玩家做好良好的游戏引导，并播放游戏的背景音乐。

3、游戏主界面

游戏欢迎界面结束后，将进入游戏主界面，包括三个主按钮：“开始游戏”、“玩法说明”、“退出游戏”。

（1）、点击“玩法说明”，将弹框游戏的游玩操作提示：

玩家需要通过键盘进行控制，‘W’、‘A’、‘S’、‘D’、‘↑’、‘←’、‘↓’、‘→’、‘Space’（空格）、‘Esc’等按键进行移动与发射导弹以及打开暂停界面等操作。

（2）、点击“退出游戏”，将直接退出游戏程序。

（3）、点击“开始游戏”（具体见 4、）。

4、开始游戏界面

点击“开始游戏”按钮后：

（1）、选车画面：

提供三辆不同类型、不同品牌、不同外观的车辆供玩家选择，玩家可以通过点击对应车辆下方的选择按钮进行选择。

（2）、游玩界面：

成功选择车辆后，将正式开始游戏，游戏的目标为在规定的游戏时间内存活更长的时间以及争取获得更多的分数。游戏可以通过方向键以及 wasd 按键进行车辆位移的控制，空格键进行导弹的发射。若玩家触碰石头路障将减少一滴血量，若玩家血量归零，则提前结束游戏。玩家可以通过发射导弹击碎路障石头，并获得 100 分数。游戏提供信息显示区，将显示车辆的相关信息，如速度，公里数等以及游戏的相关信息，如剩余时间，导弹数量，血量，分数等。

（3）、结算界面：

游戏失败与成功将进入相对应的游戏结算界面。游戏还应提供暂停界面，并通过按键 Esc 进行触发。若在游戏运行时间归零之前，玩家血量归零，则进入游戏失败界面。待游戏运行时间归零时，若玩家血量未归零则进入游戏获胜的结算界面，并根据玩家获得的分数，获得相应的星星数量（星星将按照获得的分数与星级获得表对照后，按规则点亮对应数量的星星）。此外，游戏将具有暂停菜单界面，可以提供时停功能。

5、暂停界面

暂停界面提供继续游戏以及退出游戏等按钮，方便玩家合力规划游戏时间。并在暂定界面进行时停处理，即暂停界面的时间流失不影响游戏中的剩余倒计时计算。

6、结算界面

结算界面将提供继续游戏以及退出游戏的按钮。

1.1.2 扩展功能

游戏已制作好安装程序，其名称为“疯狂赛车安装游戏”，双击运行安装程序，按照其指示进行安装，即可直接一键运行游戏，方便玩家进行游玩。

1、用户名及密码的合理性检测

用户进入游戏，将提示输入用户名以及密码，系统将对用户的输入进行合理性检测，以避免对程序造成破坏。用户的输入中不能含有空格以及输入不能入空，否则，将提示用户重新进行输入，并再次进行检测直至输入正确。系统还将读取目录下的 user.txt 文件，将用户名与已注册玩家用户名进行比对，如果相同，即为用户非第一次登录，则将会对本次输入密码以及注册时提供的密码进行比对，如果用户密码输入错误则提示用户重新输入并指出剩余尝试错误，并提示程序将在 x 次尝试后关闭，如果用户反复尝试失败则自动关闭程序，避免

暴力破解获得其他用户的信息。如果不相同，即为用户第一次登录，则将用户的信息存入 user.txt 文件中。

2、车辆选择

点击“开始游戏”后会进入选车界面，点击不同类型、不同外观的车辆下方对应的选车按钮后，可以改变后续游戏内的车辆状态，使得用户可以时玩时新，具有较强的可玩性。

3、发射导弹

在游戏中按空格按键可以发射导弹，并且导弹图片会与石头障碍物进行碰撞检测，若两图片发生碰撞，则将石头障碍物的图片贴图转变为爆炸火焰贴图，使得玩家游玩更加有代入感。

4、生命值||导弹数量||公里数||游戏分数

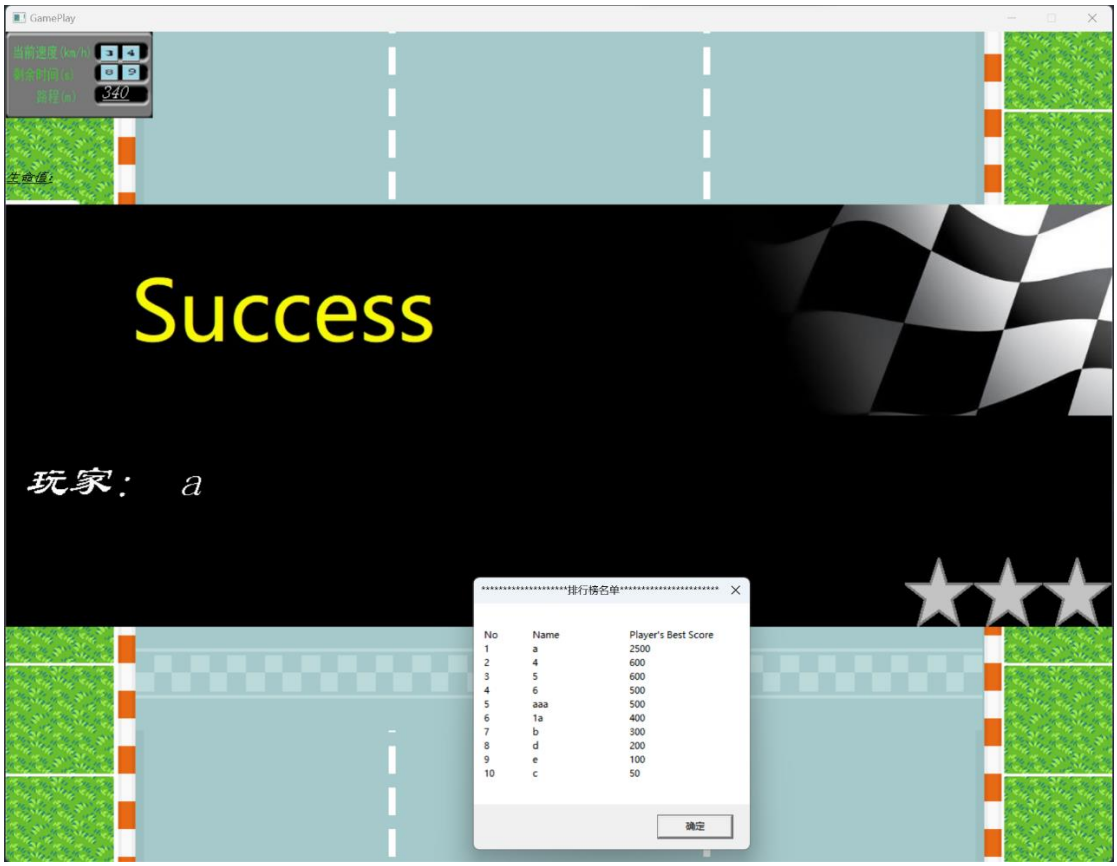
游戏游玩界面的左上角具有丰富的游戏数据及玩家数据显示信息，并将所有的信息呈现在信息框内，使得玩家可以更加便捷快速的获取自己想要获取的游戏信息。

5、排行榜

游戏结束后，若成功完成比赛，则会呈现排行榜画面，通过弹框的形式呈现 tops.txt 文件中的排行榜信息，实时排名，会先对文件内的信息进行合理性检测，并按照分数由高到低的顺序进行排序，随后将本次玩家游玩的数据与原本排行榜内的数据进行对比，如成功入榜，则恭喜玩家登榜成功，若未进入排行榜则鼓励玩家继续加油下次努力。最后再将新的排行榜数据信息录入到 tops.txt 文件当中，留取前十名排行榜信息，使得排行榜文件整洁有效。

6、结算界面

游戏结束后，若成功完成比赛，则会呈现结算界面，根据玩家本次游戏获得的分数，进行得分总结，将玩家的姓名信息以及分数信息呈现在屏幕上，并根据得分星级规则进行评星，将初始灰色的星星依次点亮，使得游戏结算更加能激发玩家的游玩兴趣。



7、配乐合理

游戏过程中，在对应操作的部分，及时给予配乐反馈，并合理的读取配乐文件，使得配乐在不影响游戏流畅运行的前提下，为玩家提供更强的代入感。

8、暂停界面

游戏过程中，提供通过 esc 按键进入暂停界面的功能，其中特别的，进入暂停界面后通过特定的算法实现游戏时间倒计时的暂停，使得玩家可以在不影响正常游戏进行的情况下，随意的进行时停，可以更加流畅和自由的体验本款游戏。

9、游戏说明

点击主界面中央的“游戏说明”按钮，将弹框游戏说明（教程）页面，这里列举了一些简单的游戏玩法及目标解释，给予玩家简单的指引与按钮作用说明。使得玩家可以更好的游玩游戏。

10、胜利过线

当游戏胜利时，游戏会自动将赛车平缓的移动至画面中央，在之后将出现赛车通过终点线的画面，使得玩家通关可以更加的有成就感。

1.2 数据需求

输入数据：用户名、密码信息
中间数据：玩家分数、玩家血量、导弹数量、障碍物数组、导弹数组、玩家车辆结构体、按钮结构体等。
输出数据：排行榜信息（包括分数排名前十的玩家的排名、用户名、分数信息）

1.3 界面需求

游戏程序设计了三级界面，第一级游戏界面为主菜单主界面，主要用于统筹游戏的不同功能区，具体包括“开始游戏”、“玩法说明”、“退出游戏”共三个功能区。第二级为点击开始游戏后的选车界面，包含三个可选择车辆以及选择按钮和选择状态的提示区。第三级为游戏的游玩界面，包含游戏运行过程中玩家的基本信息，如里程数、速度、剩余时间、生命值、导弹数量、玩家分数、玩家车辆、石头障碍物等，后在游戏运行结束后又分为其下级结算界面，如失败界面和胜利结算界面，在游戏过程当中，还包含游戏的暂停界面。

1.3.1 一级主菜单界面

一级菜单界面如下列图片所示：

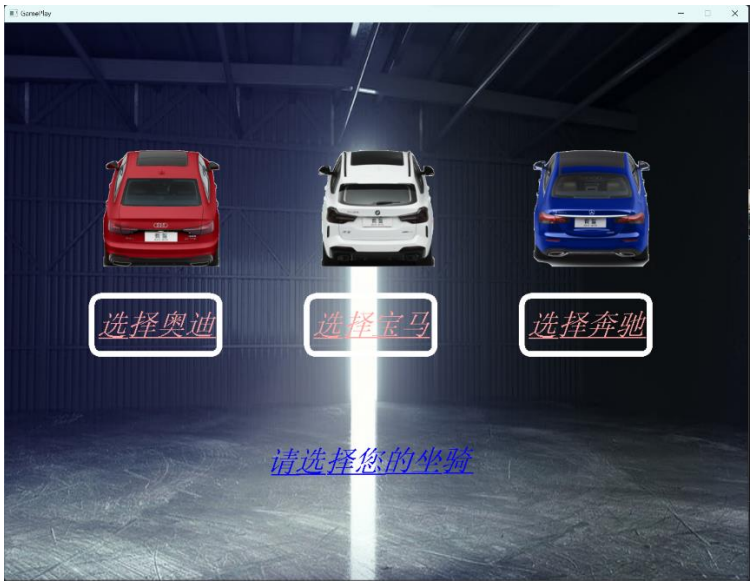
图表 1 登陆界面

图表 2 欢迎界面

图表 3 玩法介绍

1.3.2 二级选车界面

二级选车界面如下列图片所示：



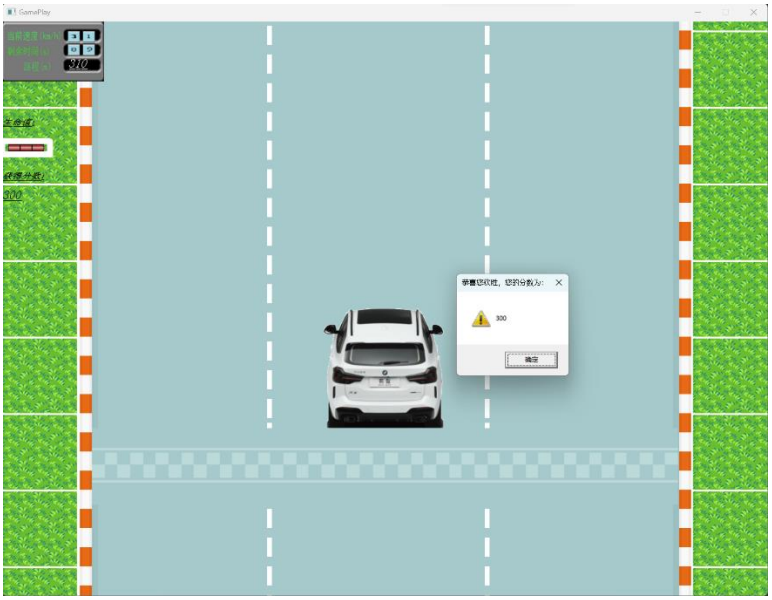
图表 4 选车界面

1.3.3 三级游戏界面

界面分为两个区域，游戏信息呈现区，主要以信息栏为中心，呈现游戏的相关信息，如里程数、速度、剩余时间、血量、游戏分数等。另一个区域为赛车运行的区域，围绕玩家赛车，石头障碍物和导弹进行运行。

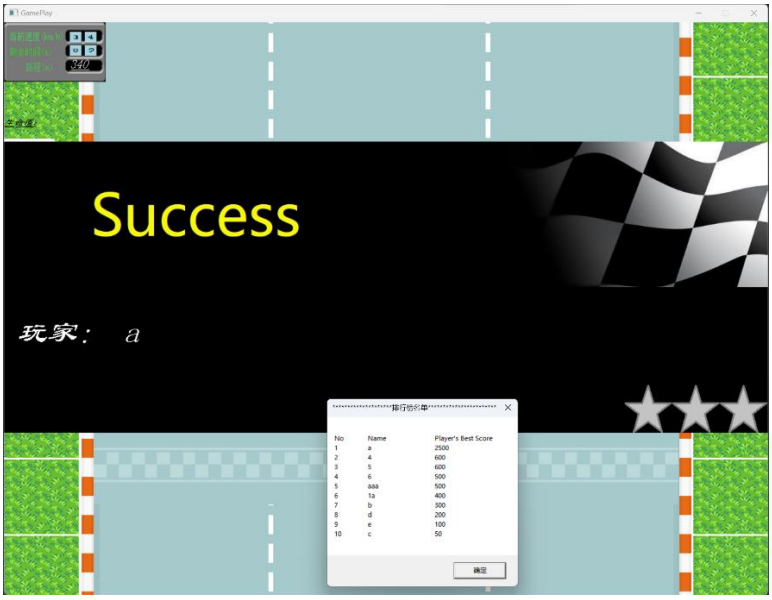
三级游戏运行界面如下列图片所示：

图表 5 游戏运行画面



图表 6 获胜界面

图表 7 失败界面



图表 8 结算界面

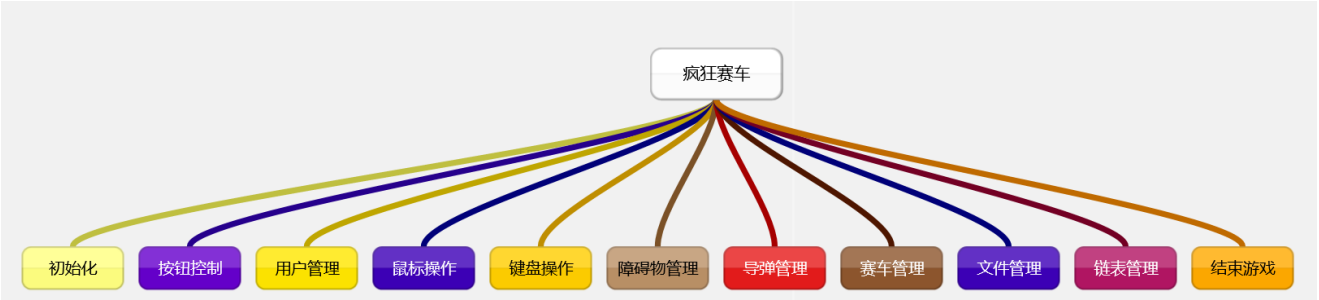
1.4 开发环境

Visual studio 2022 和 EasyX_2022。

2 、游戏程序总体设计

根据分析，游戏程序可大致划分为如下十一个模块：

1、 初始化
2、 按钮控制
3、 用户管理
4、 鼠标操作
5、 键盘操作
6、 障碍物管理
7、 导弹管理
8、 赛车管理
9、 文件管理
10、 链表管理
11、 结束游戏



