****

**《程序设计课程实践》设计文档**

# airwar程序设计报告

**团队成员：**

**18051624 沈建鑫**

**18051623 秦嘉珩**

**18051622 倪梓皓**

完成时间 2018 年 12 月

# airwar程序设计报告

## 人员组成及分工

### 人员组成及分工

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **姓名** | **任务** | **完成工作量** |
| **18051624** | **沈建鑫** | **设计程序文件操作相关部分及文案负责** | **33.3%** |
| **18051623** | **秦嘉珩** | **主体框架、界面绘制及背景音乐** | **33.3%** |
| **18051622** | **倪梓