图表 9 总体设计

2.1 程序模块设计

2.1.1 初始化模块(init)

本模块用于游戏主体运行前的初始化操作，为后续程序算法运行过程中需要用到的数据结构进行初始化赋值以及加载。

编号：init_1
函数原型：

```
Button* createButton(int x, int y, int width, int height, COLORREF color,
const char* buttonText);                    //按钮对象的创建
```

功能：实例化按钮结构体
参数：目标按钮想要创建的位置的左上角的（x，y）坐标，按钮宽度、高度，按钮的背景颜色，按钮中的文字内容。
返回值：按钮结构体指针（实例化后的按钮对象）

编号：init_2

函数原型：
Car* createCar(int x, int y, int speed, bool life, time_t last);//车对象的创建
功能：实例化玩家车辆结构体
参数：目标车辆想要创建的位置的左上角的（x，y）坐标，车辆速度，是否存活状态，上次碰撞时间。
返回值：玩家车辆结构体指针（实例化后的玩家车辆对象）

编号：init_3
函数原型：
void loadImage();　　//加载图片
功能：加载游戏主体运行过程中所需的所有图片
参数：无参数
返回值：无返回值

编号：init_4
函数原型：
Obstacle* createObstacle(int x, int y, int speed, bool status,
    time_t last);　　//障碍对象的创建
功能：实例化石头障碍物结构体
参数：目标石头障碍物想要创建的位置的左上角的（x，y）坐标，石头障碍物速度，石头障碍物的存在状态，石头障碍物的上次碰撞时间。

返回值：石头障碍物结构体指针（实例化后的石头障碍物对象）

编号：init_5
函数原型：
Map* createMap(int x, int y, int speed);　　//地图对象的创建
功能：实例化地图结构体
参数：目标地图想要创建的位置的左上角的（x，y）坐标，地图速度。
返回值：地图结构体指针（实例化后的地图对象）

编号：init_6
函数原型：
Rocket* createRocket(int x, int y, int speed, bool status); //火箭对象的创建
功能：实例化火箭结构体
参数：目标火箭想要创建的位置的左上角的（x，y）坐标，火箭速度，火箭发射的状态。
返回值：火箭结构体指针（实例化后的火箭对象）

2.1.2 按钮控制模块(bc)

本模块用于按钮实例化后的对象的控制以及负责点击其范围内后的响应事件。

编号：bc_1

函数原型：
void drawButton(Button* button, bool title);
//绘制按钮(title 用于调整字体)功能：实例化按钮结构体
参数：按钮结构体指针，该按钮是否用于标题位置
返回值：无返回值

编号：bc_2
函数原型：
bool mouseDetect(Button* button, ExMessage m);
//判断鼠标点击事件是否发生在按钮边框范围内
功能：判断鼠标点击事件是否发生在按钮结构体指针的范围内
参数：按钮结构体指针，鼠标信息
返回值：
若鼠标点击事件发生在按钮结构体指针内，则返回 true
否则，返回 false

2.1.3 用户管理模块(um)

本模块用于玩家用户信息与文件及游戏主体程序间的交互传输过程。

编号：um_1
函数原型：
char* login_name();　　//输入用户姓名
功能：获取用户输入的玩家名信息
参数：无参数
返回值：char 类型指针（用户玩家名）

编号：um_2
函数原型：
bool haveSpace(char* input);　　//检测用户输入是否含有空格
功能：判断用户输入的信息中是否含有空格
参数：char 类型指针（用户输入信息）
返回值：
若输入含有空格，返回 true
否则，返回 false

编号：um_3
函数原型：
char* login_password();　　//输入用户密码
功能：获取用户输入的密码信息
参数：无参数
返回值：char 类型指针（用户密码）

编号：um_4
函数原型：
void checkUsers(char* name);　　//检测用户是否存在
功能：判断本次游玩的用户是否已经注册过（信息已处于 users.txt 文件中）
参数：char 类型指针（用户玩家名）
返回值：无返回值

2.1.3 游戏管理模块(gm)

本模块主要用于游戏主体运行过程中的逻辑实践以及游戏内容的渲染。

编号：gm_1
函数原型：
void drawFrontCover();　　//创建游戏封面以及处理主界面按钮点击事件
功能:游戏主菜单的渲染以及处理鼠标点击事件(进入后续游戏主体运行过程中的哪个模块：“开始游戏”、“玩法介绍”、“退出游戏”)
参数：无参数
返回值：无参数

编号：gm_2
函数原型：
int drawCarSelection();　　//创建选车画面
功能：“开始游戏”内的选择玩家车辆的界面渲染
参数：无参数
返回值：int 值（对应所选择的车辆编号，用于后续车辆渲染时的贴图选择）

编号：gm_3
函数原型：
void mode_1();　　//模式一
功能：游戏主体程序运行（包括车辆控制，导弹控制，碰撞检测等）
参数：无参数
返回值：无返回值

编号：gm_4
函数原型：
void pauseInterface();　　//暂停界面
功能：暂停界面的渲染
参数：无参数
返回值：无返回值

编号：gm_5
函数原型：
void PutNumber(long x, long y, long number, int TextSize = 10,

```
    int TextProportion = 2, int TextInterval = 0,
    COLORREF tc = 0x000000, LPCTSTR font = "宋体");
```
//将 long 类型数字呈现在画布上
功能：将 long 类型数字呈现在画布上
参数：文字输出位置初始坐标（x，y），long 类型数字值
返回值：无返回值

编号：gm_6
函数原型：
```
void winInterface();                    //获胜界面
```
功能：获胜界面的渲染
参数：无参数
返回值：无返回值

编号：gm_7
函数原型：
```
void loseInterface();                   //失败界面
```
功能：失败界面的渲染
参数：无参数
返回值：无返回值

2.1.4 排行榜模块(top)

本模块主要用于排行榜的创建、排序、重构与输出等逻辑操作的实现

编号：top_1
函数原型：
```
TOPNODE* loadNode();                    //读取排行榜
```
功能：读取 tops.txt 文件中的排行榜信息
参数：无
返回值：TOPNODE 结构体头指针（排行榜头指针）

编号：top_2
函数原型：
```
void showNode();                            //展示排行榜
```
功能：展示排行榜信息
参数：无
返回值：无

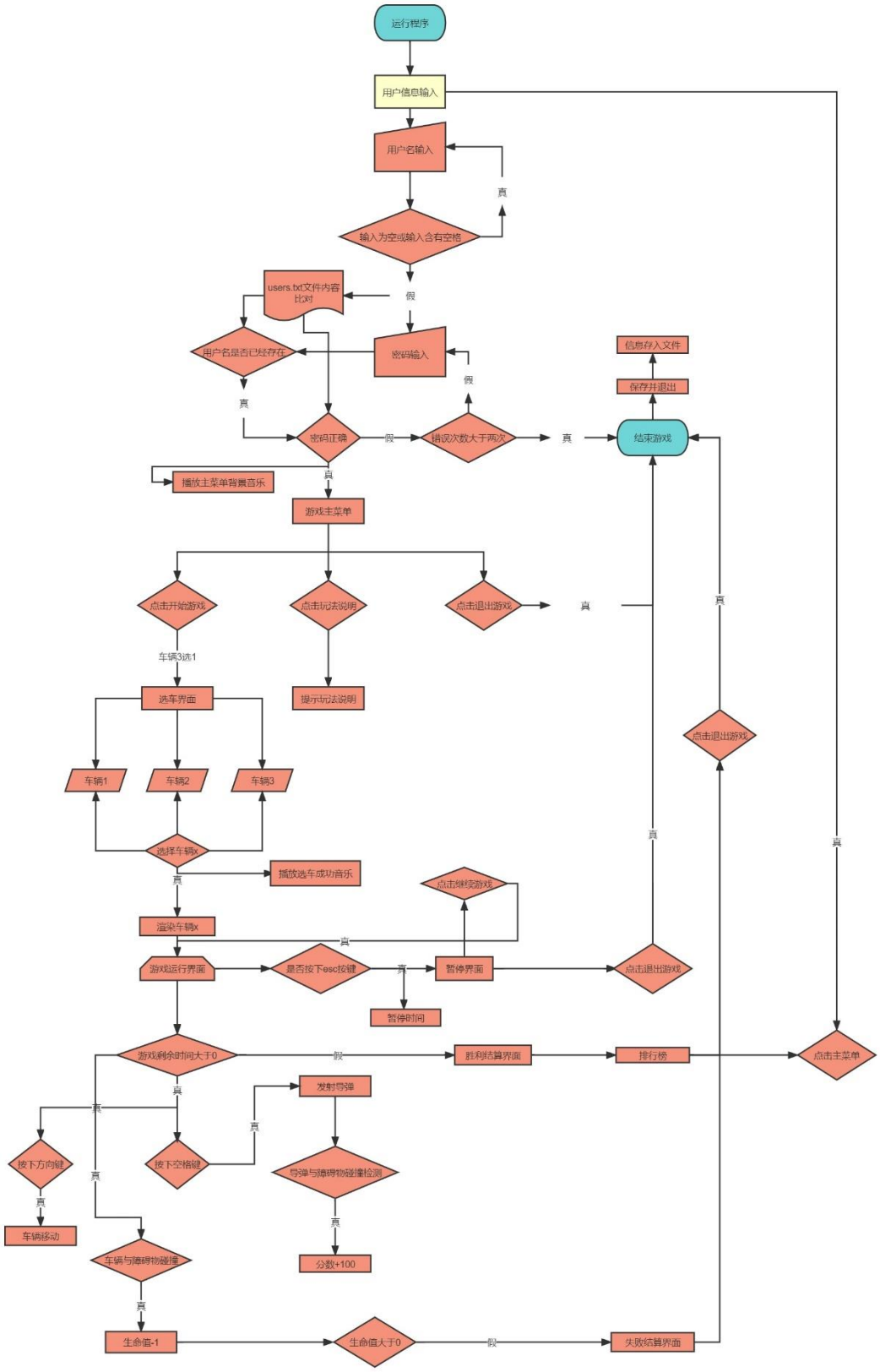
编号：top_3
函数原型：
```
void changeNode(TOPNODE* h);                //改变排行榜链表
```
功能：将本次用户游玩的分数信息与排行榜链表进行排序更改

参数：TOPNODE 链表结构体头指针
返回值：无

编号：top_4
函数原型：
TOPNODE* reNode(TOPNODE* h);　　//重构排行榜顺序
功能：将重新排序过后的排行榜链表进行重构
参数：TOPNODE 链表结构体头指针
返回值：TOPNODE 链表结构体头指针（重构后的排行榜头指针）

编号：top_5
函数原型：
void node_main();　　//排行榜链表主函数
功能：整合排行榜功能的核心函数
参数：无
返回值：无

2.2 程序动态流程设计



图表 10 程序动态流程设计

2.3 主要数据结构

2.3.1 按钮结构

包括按钮创建的初始（x，y）坐标以及其宽度、高度和该按钮的颜色信息与呈现文字内容信息。

```
//创建按钮的结构体
typedef struct button
{
    int x;                      //按钮 x 坐标
    int y;                      //按钮 y 坐标
    int width;                  //按钮宽度
    int height;                 //按钮高度
    COLORREF color;             //按钮颜色
    const char* buttonText;     //按钮呈现文字
}Button;
```

2.3.2 玩家车辆结构

包括玩家车辆的（x，y）坐标，以及速度信息，上次碰撞的时间，以及车辆存活信息。

```
//创建车的结构体
typedef struct car
{
    int x;                          //车的(x, y)坐标，以及速度
    int y;
    int speed;
    time_t last;                    //上次碰撞的时间
    bool life;                      //是否存活
}Car;
```

2.3.3 石头障碍物结构

包括石头障碍物刷新的初始信息，（x，y）坐标以及移动速度，上次接触时间以及其是否被碰撞的状态。

```
//创建障碍物的结构体
typedef struct obstacle {
    int x;
    int y;
    int speed;
```

```
    time_t last;                    //上次时间
    bool status;
}Obstacle;
```

2.3.4 地图结构

包括地图创建初始化时的（x，y）坐标信息以及图片位移的速度信息。

```
//创建地图的结构体
typedef struct map {
    int x;
    int y;
    int speed;
}Map;
```

2.3.5 排行榜结构

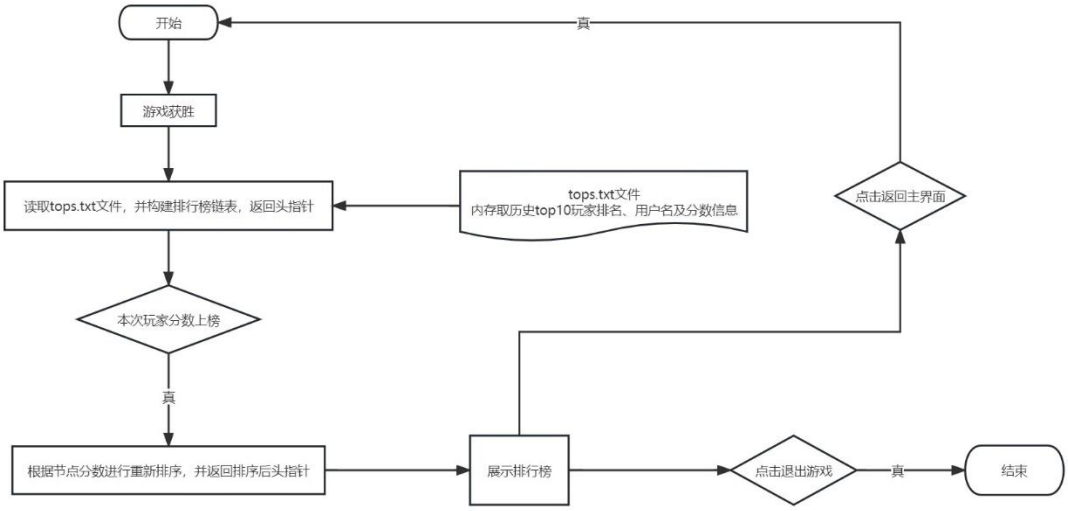
包括火箭创建初始化时的（x，y）坐标信息以及火箭位移速度和其发射状态相关信息

```
//创建火箭的结构体
typedef struct rocket {
    int x;
    int y;
    int speed;
    bool status;                    //火箭的状态，true 为可用，false 为不可用
}Rocket;
```

2.3.6 排行榜链表结构

用于创建排行榜信息的链表结构，包含玩家排名、游戏名、分数信息。

```
//创建排行榜链表
typedef struct node {
    int no;
    char name[50];
    int score;
    struct node* next;
}TOPNODE;
```



图表 11 链表实现

2.3.7 文件

tops.txt 用于存储游戏排行榜信息，即游戏程序运行有史以来的分数前十名的姓名以及分数信息（排名按从高到低）。

文件格式样例(排名 用户名 分数)：
1 user 2500
2 admin 2000
…

users.txt 用于保存玩家的登录信息(名称以及密码)用于登录检测。

文件格式样例(用户名 密码)：
user password
admin 0000000
…

2.3.8 数据结构杂项

```
//宏定义
#define LENGTH 2                    //车图片数组长度
#define WIDTH 1280                  //游戏窗口宽度(x)
#define HEIGHT 960                  //游戏窗口高度(y)
#define SIZE 200                    //车的尺寸
#define SIZE_OBSTACLE 128           //障碍物的尺寸
#define SIZE_NUM 20                 //数字的尺寸
#define SIZE_ROCKET 100             //火箭的尺寸
#define PI 3.14159265359            //Π
```

```
//状态全局变量
bool isMenu = false;                //是否点开过菜单
bool onFire = false;                //是否发射过火箭
bool isUp = false;                  //是否已按下过向上的按键
bool isDown = false;                //是否已按下过向下的按键
bool isHit = false;                 //是否已发生碰撞
bool first_play;                    //是否第一次游玩

//图片全局变量
IMAGE frontCover;                   //游戏封面图片
IMAGE grass;                        //草地图片
IMAGE car_Audi[LENGTH];             //奥迪车图片
IMAGE car_Bmw[LENGTH];              //宝马车图片
IMAGE car_Mercedes[LENGTH];         //奔驰车图片
IMAGE car_black[LENGTH];            //黑车图片
IMAGE garage;                       //车库图片
IMAGE map_1;                        //地图图片
IMAGE map_2;
IMAGE car_circle[LENGTH];           //旋转后汽车图片
IMAGE rock[LENGTH];                 //障碍物石头图片
IMAGE rocket_img[LENGTH];           //火箭图片
IMAGE number[10];                   //用于时间的数字组图片
IMAGE show;                         //信息框
IMAGE bomb[LENGTH];                 //爆炸图片
IMAGE life;                         //生命值图片
IMAGE success;                      //成功图片
IMAGE black;                        //黑色背景图
IMAGE star_black;                   //黑星
IMAGE star_yellow;                  //黄星
IMAGE terminalPoint;                //终点图片
IMAGE gameOver;                     //gameover 图片
IMAGE gameWin;                      //gamewin 图片
IMAGE bomb_car[LENGTH];             //车辆爆炸图片
IMAGE site;                         //设置图片
IMAGE temp;                         //暂时存储游戏信息
IMAGE add_life;                     //增加生命值图片
IMAGE add_rocket;                   //增加弹药图片

//消息全局变量
ExMessage m;                        //定义消息变量（鼠标）

//数据全局变量
double Forward = 0;                 //车辆方向
```

```
double Rota = 360;                    //角速度(转向速度)

//用户信息
char* name;                           //用户姓名
char* password;                       //用户密码

//地图像素点信息数组
int map_information[WIDTH][HEIGHT];

//终点初始 y 坐标
int end_y = -100;

//玩家获得分数全局变量
long score = 0;

//车辆速度全局变量
int speed_show = 30;
const int speed_min = 20;             //速度最小值
const int speed_max = 98;             //速度最大值
long distance = 0;                    //路程

//获得三星的条件
const int first_star = 500;           //一星分数下限
const int second_star =   2000;       //二星分数下限
const int third_star = 5000;          //三星分数下限

//时间全局变量
time_t game_sum = 0;                  //游戏总时间
time_t game_begin = 0;                //游戏开始时间
time_t game_progress = 0;             //游戏进程时间
time_t menu_begin = 0;                //开始进入菜单的时间
time_t menu_progress = 0;             //菜单进程时间
time_t menu_sum = 0;                  //菜单总时间
time_t fire_last = 0;                 //火箭上次发射时间
time_t fire_now = 0;                  //火箭现在发射时间
time_t control_up_last = 0;           //上次向上控制时间
time_t control_down_last = 0;         //上次向下控制时间

//模式一地图可供生成障碍物的横坐标
int map_obstacle[3] = {
    //一道
    230,
    //二道
    530,
```

```
    //三道
    900,
};
```

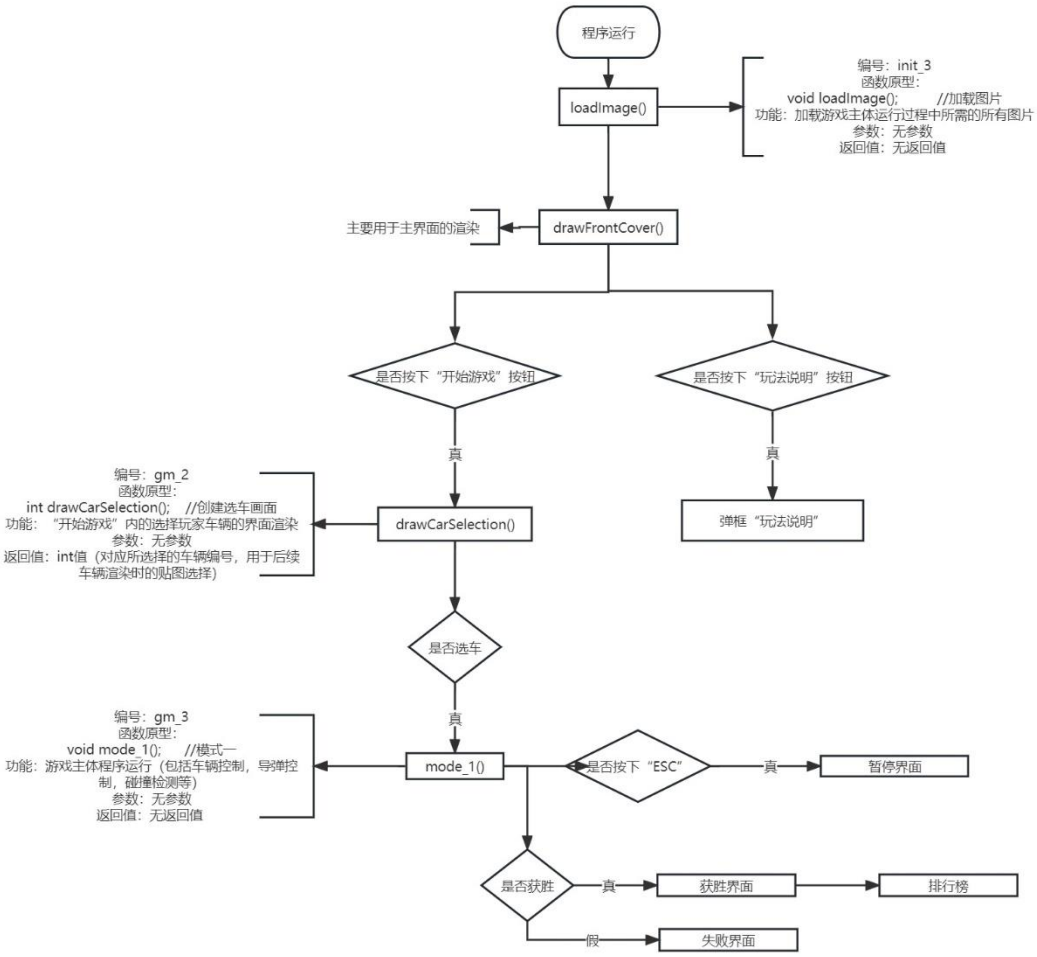
3 、游戏实现技术点分析

3.1 静态画面设计

在玩家运行程序后首先通过 loadImage 函数加载程序运行过程中所需的所有 IMAGE 类型的图片进行初始化，使得后续其他函数中可以直接进行图片的调用 putimage 函数。

静态画面的渲染主要出现在主菜单界面、选车界面、游戏运行界面、暂停界面、获胜结算界面、失败结算界面中。

在 drawFrontCover 函数即主菜单界面中，通过实例化三个按钮 Button 结构体的对象，配合 mouseDetect 函数的鼠标点击位置检测，使得对应的按钮对象在触发鼠标点击事件后，可以执行逻辑判断进入对应的函数。如点击“玩法说明”按钮，会弹框出玩法说明信息。点击“退出游戏”按钮会直接退出游戏。而点击“开始游戏”按钮，则进入 drawCarSelction 函数，背景贴图为车库图片，背景图片的上层贴图三辆不同类型的车辆供玩家进行选择，在玩家点击车辆下方对应的实例化按钮对象后，则跳转至 mode_1 函数即游戏主函数，此时游戏的背景图片为赛车车道，地图的上层附有玩家的车辆相关信息展示栏（速度、剩余时间、获得分数等），若玩家在游玩过程中按下“esc”按键则进入 pauseInterface 函数即暂停界面，以矩形框图形为背景，图层上附有继续游戏以及退出游戏两个实例化的按钮对象。若玩家顺利通关游戏（存活至游戏倒计时结束），则跳转至 winInterface 函数即胜利结算界面中，其背景图片为恭喜玩家顺利通关，还会显示玩家的姓名及分数信息，初始评星为三颗黑星，经系统对于玩家的分数与星级评分表对比后获得玩家所得星级，随后胜利界面将逐颗点亮对应的星星数量，对玩家的表现进行评星。若玩家未存活至倒计时结束，则进入失败界面。



图表 12 静态画面

3.2 游戏动画设计

通过在 mode_1 函数的游戏运行主函数中的 while（true）循环里对图片贴图的逻辑顺序以及贴图图片的改变进行游戏动画实现。游戏贴图通过位运算即制作同一张图片的黑底白图以及白底原图两图为一数组的掩码图形式实现去除图片背景的操作。同时利用 cleardevice（）、BeginBatchDraw（）和 FlushBatchDraw（）及 EndBatchDraw（）等函数进行图片缓存流的存入与输出实现图片加载的双缓冲操作，避免图片位移造成的画面撕裂或卡顿。游戏循环内部利用对时间指针的控制，避免了碰撞检测过程中持续碰撞造成的血量反复减少造成的游戏直接结束，即实现了游戏的碰撞后几秒内的碰撞保护机制。还通过对导弹与障碍物之间的碰撞检测实现对障碍物图片贴图状态的改变，即障碍物未被击碎时为石头贴图，而在导弹击碎石头后将呈现爆炸的动画。在游戏结束之后，还将对玩家操控车辆进行居中处理，并绘制玩家车辆通过终点线的动画，使玩家游玩更加的有代入感。在获胜结算界面中，实现了对玩家游玩操作进行评级的操作，即评级星星随玩家分数逐渐点亮的过程，以及结算画面动态插入画面的过程。

3.3 游戏交互设计

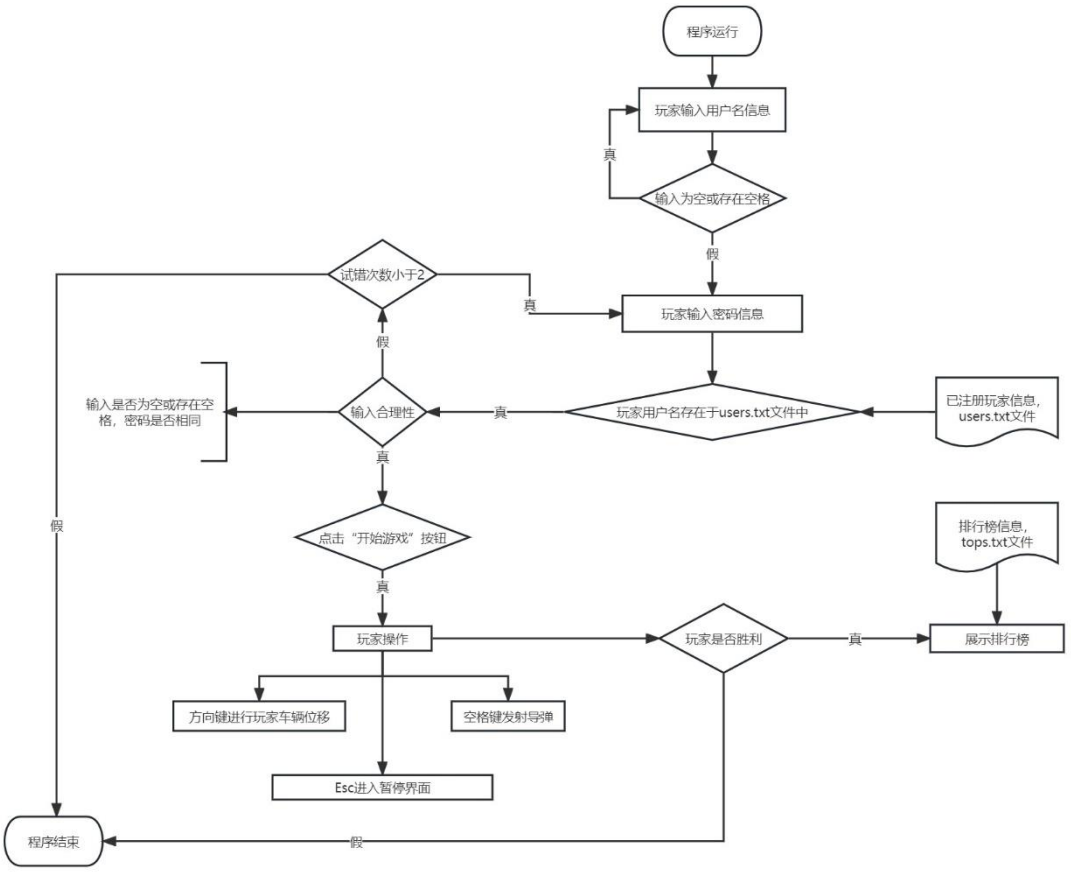
游戏交互，包括玩家初始的玩家信息输入以及后续游玩过程中对于玩家车辆的控制与操作。

（1）、玩家信息输入：游戏程序正常运行后，将弹出 InputBox 即输入消息框，提示玩家输入自己想要设置的玩家名及密码信息。随后程序将对于玩家的输入进行合理性检测（输入是否为空、是否含有空格并提示玩家合力输入）并且会进行文件操作，将玩家输入的用户名与 users.txt 文件中的已注册玩家名进行比对，若存在则对玩家输入的密码与先前注册的密码进行比对（实行密码校验，若错误，则将提示剩余错误次数并在剩余错误次数耗尽时结束游戏），若不存在则将玩家的注册信息写入文件。新玩家与老玩家进入游戏后将接收到不同的欢迎界面（欢迎第一次游玩或打破记录的鼓励）。

（2）、游玩过程中，为了实现玩家按键多键无冲，提高玩家操作的灵敏度以及避免造成程序阻塞，使用了 GetansyKeystate 函数，并对于玩家的操作，如‘W’、‘A’、‘S’、‘D’及上下左右、空格键、ESC 按键等进行逻辑判断，使玩家可以通过键盘在游戏中进行交互，按下对应的方向键将使得玩家操控车辆在地图中发生位移，按下空格键将发射导弹（可用于击碎障碍物）， 若玩家按下 ESC 按键则进入暂停界面。

（3）、鼠标操作，程序通过对 Emessage 的捕获，实现对玩家鼠标输入信息的捕获，进而实现不同菜单下对于按钮响应的实现。

（4）、游戏结束后，程序通过与 tops.txt 的交互，实现排行榜，将历代玩家的 top10 信息呈现于游戏界面上。



图表 13 游戏交互设计

3.3 其他技术点分析

3.3.1 图片显示

通过在 putimage 中添加位运算的参数，对于图片进行位运算以实现图片去背景话。为实现该操作，通过 ps 对于一张图片进行重构，绘制黑底白图与白底原图两张图片存入一个数组中后进行位运算来达到消除图片背景的操作。

3.3.2 音乐播放

音乐播放利用了 mciSendString 函数，通过#pragma comment(lib,"Winmm.lib")，实现了在 windows 系统上调用多媒体文件，进行打开，单次播放、循环播放、关闭等操作。
如下：

```
mciSendString("open .\\sound\\bk.mp3 alias bkmusic", NULL, 0, NULL);
mciSendString("play bkmusic repeat from 0", NULL, 0, NULL);
```

3.3.2 去除闪烁

在图片位移进行贴图的过程中，会造成图片留存的效应或是图片贴图闪烁、卡顿等问题。为了解决这一问题，利用 cleardevice（）、BeginBatchDraw（）和 FlushBatchDraw（）及 EndBatchDraw()等函数实现了双缓冲技术,即将将要贴图的图片先都进行存入缓存的操作，而后在调用 FlushBatchDraw（）时再同时进行同步贴图释放缓存，这样可以避免画面撕裂的情况。

4、测试

对程序从下面几部分进行了测试:

4.1 用户输入空格(或空输入)测试

用户输入空格后，将提示重新输入用户名，直至用户名通过合理性检测

图表 14 用户输入合理性测试

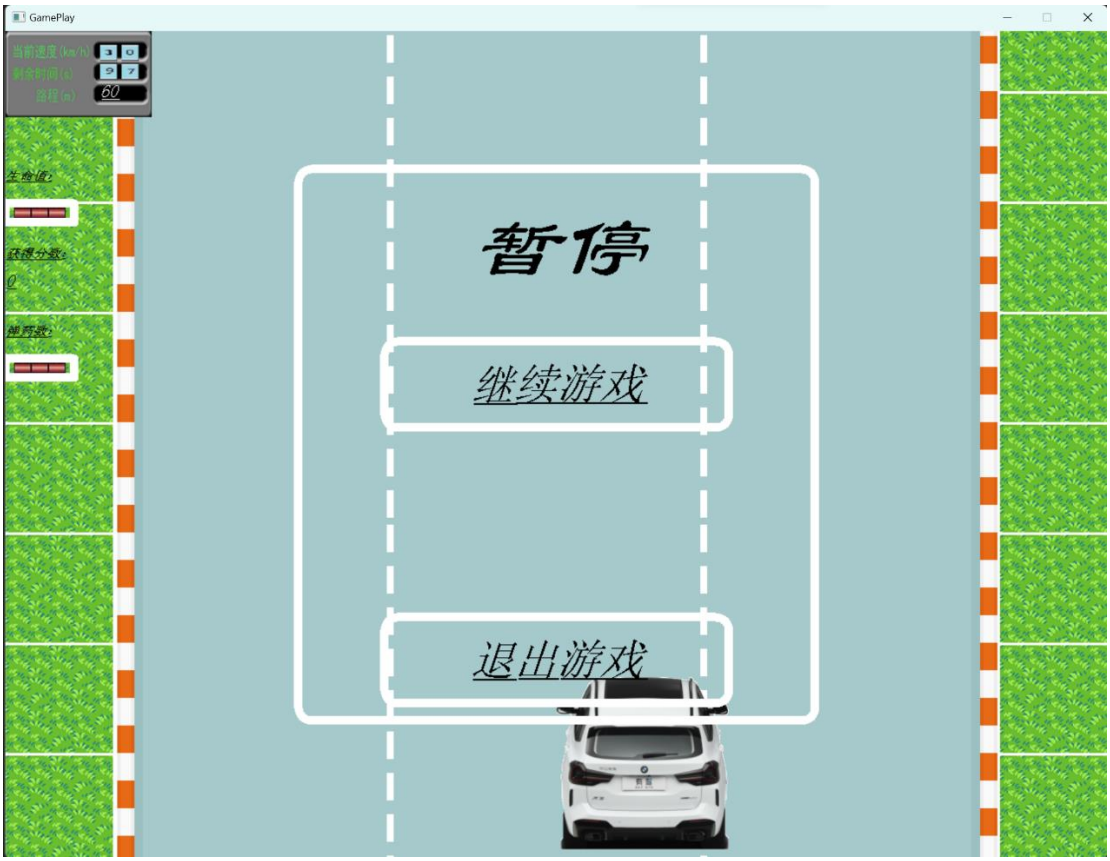
4.2 已注册用户密码输入错误测试

提示用户剩余密码尝试次数，并将在错误尝试次数耗尽时，关闭程序。

图表 15 密码正确性测试

4.3 暂停界面时停功能测试

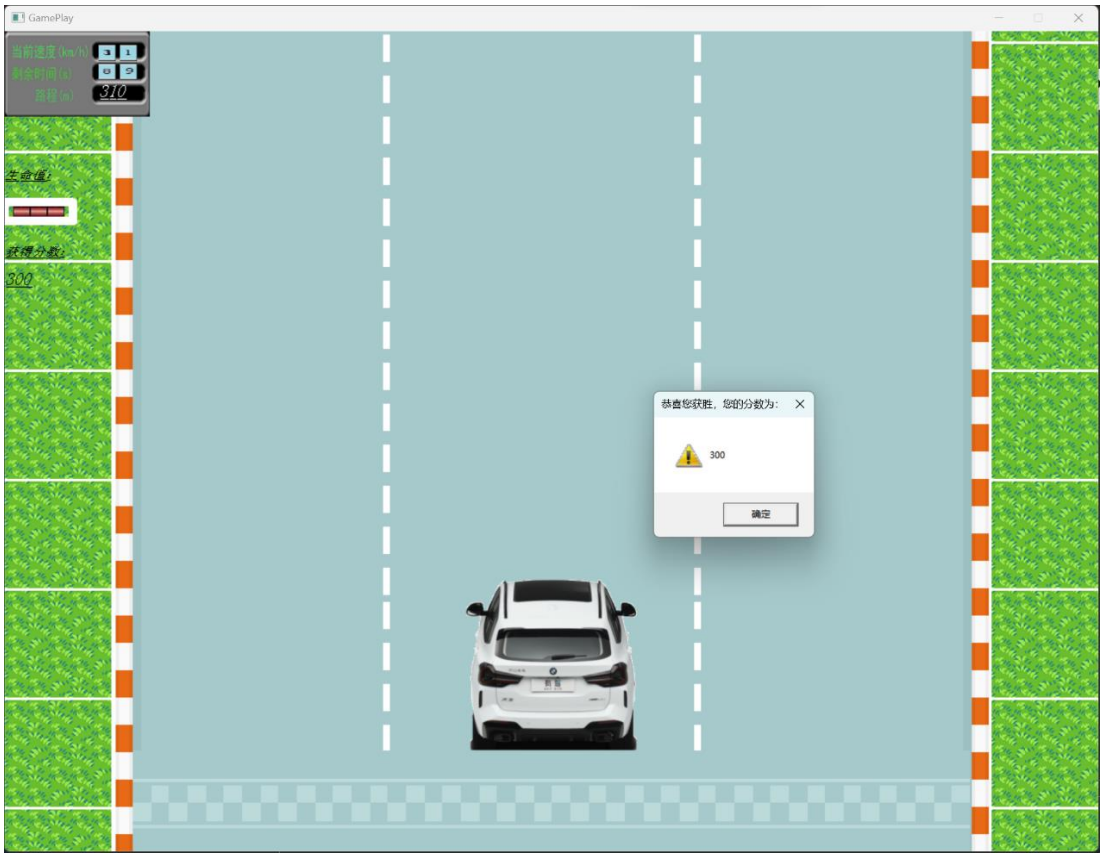
在进入暂停界面后，倒计时可以实现暂停，在再次恢复到游戏时，倒计时与刚进入暂停界面时相同



图表 16 时停功能测试

4.4 游戏通关过线画面测试

在游戏通关后，车辆将被进行居中处理，并绘制通关过线画面。



图表 17 游戏通关车辆过线测试

4.5 排行榜及胜利界面结算测试

对于玩家通关后，将要呈现的结算界面以及排行榜功能的检测。

图表 18 排行榜及胜利结算界面测试

5、用户手册

（1）、运行游戏程序.exe 文件或由游戏编写解决方案工程.sln 文件中编译运行。可点击疯狂赛车安装程序.exe 按照提示进行游戏安装游玩也可以自行点击 GamePlay 中的 x64 文件夹下 Release 文件夹内的 GamePlay.exe 进行游玩或是 GamePlay 文件夹中的 GamePlay.sln 用 visualstudio 打开后编译游玩。

（2）、游戏功能包含，通过方向键对车辆进行控制：

‘W’和‘↑’进行上位移；

‘A’和‘←’进行左位移；

‘S’和‘↓’进行下位移；

‘D’和‘→’进行右位移；

游玩过程中，将随机刷出障碍物石头，若用户所控制车辆与石头障碍物相撞，将会扣除一点血量，待血量归零，则游戏结束。

玩家可以通过空格键发射导弹，当导弹接触石头障碍物时，将击碎石头，并加分一百分。

游戏的获胜条件为通过位移操作或是攻击操作躲避石头障碍物，存活至倒计时结束。

当游戏结束时，将通过玩家的分数进行星级奖励，并展示排行榜。

（3）、应用程序运行环境的要求：Windows 系统，1GB 以上内存，1080p 以上屏幕，i5 及以上 cpu，1050ti 及以上显卡，关闭杀毒软件。

（4）、输入数据格式：需要玩家输入用户名及密码，输入内容应不为空且不含有空格，且

输入内容应符合社会主义核心价值观，轻松游戏，切勿沉迷。

祝您游玩愉快！

6 、总结提高

6.1 课设设计总结

一个学期的 c 语言课设学习，不但是对我实际 c 语言业务水平的一次考验，更是一次难得的在自我学习提升的机会。这一学期虽然经受了许多的挫折，但可谓是收获满满，我几乎自学了整个 EasyX 图形绘制库中所提供的基本函数应用，并对上学期的文件以及链表等较难掌握的知识，有了进一步的理解和实操能力。我认识到，程序的开发是需要具有前瞻性以及规划性、目的性的，在最初需要规划好自己所需要满足的客户需求以及构建数据结构的底层逻辑和实现更优解的算法。我这次的程序设计，就属于没有较好的规划自己想要实现功能的路线与两两模块间的关联性，导致代码的可维护性较差，存在一定的代码缺陷，导致后期进行了大量的优化整改，在此过程中我深刻的理解了上学期老师所讲的代码注释的重要作用，代码注释可以使我们以及他人快速了解代码结构以及函数功能等，使我们更快的锁定自己所需要的部分。

在代码的实现部分，由于时间紧迫，我有部分已经设计与完成的代码块并未应用到已经设想好的模式二中，只做到了优化与完善了模式一部分，但整体的效果已经让我感到十分满意了，也更加坚定了我学习编程的决心，在完成课设的过程，就像是在照看着自己的小孩一点点长大，让我感到十分有成就感，也感到十分的开心，谢谢老师提供的这样一次宝贵的机会。

最后的最后，课设总算是顺利完成了，在设计中的确遇到了许多许多的 bug 以及问题，但在蔡越江老师以及许多热心同学的帮助下，均逐一解决。回望这个学年，我的编程能力真是有了很大的进步，在此，我对所有帮助过我的同学以及我的蔡越江表示由衷的感谢！

6.2 对课程的一些建议

在课程初期，面对庞大的代码体系以及从未接触过的游戏开发领域，很多同学是没有头绪的，希望老师可以在以后的教学过程中，在同学们刚开始上这门课程的初期可以提供一定的引导和示范，如提供一些优秀游戏代码的 demo 或是可运行样例，供同学们有一个大致的头绪，其次是希望课程可以引入开源的思想，将历代同学们做的不同软件以及写的优秀代码发布在统一的平台上，既可以供后面的学弟学妹们学习，也可以使完成后的课设游戏发挥更大的余温。

在最后，再次由衷的感谢所有帮助过我的同学以及老师们！

7、附件：程序源代码

```
//头文件
#pragma comment(lib,"Winmm.lib")
#include <graphics.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>

//宏定义
#define LENGTH 2                    //车图片数组长度
#define WIDTH 1280                      //游戏窗口宽度(x)
#define HEIGHT 960                      //游戏窗口高度(y)
#define SIZE 200                    //车的尺寸
#define SIZE_OBSTACLE 128           //障碍物的尺寸
#define SIZE_NUM 20                     //数字的尺寸
#define SIZE_ROCKET 100                 //火箭的尺寸
#define PI 3.14159265359            //Π

//状态全局变量
bool isMenu = false;                    //是否点开过菜单
bool onFire = false;                //是否发射过火箭
bool isUp = false;                      //是否已按下过向上的按键
bool isDown = false;                    //是否已按下过向下的按键
bool isHit = false;                     //是否已发生碰撞
bool first_play;                        //是否第一次游玩

//图片全局变量
IMAGE frontCover;                       //游戏封面图片
IMAGE grass;                            //草地图片
IMAGE car_Audi[LENGTH];                     //奥迪车图片
IMAGE car_Bmw[LENGTH];                      //宝马车图片
IMAGE car_Mercedes[LENGTH];                 //奔驰车图片
IMAGE car_black[LENGTH];                //黑车图片
IMAGE garage;                           //车库图片
IMAGE map_1;                                //地图图片
IMAGE map_2;
IMAGE car_circle[LENGTH];               //旋转后汽车图片
```

```
IMAGE rock[LENGTH];                    //障碍物石头图片
IMAGE rocket_img[LENGTH];              //火箭图片
IMAGE number[10];                      //用于时间的数字组图片
IMAGE show;                            //信息框
IMAGE bomb[LENGTH];                    //爆炸图片
IMAGE life;                            //生命值图片
IMAGE success;                         //成功图片
IMAGE black;                           //黑色背景图
IMAGE star_black;                      //黑星
IMAGE star_yellow;                     //黄星
IMAGE terminalPoint;                   //终点图片
IMAGE gameOver;                        //gameover 图片
IMAGE gameWin;                         //gamewin 图片
IMAGE bomb_car[LENGTH];                //车辆爆炸图片
IMAGE site;                            //设置图片
IMAGE temp;                            //暂时存储游戏信息
IMAGE add_life;                        //增加生命值图片
IMAGE add_rocket;                      //增加弹药图片

//消息全局变量
ExMessage m;                           //定义消息变量（鼠标）

//数据全局变量
double Forward = 0;                    //车辆方向
double Rota = 360;                     //角速度(转向速度)

//用户信息
char* name;                            //用户姓名
char* password;                        //用户密码

//地图像素点信息数组
int map_information[WIDTH][HEIGHT];

//终点初始 y 坐标
int end_y = -100;

//玩家获得分数全局变量
long score = 0;

//车辆速度全局变量
int speed_show = 30;
const int speed_min = 20;              //速度最小值
const int speed_max = 98;              //速度最大值
long distance = 0;                     //路程
```

```
//获得三星的条件
const int first_star = 500;                //一星分数下限
const int second_star = 2000;       //二星分数下限
const int third_star = 5000;         //三星分数下限

//时间全局变量
time_t game_sum = 0;                       //游戏总时间
time_t game_begin = 0;                     //游戏开始时间
time_t game_progress = 0;                  //游戏进程时间
time_t menu_begin = 0;                     //开始进入菜单的时间
time_t menu_progress = 0;                  //菜单进程时间
time_t menu_sum = 0;                       //菜单总时间
time_t fire_last = 0;                      //火箭上次发射时间
time_t fire_now = 0;                       //火箭现在发射时间
time_t control_up_last = 0;                //上次向上控制时间
time_t control_down_last = 0;              //上次向下控制时间

//模式一地图可供生成障碍物的横坐标
int map_obstacle[3] = {
    //一道
    230,
    //二道
    530,
    //三道
    900,
};

//结构体
//创建按钮的结构体
typedef struct button
{
    int x;
    int y;
    int width;
    int height;
    COLORREF color;
    const char* buttonText;
}Button;

//创建车的结构体
typedef struct car
{
    int x;                                 //车的(x, y)坐标，以及速度
```

```
    int y;
    int speed;
    time_t last;                    //上次碰撞的时间
    bool life;                      //是否存活
}Car;

//创建障碍物的结构体
typedef struct obstacle {
    int x;
    int y;
    int speed;
    time_t last;                    //上次时间
    bool status;
}Obstacle;

//创建地图的结构体
typedef struct map {
    int x;
    int y;
    int speed;
}Map;

//创建火箭的结构体
typedef struct rocket {
    int x;
    int y;
    int speed;
    bool status;                    //火箭的状态，true 为可用，false 为不可用
}Rocket;

//创建排行榜链表
typedef struct node {
    int no;
    char name[50];
    int score;
    struct node* next;
}TOPNODE;

//函数声明
int main(void);                                                         // 主 函
数
Button* createButton(int x, int y, int width, int height, COLORREF color,
    const char* buttonText);                                            // 按 钮 对 象
的创建
```

```
void drawButton(Button* button, bool title);                    //绘制按钮
( title 用于调整字体)
Car* createCar(int x, int y, int speed, bool life, time_t last);    //车对象的创建
void loadImage();                                               // 加载
图片
bool mouseDetect(Button* button, ExMessage m);                  // 判断
鼠标点击事件是否发生在按钮边框范围内
void drawFrontCover();                                          // 创建
游戏封面以及处理主界面按钮点击事件
int drawCarSelection();                                         // 创建
选车画面
void controlCar();                                              // 控制
汽车
int pointTsm(int x, int y, int wide, int high);                 //越界检测
void edgeDetection(Car* car);                                   // 边缘
检测
bool canCircle(Car* car);                                       //判断是否
可以旋转
void onUp(Car* car);
void onDown(Car* car);
void onLeft(Car* car);
void onRight(Car* car);
void mode_1();                                                  //
模式一
Obstacle* createObstacle(int x, int y, int speed, bool status,
    time_t last);                                               // 障碍
对象的创建
Map* createMap(int x, int y, int speed);                        //地图对象的创
建
Rocket* createRocket(int x, int y, int speed, bool status);     //火箭对象的创
建
void pauseInterface();                                          // 暂停
界面
char* login_name();                                             //
输入用户姓名
bool haveSpace(char* input);                                    //检测用户
输入是否含有空格
char* login_password();                                         // 输入
用户密码
void checkUsers(char* name);                                    // 检测
用户是否存在
void PutNumber(long x, long y, long number, int TextSize = 10,
    int TextProportion = 2, int TextInterval = 0,
    COLORREF tc = 0x000000, LPCTSTR font = "宋体");              //    将
```

```
long 类型数字呈现在画布上
void winInterface();                                                    //获胜界面
void loseInterface();                                                   // 失 败
界面
TOPNODE* loadNode();                                                    // 读 取
排行榜
void showNode();                                                        // 展 示
排行榜
void changeNode(TOPNODE* h);                                            // 改 变
排行榜链表
TOPNODE* reNode(TOPNODE* h);                                            // 重 构
排行榜顺序
void rewriteNode(TOPNODE* h);                                           // 重 写
排行榜
void node_main();                                                       // 排 行
榜链表主函数

int main(void)
{
    loadImage();                                        //加载图片
    initgraph(WIDTH, HEIGHT);                           //初始化游戏窗口(1280 * 960)
    drawFrontCover();
    system("pause");
    closegraph();
}

//按钮对象的创建
Button* createButton(int x, int y, int width, int height, COLORREF color, const char* buttonText) {
    Button* button = (Button*)malloc(sizeof(Button) * 1);

    button->x = x;
    button->y = y;
    button->width = width;
    button->height = height;
    button->color = color;
    button->buttonText = buttonText;

    return button;
}

//绘制按钮( title 用于调整字体)
void drawButton(Button* button, bool title) {
    int width, height;                                  //用于计算文字居中显示所需
距离边框的宽与高的变量
```

```
    setlinecolor(WHITE);                                    //设置按钮边框线条颜色
    setbkmode(TRANSPARENT);                                 //设置文字背景透明
    settextcolor(button->color);                            //设置文字颜色

    //字符串内容是否用于标题，将选择不同的字体设置
    if (title == true) {
        settextstyle(120, 0, "隶书", 5, 0, 10, true, true, false);
        setlinestyle(PS_DASHDOT, 10);
    }
    else {
        settextstyle(50, 0, "楷书");
        setlinestyle(PS_SOLID, 10);
    }

    //绘制按钮
    roundrect(button->x, button->y,
        button->x + button->width, button->y + button->height, 30, 30);      //绘制圆角矩形
边框
    width = button->width / 2 - textwidth(button->buttonText) / 2;           //文字距左边框
的距离
    height = button->height / 2 - textheight(button->buttonText) / 2;        //文字距上边框
的高度
    outtextxy(button->x + width, button->y + height, button->buttonText);    //输出文字内容

    return;
}

//判断鼠标点击事件是否发生在按钮边框范围内
bool mouseDetect(Button* button, ExMessage m) {
    if (m.x >= button->x
        && m.x <= (button->x + button->width)
        && m.y >= button->y
        && m.y <= (button->y + button->height))
        return true;
    else
        return false;
}

//创建游戏封面以及处理主界面按钮点击事件
void drawFrontCover() {
    Button* title, * start, * instruction, * quit;                                //定义
按钮结构体指针
    //loadimage(&frontCover, (".\\image\\封面.jpg"), NULL);
    putimage(0, 0, &frontCover);                                                  //
```

```
呈现封面背景图
    score = 0;
    //创建按钮对象
    title = createButton(350, 50, 580, 150, RGB(237, 255, 83), "赛车游戏");                // 标 题
对象
    start = createButton(500, 300, 280, 80, BLACK, "开始游戏");                                //
开始对象
    instruction = createButton(500, 430, 280, 80, RGB(51, 235, 149), "玩法介绍"); //介绍对象
    quit = createButton(500, 560, 280, 80, RGB(3, 0, 255), "退出游戏");                    // 退 出
对象

    //绘制按钮
    drawButton(title, true);            //创建标题按钮
    drawButton(start, false);           //创建“开始游戏”按钮
    drawButton(instruction, false);         //创建“玩法介绍”按钮
    drawButton(quit, false);            //创建“退出游戏"按钮

    //用户信息输入
    name = login_name();
    checkUsers(name);

    //背景音乐
    //循环播放
    //PlaySound(".\\sound\\bk.wav", NULL, SND_ASYNC | SND_FILENAME | SND_LOOP);
    mciSendString("open .\\sound\\bk.mp3 alias bkmusic", NULL, 0, NULL);
    mciSendString("play bkmusic repeat from 0", NULL, 0, NULL);
    //检测鼠标对于封面按钮的点击事件
    while (true) {
        m = getmessage(EX_MOUSE);           //获取鼠标信息
        switch (m.message) {
        case WM_LBUTTONDOWN:
            //检测是否按下“开始游戏”按钮
            if (mouseDetect(start, m)) {
                printf("开始游戏");
                cleardevice();
                //controlCar();
                mode_1();
            }
            //检测是否按下“玩法介绍”按钮
            if (mouseDetect(instruction, m)) {
                MessageBox(NULL,
                    "用户可以通过'W','S','A','D' 或 ↑ ↓ ← → 按键分别控制汽车的前
后左右\n 按空格键发射导弹，接触石头后可打碎石头，并加 100 分\n 去挑战更高分吧，勇
士！",
```

```
                    "玩法介绍", MB_ICONEXCLAMATION | MB_SETFOREGROUND);
            }
            //检测是否按下“退出游戏”按钮
            if (mouseDetect(quit, m)) {
                exit(-1);
            }
        }
    }
}

//加载图片
void loadImage() {
    //封面图片
    loadimage(&frontCover, (".\\image\\封面.jpg"), NULL);

    //车库图片
    loadimage(&garage, ".\\image\\车库.jpg", WIDTH, HEIGHT);

    //奥迪车图片
    loadimage(&car_Audi[0], ".\\image\\奥迪 A4L1.png", SIZE, SIZE);
    loadimage(&car_Audi[1], ".\\image\\奥迪 A4L2.png", SIZE, SIZE);

    //宝马车图片
    loadimage(&car_Bmw[0], ".\\image\\宝马 X31.png", SIZE, SIZE);
    loadimage(&car_Bmw[1], ".\\image\\宝马 X32.png", SIZE, SIZE);

    //奔驰车图片
    loadimage(&car_Mercedes[0], ".\\image\\奔驰 E 级 1.png", SIZE, SIZE);
    loadimage(&car_Mercedes[1], ".\\image\\奔驰 E 级 2.png", SIZE, SIZE);

    //黑车图片
    loadimage(&car_black[0], ".\\image\\car_black1.png", NULL);
    loadimage(&car_black[1], ".\\image\\car_black2.png", NULL);

    //地图图片
    loadimage(&map_1, ".\\image\\map1.png", NULL);
    loadimage(&map_2, ".\\image\\地图 1.png", NULL);

    //障碍物石头图片
    loadimage(&rock[0], ".\\image\\石头 1.png", SIZE_OBSTACLE, SIZE_OBSTACLE);
    loadimage(&rock[1], ".\\image\\石头 2.png", SIZE_OBSTACLE, SIZE_OBSTACLE);

    //火箭图片
    loadimage(&rocket_img[0], ".\\image\\火箭 1.png", SIZE_ROCKET, SIZE_ROCKET);
```

```
loadimage(&rocket_img[1], ".\\image\\火箭 2.png", SIZE_ROCKET, SIZE_ROCKET);

//数字图片 0~9
loadimage(&number[0], ".\\image\\0.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[1], ".\\image\\1.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[2], ".\\image\\2.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[3], ".\\image\\3.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[4], ".\\image\\4.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[5], ".\\image\\5.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[6], ".\\image\\6.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[7], ".\\image\\7.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[8], ".\\image\\8.jpg", SIZE_NUM, SIZE_NUM);
loadimage(&number[9], ".\\image\\9.jpg", SIZE_NUM, SIZE_NUM);

//信息栏图片
loadimage(&show, ".\\image\\show.png", NULL);

//爆炸图片
loadimage(&bomb[0], ".\\image\\爆炸 1.png", SIZE_OBSTACLE, SIZE_OBSTACLE);
loadimage(&bomb[1], ".\\image\\爆炸 2.png", SIZE_OBSTACLE, SIZE_OBSTACLE);

//车辆爆炸图片
loadimage(&bomb_car[0], ".\\image\\爆炸 1.png", SIZE, SIZE);
loadimage(&bomb_car[1], ".\\image\\爆炸 2.png", SIZE, SIZE);

//生命值图片
loadimage(&life, ".\\image\\life.bmp", NULL);

//结算成功图片
loadimage(&success, ".\\image\\flag.jpg", NULL);

//黑色背景图片
loadimage(&black, ".\\image\\black.png", NULL);

//黑星图片
loadimage(&star_black, ".\\image\\star1.bmp", NULL);

//黄星图片
loadimage(&star_yellow, ".\\image\\star2.bmp", NULL);

//终点图片
loadimage(&terminalPoint, ".\\image\\终点.png", NULL);

//gamewin 图片
```

```
	loadimage(&gameWin, ".\\image\\gamewin.bmp", NULL);

	//gameover 图片
	loadimage(&gameOver, ".\\image\\gameover.bmp", WIDTH, HEIGHT);

	//设置图片
	loadimage(&site, ".\\image\\site.jpg", NULL);

	//增加生命值图片
	loadimage(&add_life, ".\\image\\加生命值.png", NULL);

	//增加弹药图片
	loadimage(&add_rocket, ".\\image\\加弹药数.png", NULL);
}

//车对象的创建
Car* createCar(int x, int y, int speed, bool life, time_t last) {
	Car* car = (Car*)malloc(sizeof(Car) * 1);

	car->x = x;
	car->y = y;
	car->speed = speed;
	car->life = life;
	car->last = last;

	return car;
}

//障碍对象的创建
Obstacle* createObstacle(int x, int y, int speed, bool status, time_t last) {
	Obstacle* obstacle = (Obstacle*)malloc(sizeof(Obstacle) * 1);

	obstacle->x = x;
	obstacle->y = y;
	obstacle->speed = speed;
	obstacle->status = status;
	obstacle->last = last;

	return obstacle;
}

//地图对象的创建
Map* createMap(int x, int y, int speed) {
	Map* map = (Map*)malloc(sizeof(Map) * 1);
```

```
	map->x = x;
	map->y = y;
	map->speed = speed;

	return map;
}

//火箭对象的创建
Rocket* createRocket(int x, int y, int speed, bool status) {
	Rocket* rocket = (Rocket*)malloc(sizeof(Rocket) * 1);

	rocket->x = x;
	rocket->y = y;
	rocket->speed = speed;
	rocket->status = status;

	return rocket;
}

//检测用户输入是否含有空格
bool haveSpace(char* input) {
	bool isSpace = false;
	//遍历字符串数组
	for (int i = 0; i < strlen(input); i++) {
		if (input[i] == ' ') {
			isSpace = true;
		}
	}

	return isSpace;
}

//输入用户姓名
char* login_name() {
	char* name;
	char title[30] = "欢迎游玩！这是游戏的登陆界面";		//提示输入框的标题
	char clue[40] = "请输入您的用户名(不能含有空格)";		//提示输入框的输入
	//动态分配名字空间
	name = (char*)malloc(sizeof(char) * 50);
	while (1) {
		InputBox(name, 20, clue, title);

		//如果用户的输入含有空格
```

```
        if (haveSpace(name))
        {
            /*
            后两个参数的作用:
            MB_ICONQUESTION      显示  Warning Message  图标
            MB_SETFOREGROUND   标志确保消息框成为前台窗口。
            */
            //弹出窗口提示输入的内容
            MessageBox(NULL,
                "请正确输入用户名",
                "请正确输入用户名", MB_ICONEXCLAMATION | MB_SETFOREGROUND);
            continue;
        }

        //如果用户未输入
        if (*name == '\0')
        {
            MessageBox(NULL,
                "您未输入用户名",
                "您未输入用户名", MB_ICONEXCLAMATION | MB_SETFOREGROUND);
            continue;
        }
        break;
    }

    return name;
}

//输入用户姓名
char* login_password() {
    char* password;
    char title[30] = "登录密码";                                    //提示输入框的标题
    char clue[40] = "请输入您的密码(不能含有空格)";          //提示输入框的输入
    //密码动态分配空间
    password = (char*)malloc(sizeof(char) * 50);
    while (1) {
        InputBox(password, 20, clue, title);

        //如果用户的输入含有空格
        if (haveSpace(password))
        {
            /*
            后两个参数的作用:
            MB_ICONQUESTION      显示  Warning Message  图标
```

```
            MB_SETFOREGROUND    标志确保消息框成为前台窗口。
            */
            //弹出窗口提示输入的内容
            MessageBox(NULL,
                "请正确输入密码",
                "请正确输入密码", MB_ICONEXCLAMATION | MB_SETFOREGROUND);
            continue;
        }

        //如果用户未输入
        if (*password == '\0')
        {
            MessageBox(NULL,
                "您未输入密码",
                "您未输入密码", MB_ICONEXCLAMATION | MB_SETFOREGROUND);
            continue;
        }
        break;
    }

    return password;
}

// 检测用户是否存在
void checkUsers(char* name)
{
    FILE* fp;
    errno_t err;                                                    //int 重定义，用于
后面的错误鉴定
    //fopen_s 的返回值是相应的错误代码，有助于排查问题(打开文件成功返回 0，失败返
回非 0)
    err = fopen_s(&fp, ".\\users.txt", "a+");                       //打开 users 文件，并执行
读写操作

    //如果文件读写没有错误
    if (err == 0) {
        //将文件指针置于文件起始位置，且偏移量为 0
        fseek(fp, 0, SEEK_SET);
        char check_password[50];
        // 检测用户名单是否有当前用户，如果没有，添加用户并且欢迎，如果有，不再添
加用户，并且欢迎用户回来
        int flag;
        //当文件指针不在文件结尾时，进行循环
        while (!(feof(fp)))
```

```
{
    char check_name[50];
    fscanf(fp, "%s", check_name);
    fscanf(fp, "%s", check_password);
    if (!(strcmp(check_name, name)))
    {
        flag = 1;

        break;          // 一旦发现有本玩家立刻退出循环，防止 flag 被刷新
    }
    else
        flag = 0;
}

//如果玩家第一次游玩游戏
if (flag == 0)
{
    first_play = true;
    password = login_password();
    fprintf(fp, "%s %s\n", name, password);
    char string[70];
    sprintf_s(string, "用户:%s  欢迎游玩！", name);
    MessageBox(NULL,
        "欢迎来游玩 CrazzyHe 的第一个游戏\n，准备好挑战疯狂赛车游戏了吗？
\n 点击主界面的第二个栏目，有更详细的游戏玩法解释哦！",
        string, MB_SETFOREGROUND);

}

//如果玩家是非第一次游玩游戏
else
{
    first_play = false;
    int flag = 0;    //用于密码错误次数检测
    while (1)
    {
        password = login_password();

        // 如果密码输入正确
        if (!(strcmp(password, check_password)))
        {
            char string[70];
            sprintf_s(string, "用户:%s  欢迎回来", name);
            MessageBox(NULL,
```

```
                    "欢迎回来！准备好刷新记录了吗？",
                    string, MB_SETFOREGROUND);
                break;
            }

            // 如果密码输入错误
            else
            {
                if (flag == 0)
                {
                    MessageBox(NULL,
                        "密码错误，你还有 1 次机会重新输入密码\n 否则程序将关
闭",
                        "密码错误", MB_SETFOREGROUND);
                    flag = 1;
                    continue;
                }
                else
                {
                    MessageBox(NULL,
                        "密码错误，程序关闭",
                        "密码错误", MB_SETFOREGROUND);
                    exit(-1);
                }
            }
        }
    }
    //如果写入文件错误
    else if (err != 0) {
        MessageBox(NULL, "文件操作错误",
            "文件操作错误，请检查 users.txt 文件后，重试", MB_SETFOREGROUND);
        exit(-1);
    }
    //关闭文件指针，避免后续程序运行造成文件损坏
    fclose(fp);
}

//创建选车画面
int drawCarSelection() {
    Button* chose_Audi, * chose_Bmw, * chose_Mercedes, * success;
    int selection;
    bool choice_car = false;
```

```
    //贴图
    putimage(0, 0, &garage);
    putimage(170, 220, &car_Audi[0], NOTSRCERASE);
    putimage(170, 220, &car_Audi[1], SRCINVERT);
    putimage(540, 220, &car_Bmw[0], NOTSRCERASE);
    putimage(540, 220, &car_Bmw[1], SRCINVERT);
    putimage(910, 220, &car_Mercedes[0], NOTSRCERASE);
    putimage(910, 220, &car_Mercedes[1], SRCINVERT);
    outtextxy(450, HEIGHT / 2 + 250, "请选择您的坐骑");

    //按钮结构体初始化
    chose_Audi = createButton(150, 470, SIZE + 20, SIZE - 100, RGB(255, 165, 165), "选择奥迪");
    chose_Bmw = createButton(520, 470, SIZE + 20, SIZE - 100, RGB(255, 165, 165), "选择宝马");
    chose_Mercedes = createButton(890, 470, SIZE + 20, SIZE - 100, RGB(255, 165, 165), "选择奔
驰");
    success = createButton(520, 700, SIZE + 20, SIZE - 100, RGB(255, 165, 165), "选车成功");

    //绘制按钮结构体
    drawButton(chose_Audi, false);
    drawButton(chose_Bmw, false);
    drawButton(chose_Mercedes, false);

    //鼠标事件循环
    while (true) {
        m = getmessage(EX_MOUSE);       //获取鼠标信息
        switch (m.message) {
        case WM_LBUTTONDOWN:
            //检测是否按下对应的选车按钮

            //选择奥迪车
            if (mouseDetect(chose_Audi, m)) {
                cleardevice();
                putimage(0, 0, &garage);
                putimage(520, 400, &car_Audi[0], NOTSRCERASE);
                putimage(520, 400, &car_Audi[1], SRCINVERT);
                //单次触发选车成功音效
                PlaySound(".\\sound\\light.wav", NULL, SND_ASYNC | SND_FILENAME);
                selection = 1;
                choice_car = true;
                break;
            }

            //选择宝马车
            if (mouseDetect(chose_Bmw, m)) {
```

```
                cleardevice();
                putimage(0, 0, &garage);
                putimage(520, 400, &car_Bmw[0], NOTSRCERASE);
                putimage(520, 400, &car_Bmw[1], SRCINVERT);
                PlaySound(".\\sound\\light.wav", NULL, SND_ASYNC | SND_FILENAME);
                selection = 2;
                choice_car = true;
                break;
            }

            //选择奔驰车
            if (mouseDetect(chose_Mercedes, m)) {
                cleardevice();
                putimage(0, 0, &garage);
                putimage(520, 400, &car_Mercedes[0], NOTSRCERASE);
                putimage(520, 400, &car_Mercedes[1], SRCINVERT);
                PlaySound(".\\sound\\light.wav", NULL, SND_ASYNC | SND_FILENAME);
                selection = 3;
                choice_car = true;
                break;
            }
        }
        if (choice_car == true) {
            break;
        }
    }
    drawButton(success, false);
    Sleep(3000);
    return selection;
}

//将 long 类型数字呈现在画布上
void PutNumber(
    long x,                                         //输出位置
    long y,
    long number,                                    //需要转换的数
字
    int TextSize,                                   //文字的尺寸
    int TextProportion,                             //文字的高宽比
    int TextInterval,                               //文字之间的间
隔
    COLORREF tc,                                    //文字颜色(默认
为黑)
    LPCTSTR font                                    //字体设置
```

```
)
{
    int max;
    long int one_data;                                          //当前位数字

    //文字属性设置
    settextcolor(tc);                                           //设置文字颜色
    setbkmode(TRANSPARENT);                                     //设置文字背景为透明色
    settextstyle(TextSize * TextProportion, TextSize, font);    //设置字体与字的大小
    if (TextInterval == 0)                                      //如果间隔为 0，则自动调整间隔
        TextInterval = TextSize;
    for (max = 0; max < 10; max++)                              //获得数字位数 max
        if (number / pow(10, max) < 10)
            break;
    while (number >= 0) {                                       //按位输出数字
        if (max < 0 || x > getwidth())                          //如果位数小于 0 或者 x 超过程序边界
            return;
        one_data = number / (long)pow(10, max);
        outtextxy(x, y, (char)(one_data + 48));
        number -= one_data * (long)pow(10, max);
        --max;
        x += TextInterval;
    }
}

//获胜界面
void winInterface() {
    char* show_score;
    Button* returnMain, * quit;
    show_score = (char*)malloc(sizeof(char) * 100);
    sprintf(show_score, "%ld", score);
    MessageBox(NULL,
        show_score,
        "恭喜您获胜，您的分数为：", MB_ICONEXCLAMATION | MB_SETFOREGROUND);
    putimage(0, 200, &success);
    settextcolor(WHITE);
    settextstyle(50, 0, "隶书");
    mciSendString("close all", NULL, 0, NULL);
    mciSendString("open .\\sound\\gamewin.mp3 alias gamewinmusic", NULL, 0, NULL);
    mciSendString("open .\\sound\\light.mp3 alias lightmusic", NULL, 0, NULL);
```

```
mciSendString("play gamewinmusic from 0", NULL, 0, NULL);
BeginBatchDraw();
//静态胜利界面展示
for (int i = -1280; i <= 0; i++) {
	putimage(i, 200 + success.getheight(), &black);
	outtextxy(i + 20, 250 + success.getheight(), "玩家：");
	outtextxy(i + 200, 250 + success.getheight(), name);
	//第三颗黑星
	putimage(i + 1280 - star_black.getwidth(),
		200 + 2 * success.getheight() - star_black.getheight(),
		&star_black);
	//第二颗黑星
	putimage(i + 1280 - star_black.getwidth() * 2,
		200 + 2 * success.getheight() - star_black.getheight(),
		&star_black);
	//第一颗黑星
	putimage(i + 1280 - star_black.getwidth() * 3,
		200 + 2 * success.getheight() - star_black.getheight(),
		&star_black);
	FlushBatchDraw();
}
EndBatchDraw();
Sleep(1000);
//第一颗黄星
if (first_star <= score) {
	mciSendString("play lightmusic from 0", NULL, 0, NULL);
	putimage(1280 - star_yellow.getwidth() * 3,
		200 + 2 * success.getheight() - star_yellow.getheight(),
		&star_yellow);
	Sleep(1000);
}
//第二颗黄星
if (second_star <= score) {
	mciSendString("play lightmusic from 0", NULL, 0, NULL);
	putimage(1280 - star_yellow.getwidth() * 2,
		200 + 2 * success.getheight() - star_yellow.getheight(),
		&star_yellow);
	Sleep(1000);
}
//第三颗黄星
if (third_star <= score) {
	mciSendString("play lightmusic from 0", NULL, 0, NULL);
	putimage(1280 - star_yellow.getwidth(),
		200 + 2 * success.getheight() - star_yellow.getheight(),
```

```
                &star_yellow);
            Sleep(1000);
        }
        node_main();
        //创建按钮对象
        returnMain = createButton(0, 800, 200, 100, BLACK, "主界面");
        quit = createButton(1080, 800, 200, 100, BLACK, "退出游戏");

        //绘制按钮
        drawButton(returnMain, false);
        drawButton(quit, false);
        FlushBatchDraw();

        //检测鼠标对于暂停页面按钮的点击事件
        while (true) {
            m = getmessage(EX_MOUSE);                                    //
获取鼠标信息
            switch (m.message)
            {
            case WM_LBUTTONDOWN:
                //检测是否按下“返回主界面”按钮
                if (mouseDetect(returnMain, m)) {
                    cleardevice();
                    drawFrontCover();
                }
                //检测是否按下“退出游戏”按钮
                if (mouseDetect(quit, m)) {
                    exit(-1);
                }
            }
        }
}

//读取排行榜
TOPNODE* loadNode() {
        TOPNODE* h, * r;
        TOPNODE* p;
        FILE* top;
        errno_t err;                                                    //int 重定义,用于
后面的错误鉴定
        h = NULL;
        r = NULL;
        //fopen_s 的返回值是相应的错误代码，有助于排查问题(打开文件成功返回 0，失败返
回非 0)
```

```
    err = fopen_s(&top, ".\\tops.txt", "r");                    //打开 tops 文件，并执行读写操作
    //如果文件读写没有错误
    if (err == 0) {
        //将文件指针置于文件起始位置，且偏移量为 0
        fseek(top, 0L, SEEK_SET);
        for (int i = 0; i < 10; i++) {
            p = (TOPNODE*)malloc(sizeof(TOPNODE) * 1);
            fscanf_s(top, "%d", &p->no);
            fscanf_s(top, "%s", p->name, _countof(p->name));
            fscanf_s(top, "%d", &p->score);
            //尾插法，创建链表，读取排行榜信息
            p->next = NULL;
            if (h == NULL) {
                h = p;
                r = p;
            }
            else {
                r->next = p;
                r = p;
            }
        }
    }
    //如果读取排行榜文件错误
    else if (err != 0) {
        MessageBox(NULL, "文件操作错误",
            "文件操作错误，请检查 tops.txt 文件后，重试", MB_SETFOREGROUND);
        exit(-1);
    }
    fclose(top);
    //对于链表功能实现的测试，其功能已能正常实现
    //测试代码：
    /*
    TOPNODE* q = h;
    while (q != NULL) {
        printf("%d %s %d \n", q->no, q->name, q->score);
        q = q->next;
    }
    */
    return h;
}

// 展示排行榜
void showNode()
```

```
{

    char tops[500] = "No\tName\t\tPlayer's Best Score\n";                    //存储排行
榜前十数据
    TOPNODE* h;
    TOPNODE* q;
    h = loadNode();
    q = h;
    for (int i = 0; i < 10; i++)
    {
        char top[40];
        sprintf_s(top, "%d\t%s\t\t%d\n", q->no, q->name, q->score);
        strcat_s(tops, top);
        q = q->next;
    }
    MessageBox(NULL, tops,
        "********************  排  行  榜  名  单  **********************",
MB_SETFOREGROUND);
}

//重写排行榜
void rewriteNode(TOPNODE* h) {
    /*
    在调用该函数之前应确保 tops.txt 文件已处于关闭状态，无指针索引
    否则，可能会出现 stream.valid()的问题，tops 指针丢失
    */
    bool is_top;
    int temp_no;
    int temp_score;
    char temp_name[50];
    TOPNODE* p;
    FILE* tops;
    errno_t err;
    p = (TOPNODE*)malloc(sizeof(TOPNODE) * 1);
    p = h;
    is_top = false;
    err = fopen_s(&tops, ".\\tops.txt", "w");
    fseek(tops, 0L, SEEK_SET);
    while (p != NULL) {
        fprintf_s(tops, "%d %s %d\n", p->no, p->name, p->score);
        p = p->next;
    }
    fclose(tops);
    err = fopen_s(&tops, ".\\tops.txt", "r");
```

```
	fseek(tops, 0L, SEEK_SET);
	for (int i = 0; i < 10; i++)
	{
		fscanf_s(tops, "%d", &temp_no);
		fscanf_s(tops, "%s", temp_name, _countof(temp_name));
		fscanf_s(tops, "%d", &temp_score);
		if (strcmp(name, temp_name) == 0)
		{
			is_top = true;
			break;
		}
	}
	//玩家上榜
	if (is_top == true)
	{
		MessageBox(NULL, "恭喜上榜，进入前十！",
			"===============恭喜上榜!!!!===============", MB_SETFOREGROUND);
	}
	else
	{
		MessageBox(NULL, "恭喜通关!!!!很遗憾，你没能登上排行榜，再接再厉，加油!
",
			"===============恭喜通关!!!!===============", MB_SETFOREGROUND);
	}
	fclose(tops);
}

//重构排行榜顺序
TOPNODE* reNode(TOPNODE* h) {
	TOPNODE* p, * f;
	TOPNODE temp;
	int i = 1;
	p = (TOPNODE*)malloc(sizeof(TOPNODE) * 1);
	p = h;
	//链表排序
	while (p->next != NULL) {
		f = (TOPNODE*)malloc(sizeof(TOPNODE) * 1);
		f = p->next;
		while (f != NULL) {
			if (p->score < f->score) {
				temp = *p;
				*p = *f;
				*f = temp;
				temp.next = p->next;
```

```
                p->next = f->next;
                f->next = temp.next;
            }
            f = f->next;
        }
        p = p->next;
    }
    p = h;
    while (p != NULL)
    {
        p->no = i;
        p = p->next;
        i++;
    }
    //手动打乱 tops.txt 内部排行榜排序，已验证，排序函数已可以正常运行
    /*
    p = h;
    while (p != NULL) {
        printf("%d %s %d\n", p->no, p->name, p->score);
        p = p->next;
    }
    */
    return h;
}

// 改变排行榜链表
void changeNode(TOPNODE* h)
{
    TOPNODE* p, * f;
    int flag;
    // 检测用
    char temp_name[50];
    // 更新用
    int temp_no;
    int temp_score;
    //为链表节点 p 动态分配数组空间
    p = (TOPNODE*)malloc(sizeof(TOPNODE) * 1);
    flag = 0;
    // 是否已有本玩家检测
    FILE* tops;
    errno_t err;
    err = fopen_s(&tops, ".\\tops.txt", "r");
    //将文件指针置于文件头部
    fseek(tops, 0L, SEEK_SET);
```

```
    if (err == 0) {
        while (!(feof(tops))) {
            fscanf_s(tops, "%d", &temp_no);
            fscanf_s(tops, "%s", temp_name, _countof(temp_name));
            fscanf_s(tops, "%d", &temp_score);
            if (strcmp(temp_name, name) == 0) {
                flag = 1;                                                    // 用 户
已存在
                break;
            }
        }
        //如果玩家已经存在
        if (flag == 1) {
            if (score > temp_score) {
                p = h;
                while (p != NULL) {
                    if (strcmp(temp_name, p->name) == 0) {
                        p->score = score;
                        break;
                    }
                    p = p->next;
                }
                fclose(tops);
                h = reNode(h);
                rewriteNode(h);
                MessageBox(NULL, "恭喜你刷新了个人历史记录!!!! ",
                    "=============== 恭 喜 破 纪 录 ！ ！ ！ ！ ===============",
MB_SETFOREGROUND);
            }
            else {
                MessageBox(NULL, "恭喜通关!!!! 很遗憾，你没能刷新个人历史记录，再
接再厉，加油！ ",
                    "=============== 恭 喜 通 关 ！ ！ ！ ！ ===============",
MB_SETFOREGROUND);
            }
        }
        else if (flag == 0) {
            fclose(tops);
            p = h;
            while (p->next != NULL) {
                p = p->next;
            }
            TOPNODE* f;
            f = (TOPNODE*)malloc(sizeof(TOPNODE) * 1);
```

```
                memset(f->name, 0, 50);
                strcat(f->name, name);
                f->score = score;
                f->next = NULL;
                p->next = f;
                MessageBox(NULL, "恭喜通关!!!! 很遗憾，你之前还没有登上排行榜!",
                    "===============  恭  喜  通  关  ！  ！  ！  ！  ===============",
MB_SETFOREGROUND);
                h = reNode(h);
                rewriteNode(h);
            }
        }
        return;
}

// 排行榜链表主程序
void node_main()
{
        TOPNODE* h = NULL;
        h = loadNode();
        h = reNode(h);
        changeNode(h);
        showNode();
}

//暂停界面
void pauseInterface() {
        Button* goOn, * site, * quit;
        ExMessage m;
        bool play = false;
        menu_begin = time(NULL);
        isMenu = true;
        setbkmode(TRANSPARENT);                                                    //
设置文字背景透明
        settextcolor(BLACK);                                                       // 设 置
文字颜色
        settextstyle(100, 0, "隶书");

        //创建按钮对象
        goOn = createButton(440, 360, 400, 100, BLACK, "继续游戏");
        //site = createButton(440, 520, 400, 100, BLACK, "游戏设置");
        quit = createButton(440, 680, 400, 100, BLACK, "退出游戏");
        roundrect(340, 160, 940, 800, 30, 30);
        outtextxy(540, 200, "暂停");
```

```
    //绘制按钮
    drawButton(goOn, false);
    //drawButton(site, false);
    drawButton(quit, false);
    FlushBatchDraw();

    //检测鼠标对于暂停页面按钮的点击事件
    while (!play) {
        m = getmessage(EX_MOUSE);                                    //
获取鼠标信息
        switch (m.message)
        {
        case WM_LBUTTONDOWN:
            //检测是否按下“开始游戏”按钮
            if (mouseDetect(goOn, m)) {
                cleardevice();
                play = true;
                menu_progress = time(NULL);
                menu_sum += (menu_progress - menu_begin);
                break;
            }
            /*
            //检测是否按下“设置”按钮
            if (mouseDetect(site, m)) {
            }
            */
            //检测是否按下“退出游戏”按钮
            if (mouseDetect(quit, m)) {
                exit(-1);
            }
        }
    }
}

//失败界面
void loseInterface() {
    EndBatchDraw();
    Button* returnMain, * quit;
    MessageBox(NULL,
        "您的赛车血量已耗尽，感谢您的游玩，欢迎您再次挑战",
        "失败", MB_SETFOREGROUND);
    mciSendString("close all", NULL, 0, NULL);
    mciSendString("open .\\sound\\gameover.mp3 alias gameovermusic", NULL, 0, NULL);
```

```
    mciSendString("play gameovermusic from 0", NULL, 0, NULL);
    //创建按钮对象
    returnMain = createButton(0, 800, 200, 100, BLACK, "主界面");
    quit = createButton(1080, 800, 200, 100, BLACK, "退出游戏");
    cleardevice();
    //绘制按钮
    putimage(0, 0, &gameOver);
    drawButton(returnMain, false);
    drawButton(quit, false);
    //检测鼠标对于暂停页面按钮的点击事件
    while (true) {
        m = getmessage(EX_MOUSE);                                    //
获取鼠标信息
        switch (m.message)
        {
        case WM_LBUTTONDOWN:
            //检测是否按下“返回主界面”按钮
            if (mouseDetect(returnMain, m)) {
                cleardevice();
                drawFrontCover();
            }
            //检测是否按下“退出游戏”按钮
            if (mouseDetect(quit, m)) {
                exit(-1);
            }
        }
    }
}

//模式一
void mode_1() {
    Car* car;
    Map* map;
    Obstacle** obstacle;
    Rocket** rocket;
    int selection;
    int life_leave;                                                  // 生 命
值剩余数量
    int rocket_leave;                                                // 生 命
值剩余数量
    IMAGE img_1, img_2;
    //创建障碍物指针数组
    obstacle = (Obstacle**)malloc(sizeof(Obstacle) * 3);
    //创建火箭指针数组
```

```
    rocket = (Rocket**)malloc(sizeof(Rocket) * 3);
    map = createMap(0, -(960 * 9), 1);                                    //创建地图对象
    car = createCar(640, 750, 2, true, 0);                                //创建汽车对象
    selection = drawCarSelection();
    //暂停播放背景音乐
    //PlaySound(NULL, NULL, SND_FILENAME | SND_PURGE);
    mciSendString("stop bkmusic", NULL, 0, NULL);
    //音乐初始化打开
    mciSendString("open .\\sound\\game.mp3 alias gamemusic", NULL, 0, NULL);
    mciSendString("open .\\sound\\fire.mp3 alias firemusic", NULL, 0, NULL);
    mciSendString("open .\\sound\\crash.mp3 alias crashmusic", NULL, 0, NULL);
    mciSendString("open .\\sound\\explode.mp3 alias explodemusic", NULL, 0, NULL);
    //循环播放游戏音乐
    mciSendString("play gamemusic repeat from 0", NULL, 0, NULL);
    //基于玩家选车选择，进行图片初始化
    switch (selection) {
        //选择奥迪车
    case 1:
        img_1 = car_Audi[0];
        img_2 = car_Audi[1];
        break;

        //选择宝马车
    case 2:
        img_1 = car_Bmw[0];
        img_2 = car_Bmw[1];
        break;

        //选择奔驰车
    case 3:
        img_1 = car_Mercedes[0];
        img_2 = car_Mercedes[1];
        break;
    }

    //初始化血量信息
    life_leave = 3;

    //初始化血量信息
    rocket_leave = 3;

    //初始化障碍物信息
```

```
    for (int i = 0; i < 3; i++) {
        obstacle[i] = createObstacle(map_obstacle[rand() % 3], -(rand() % (100)), (rand() % 1) +
1, true, 0);
    }

    //初始化火箭信息
    for (int i = 0; i < 3; i++) {
        rocket[i] = createRocket(0, 0, 1, true);
    }
    cleardevice();
    game_begin = time(NULL);                                                    // 游 戏
开始时间

    //地图贴图
    putimage(map->x, map->y, &map_2);

    //玩家控制车辆初始位置显示
    putimage(car->x, car->y, &img_1, NOTSRCERASE);
    putimage(car->x, car->y, &img_2, SRCINVERT);
    BeginBatchDraw();

    //模式一主循环
    while (true) {
        game_progress = time(NULL);                                                 //
游戏进程时间
        game_sum = game_progress - game_begin;                                      // 游 戏
总时间

        //如果图片加载到尽头，则重置贴图
        if (map->y == 0) {
            map->x = 0;
            map->y = -(960 * 9);
        }

        //游戏进行时间 3s 后再布置障碍物，避免出生即触碰障碍物
        if (game_sum - menu_sum >= 3) {
            for (int i = 0; i < 3; i++) {
                obstacle[i]->y += obstacle[i]->speed;
            }
        }

        //地图移动
        map->y += map->speed;
```

```
        //车向上移动
        //当按下 ↑ 与 W 时
        if (GetAsyncKeyState(VK_UP) || GetAsyncKeyState('W')) {
            if (0 <= car->y) {
                car->y -= car->speed;

                //控制速度改变间隔
                if (!isUp) {
                    isUp = true;
                    control_up_last = game_sum - menu_sum;
                }

                //速度间隔 1s 及以上
                if (speed_show <= speed_max && game_sum - menu_sum -
control_up_last >= 1) {
                    isUp = false;
                    speed_show += 1;
                }
            }
        }

        //车向下移动
        //当按下 ↓ 与 S 时
        if (GetAsyncKeyState(VK_DOWN) || GetAsyncKeyState('S')) {
            if (car->y <= 760) {
                car->y += car->speed;

                //控制速度改变间隔
                if (!isDown) {
                    isDown = true;
                    control_down_last = game_sum - menu_sum;
                }

                //速度间隔 3s 及以上
                if (speed_show >= speed_min && game_sum - menu_sum -
control_down_last >= 3) {
                    isDown = false;
                    speed_show -= 1;
                }
            }
        }

        //车向左移动
        //当按下 ← 与 A 时
```

```
if (GetAsyncKeyState(VK_LEFT) || GetAsyncKeyState('A')) {
	if (150 <= car->x) {
		car->x -= car->speed;
	}
}

//车向右移动
//当按下 → 与 D 时
if (GetAsyncKeyState(VK_RIGHT) || GetAsyncKeyState('D')) {
	if (car->x <= 930) {
		car->x += car->speed;
	}
}

//发射火箭
//第一次发射火箭
if (!onFire) {
	fire_last = time(NULL);
	onFire = true;

	//当按下空格键时
	if (GetAsyncKeyState(VK_SPACE)) {
		mciSendString("play firemusic from 0", NULL, 0, NULL);
		for (int i = 0; i < 3; i++) {
			//如果障碍物状态为真
			if (rocket[i]->status == true) {
				rocket[i]->x = car->x + (SIZE / 2);
				rocket[i]->y = car->y;
				rocket[i]->status = false;
				rocket_leave -= 1;
				break;
			}
		}
	}
}

//非第一次发射火箭
else {
	//计算两次发射间隔
	fire_now = time(NULL);

	//按下空格键且发射间隔 1s 及以上，才可以发射火箭
	if (GetAsyncKeyState(VK_SPACE) && (fire_now - fire_last) >= 2) {
		mciSendString("play firemusic from 0", NULL, 0, NULL);
```

```
                for (int i = 0; i < 3; i++) {
                    if (rocket[i]->status == true) {
                        rocket[i]->x = car->x + (SIZE / 2);
                        rocket[i]->y = car->y;
                        rocket[i]->status = false;
                        rocket_leave -= 1;
                        break;
                    }
                }
                fire_last = fire_now;
            }
        }

        //如障碍物坐标超出范围，则初始化其位置
        for (int i = 0; i < 3; i++) {
            if (obstacle[i]->y >= 960) {
                obstacle[i]->status = true;
                obstacle[i]->y = -(rand() % 960);
                obstacle[i]->x = map_obstacle[rand() % 3];
                obstacle[i]->speed = (rand() % 1) + 1;
            }
        }

        //改变火箭位置，并判断其是否超出范围，如若超出范围，则更新其状态为可发射
        for (int i = 0; i < 3; i++) {
            if (rocket[i]->status == false) {
                if (rocket[i]->y <= 0) {
                    rocket[i]->status = true;
                    rocket_leave += 1;
                    continue;
                }
                rocket[i]->y -= rocket[i]->speed;
            }
        }

        //车辆与障碍物的碰撞检测
        for (int i = 0; i < 3; i++) {
            //重置碰撞状态
            if (isHit == true && game_sum - menu_sum >= 0 && game_sum - menu_sum -
car->last > 1) {
                isHit = false;
            }
            //当两图片有交集时 if 判断成立
```

```
            if (obstacle[i]->status == true
                && abs((obstacle[i]->x + SIZE_OBSTACLE) / 2 - (car->x + SIZE) / 2) <=
abs(SIZE_OBSTACLE - SIZE)
                && abs((obstacle[i]->y + SIZE_OBSTACLE) / 2 - (car->y + SIZE) / 2) <=
abs(SIZE_OBSTACLE - SIZE)
                ) {
                //如果生命值归零
                if (life_leave == 0) {
                    Sleep(1000);
                    loseInterface();
                }

                //与上次碰撞间隔 1s 以上才会扣除生命值，避免持续碰撞，血量立即归
零
                if (!isHit) {
                    mciSendString("play crashmusic from 0", NULL, 0, NULL);
                    life_leave -= 1;
                    car->last = game_sum - menu_sum;
                    isHit = true;
                }
            }
        }

        //火箭与障碍物的碰撞检测
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                //当两图片有交集时 if 判断成立
                if (rocket[j]->status == false
                    && rocket[j]->x >= obstacle[i]->x
                    && rocket[j]->x <= (obstacle[i]->x + SIZE_OBSTACLE)
                    && rocket[j]->y <= (obstacle[i]->y + (SIZE_OBSTACLE / 2))
                    && rocket[j]->y >= obstacle[i]->y
                    || (rocket[j]->status == false
                        && (rocket[j]->x + SIZE_ROCKET) <= (obstacle[i]->x + SIZE_OBSTACLE)
                        && (rocket[j]->x + SIZE_ROCKET) >= obstacle[i]->x
                        && (rocket[j]->y + SIZE_ROCKET) <= (obstacle[i]->y + (SIZE_OBSTACLE
/ 2))
                        && (rocket[j]->y + SIZE_ROCKET) >= obstacle[i]->y)
                    ) {
                    mciSendString("play explodemusic from 0", NULL, 0, NULL);
                    rocket[j]->status = true;
                    rocket_leave += 1;
                    rocket[j]->y = 2000;
                    obstacle[i]->status = false;
```

```
                    obstacle[i]->last = game_sum - menu_sum;
                    score += 100;
                }
            }
        }
        //清理贴图
        cleardevice();

        //地图贴图
        putimage(map->x, map->y, &map_2);

        //避免障碍物图片贴图重叠
        if (game_sum - menu_sum >= 3) {
            for (int i = 0; i < 3; i++) {
                for (int j = i + 1; j < 3; j++) {
                    //当两图片有交集时 if 判断成立
                    if (obstacle[j]->x <= obstacle[i]->x
                        && obstacle[j]->x <= (obstacle[i]->x + SIZE_OBSTACLE)
                        && obstacle[j]->y <= (obstacle[i]->y + SIZE_OBSTACLE)
                        && obstacle[j]->y >= obstacle[i]->y
                        || ((obstacle[j]->x + SIZE_OBSTACLE) <= (obstacle[i]->x + SIZE_OBSTACLE)
                            && (obstacle[j]->x + SIZE_OBSTACLE) >= obstacle[i]->x
                            && (obstacle[j]->y + SIZE_OBSTACLE) <= (obstacle[i]->y + SIZE_OBSTACLE)
                            && (obstacle[j]->y + SIZE_OBSTACLE) >= obstacle[i]->y)
                        ) {
                        obstacle[j]->y = -(rand() % 960);
                    }
                }
            }
        }

        //放置障碍物图片
        if (game_sum - menu_sum >= 3) {
            for (int i = 0; i < 3; i++) {
                //如果障碍物状态为真，贴图石头
                if (obstacle[i]->status == true) {
                    putimage(obstacle[i]->x, obstacle[i]->y, &rock[0], NOTSRCERASE);
                    putimage(obstacle[i]->x, obstacle[i]->y, &rock[1], SRCINVERT);
                }
                else {
                    //爆炸画面持续 1s
                    if (game_sum - menu_sum >= 3 && game_sum - menu_sum -
```

```
obstacle[i]->last >= 1) {
                                obstacle[i]->status = true;
                                obstacle[i]->x = map_obstacle[rand() % 3];
                                obstacle[i]->speed = (rand() % 1) + 1;
                                obstacle[i]->y = -(rand() % 960);
                            }
                            else {
                                putimage(obstacle[i]->x, obstacle[i]->y, &bomb[0], NOTSRCERASE);
                                putimage(obstacle[i]->x, obstacle[i]->y, &bomb[1], SRCINVERT);
                            }
                        }
                    }
                }

                //放置火箭图片
                for (int i = 0; i < 3; i++) {
                    //如果火箭状态为假,贴图火箭
                    if (rocket[i]->status == false) {
                        putimage(rocket[i]->x, rocket[i]->y, &rocket_img[0], NOTSRCERASE);
                        putimage(rocket[i]->x, rocket[i]->y, &rocket_img[1], SRCINVERT);
                    }
                }

                //展示信息表
                putimage(0, 0, &show);

                //时间显示
                putimage(110, 37, &number[9 - ((game_sum - menu_sum) / 10)]);    //时间贴图十位
                putimage(135, 37, &number[9 - ((game_sum - menu_sum) % 10)]);   //时间贴图个位

                //车辆速度显示
                putimage(110, 15, &number[speed_show / 10]);                    //速度贴图十位
                putimage(135, 15, &number[speed_show % 10]);                    //速度贴图个位

                //路程显示
                distance = speed_show * (game_sum - menu_sum);
                PutNumber(110, 60, distance, 10, 2, 0, WHITE, "宋体");

                //赛车图片显示
                putimage(car->x, car->y, &img_1, NOTSRCERASE);
                putimage(car->x, car->y, &img_2, SRCINVERT);

                //生命值信息显示
                settextstyle(16, 0, ("Consolas"));
```

```
            settextcolor(BLACK);
            outtextxy(0, 160, "生命值：");
            rectangle(0, 200, (3 * life.getwidth() + 20), 200 + life.getheight() + 10);
            for (int i = 0; i < life_leave; i++) {
                putimage((i * life.getwidth() + 10), 205, &life);
            }

            //弹药数信息显示
            settextstyle(16, 0, ("Consolas"));
            settextcolor(BLACK);
            outtextxy(0, 340, "弹药数：");
            rectangle(0, 380, (3 * life.getwidth() + 20), 380 + life.getheight() + 10);
            for (int i = 0; i < rocket_leave; i++) {
                putimage((i * life.getwidth() + 10), 385, &life);
            }

            //保护时间信息显示
            if (isHit == true && game_sum - menu_sum >= 0 && game_sum - menu_sum - car->last
< 1) {
                outtextxy(100, 160, "受保护状态");
            }

            //分数信息显示
            outtextxy(0, 250, "获得分数：");
            PutNumber(0, 280, score);
            FlushBatchDraw();

            //按下 ESC，呈现暂停画面
            if (GetAsyncKeyState(VK_ESCAPE)) {
                getimage(&temp, 0, 0, WIDTH, HEIGHT);
                mciSendString("pause gamemusic", NULL, 0, NULL);
                //暂停界面显示
                pauseInterface();
                mciSendString("resume gamemusic", NULL, 0, NULL);
            }

            //时间结束，通过终点画面呈现
            if (game_sum - menu_sum >= 98) {
                break;
            }
        }
        EndBatchDraw();
        BeginBatchDraw();
```

```
//通过终点线画面加载
while (true) {
	cleardevice();
	//如果图片加载到尽头，则重置贴图
	if (map->y == 0) {
		map->x = 0;
		map->y = -(960 * 9);
	}

	//地图移动
	map->y += map->speed;

	//将汽车置于地图中央
	if (car->x != WIDTH / 2 - 100 && car->y != HEIGHT / 2) {
		if (car->x <= WIDTH / 2 - 100) {
			car->x += car->speed;
		}
		if (car->x >= WIDTH / 2 - 100) {
			car->x -= car->speed;
		}
		if (car->y <= HEIGHT / 2) {
			car->y += car->speed;
		}
		if (car->y >= HEIGHT / 2) {
			car->y -= car->speed;
		}
	}
	//如果已置于地图中央，则加载终点
	else {
		end_y += map->speed;
	}
	//地图图片显示
	putimage(map->x, map->y, &map_2);

	//展示信息表
	putimage(0, 0, &show);

	//时间显示
	putimage(110, 37, &number[9 - ((game_sum - menu_sum) / 10)]);	//时间贴图十位
	putimage(135, 37, &number[9 - ((game_sum - menu_sum) % 10)]);	//时间贴图个位

	//车辆速度显示
	putimage(110, 15, &number[speed_show / 10]);				//速度贴图十位
	putimage(135, 15, &number[speed_show % 10]);				//速度贴图个位
```

```
            //路程显示
            distance = speed_show * (game_sum - menu_sum);
            PutNumber(110, 60, distance, 10, 2, 0, WHITE, "宋体");

            //终点图片显示
            putimage(150, end_y, &terminalPoint);

            //赛车图片显示
            putimage(car->x, car->y, &img_1, NOTSRCERASE);
            putimage(car->x, car->y, &img_2, SRCINVERT);

            //生命值信息显示
            settextstyle(16, 0, ("Consolas"));
            settextcolor(BLACK);
            outtextxy(0, 160, "生命值：");
            rectangle(0, 200, (3 * life.getwidth() + 20), 200 + life.getheight() + 10);
            for (int i = 0; i < life_leave; i++) {
                putimage((i * life.getwidth() + 10), 205, &life);
            }

            //保护时间信息显示
            if (game_sum - menu_sum >= 0 && game_sum - menu_sum - car->last <= 1) {
                outtextxy(100, 160, "受保护状态");
            }

            //分数信息显示
            outtextxy(0, 250, "获得分数：");
            PutNumber(0, 280, score);

            //如果汽车通过终点线，出现结算界面
            if (car->y + SIZE == end_y) {
                break;
            }

            FlushBatchDraw();
        }
        EndBatchDraw();
        winInterface();       //胜利画面
}

void PutImgWithout(IMAGE& obj, int px, int py, COLORREF withouter = WHITE, DWORD* pbWnd
= GetImageBuffer(), int wX = getwidth(), int wY = getheight(), DWORD bitsub = 0x00FFFFFF)
{
```

```
	DWORD* pbImg = GetImageBuffer(&obj);
	int iX = obj.getwidth();
	int iY = obj.getheight();
	for (int i1 = 0; i1 < iX; i1++)
	{
		for (int i2 = 0; i2 < iY; i2++)
		{
			// 检测是否越界
			if (pointTsm(i1 + px, i2 + py, wX, wY) == -1)continue;

			// 检测是否要排除该颜色
			if ((pbImg[pointTsm(i1, i2, iX, iY)] & bitsub) == BGR(withouter))continue;

			// 操作显存
			pbWnd[pointTsm(i1 + px, i2 + py, wX, wY)] = pbImg[pointTsm(i1, i2, iX, iY)];
		}
	}
}

void onUp(Car* car) {
	car->y -= car->speed * (int)round(sin(Forward + PI / 2));
	car->x -= car->speed * (int)round(cos(Forward + PI / 2));
}

void onDown(Car* car) {
	car->y += car->speed * (int)round(sin(Forward + PI / 2));
	car->x += car->speed * (int)round(cos(Forward + PI / 2));
}

void onLeft(Car* car) {
	if (canCircle(car)) {
		Forward -= PI / Rota;
		//car->x += (int)round(cos(Forward - PI / 2));
		//car->y += (int)round(sin(Forward - PI / 2));
	}
}

void onRight(Car* car) {
	if (canCircle(car)) {
		Forward += PI / Rota;
		//car->x += (int)round(cos(Forward + PI / 2));
		//car->y += (int)round(sin(Forward + PI / 2));
	}
}
```

```
//判断是否可以旋转
bool canCircle(Car* car) {
	//指向缓冲区的指针变量
	DWORD* map;                                    //创建 DWORD 类型指针(typedef unsigned int DWORD;)
	DWORD temp;
	int temp_x, temp_y;                            //用于暂时存储汽车的 x ， y 值
	int iX, iY;                                    //用于暂时存储是否可以旋转时的区域判断 x， y 值
	map = GetImageBuffer(&map_1);
	temp_x = car->x;
	temp_y = car->y;
	rotateimage(&car_circle[0], &car_black[0], -Forward, WHITE, true, false);
	rotateimage(&car_circle[1], &car_black[1], -Forward, WHITE, true, false);
	iX = car_circle[0].getwidth();
	iY = car_circle[0].getheight();
	for (int i = 0; i < iX; i++) {
		for (int j = 0; j < iY; j++) {
			temp = map[pointTsm((int)ceil(i + car->x), (int)ceil(j + car->y), WIDTH, HEIGHT)] & 0x00FFFFF;
			if (temp == BGR(RGB(215, 113, 53))
				|| temp == BGR(RGB(250, 250, 250))
				|| temp == BGR(RGB(79, 146, 202))) {
				return false;
			}
		}
	}

	return true;
}

//边缘检测
void edgeDetection(Car* car) {
	DWORD temp;
	//指向缓冲区的指针变量
	DWORD* map;                                    //创建 DWORD 类型指针(typedef unsigned int DWORD;)
	map = GetImageBuffer(&map_1);                  //获取地图的显示缓冲区指针
	int iX = car_black[0].getwidth();              //汽车图片宽度
	int iY = car_black[0].getheight();             //汽车图片高度

	//依据像素点颜色判断汽车是否接触边缘
	for (int i = 0; i < iX; i++)
	{
```

```
            for (int j = 0; j < iY; j++)
            {
                temp = map[pointTsm((int)ceil(i + car->x), (int)ceil(j + car->y), WIDTH, HEIGHT)] &
0x00FFFFFF;
                if (temp == BGR(RGB(215, 113, 53))
                    || temp == BGR(RGB(250, 250, 250))
                    || temp == BGR(RGB(79, 146, 202))) {
                    car->speed = 1;
                }
                else {
                    car->speed = 1;
                }
            }
        }
}

//越界检测
int pointTsm(int x, int y, int wide, int high)
{
    if (x < 0) return x = 0;
    if (x >= wide) return x = wide;
    if (y < 0) return y = 0;
    if (y >= high) return y = high;

    return wide * y + x;
}

//控制汽车
void controlCar() {
    Car* car;
    IMAGE img_1, img_2;
    img_1 = car_black[0];
    img_2 = car_black[1];
    car = createCar(1000, 750, 1, true, 0);
    putimage(car->x, car->y, &img_1, NOTSRCERASE);
    putimage(car->x, car->y, &img_2, SRCINVERT);
    BeginBatchDraw();                                    //内存缓冲，避免画面撕裂
    while (true) {
        //车向上移动
        if (GetAsyncKeyState(VK_UP) || GetAsyncKeyState('W')) {
            onUp(car);
        }

        //车向下移动
```

```
        if (GetAsyncKeyState(VK_DOWN) || GetAsyncKeyState('S')) {
            onDown(car);
        }

        //车向左移动
        if (GetAsyncKeyState(VK_LEFT) || GetAsyncKeyState('A')) {
            onLeft(car);
            rotateimage(&img_1, &car_black[0], -Forward);
            rotateimage(&img_2, &car_black[1], -Forward, WHITE);
        }

        //车向右移动
        if (GetAsyncKeyState(VK_RIGHT) || GetAsyncKeyState('D')) {
            onRight(car);
            rotateimage(&img_1, &car_black[0], -Forward);
            rotateimage(&img_2, &car_black[1], -Forward, WHITE);
        }
        cleardevice();
        putimage(0, 0, &map_1);
        edgeDetection(car);
        //putimage(car->x, car->y, &car_black[0], NOTSRCERASE);
        //putimage(car->x, car->y, &car_black[1], SRCINVERT);
        putimage(car->x, car->y, &img_1, NOTSRCERASE);
        putimage(car->x, car->y, &img_2, SRCINVERT);
        FlushBatchDraw();                                   //将缓冲画面释放到显示画面
    }

    EndBatchDraw();                                         //结束缓冲
}
//首次尝试以图片指针的形式，遍历像素颜色，进行图片信息存取，但面临栈溢出的问题
/*
void getMapInformation(IMAGE map_1);                        //存储地图信息
//存储地图信息
void getMapInformation(IMAGE map_1) {
    int temp[WIDTH * HEIGHT];
    int count = 0;
    map = GetImageBuffer(&map_1);
    for (int i = 0; i < WIDTH*HEIGHT; i++) {
        if (map[i] == BGR(RGB(250, 250, 250)) || map[i] == BGR(RGB(215, 113, 53)) || map[i] ==
BGR(RGB(79, 146, 202))) {
            temp[i] = 1;
            printf("1");
        }
        else {
```

```
                temp[i] = 0;
                printf("0");
            }
        }
        for (int i = 0; i < WIDTH; i++) {
            for (int j = 0; j < HEIGHT; j++) {
                map_information[i][j] = temp[count++];
            }
        }
        //
        for (int i = 0; i < WIDTH; i++) {
            for (int j = 0; j < HEIGHT; j++) {
                if (getpixel(i, j) == RGB(255, 255, 255) || getpixel(i, j) == RGB(215, 113, 53) ||
getpixel(i, j) == RGB(79, 146, 202)) {
                    map_information[i][j] = 1;
                }
            }
        }
        for (int i = 0; i < WIDTH; i++) {
            for (int j = 0; j < HEIGHT; j++) {
                printf("%d",map_information[i][j]);
            }
            printf("\n");
        }
}
*/
```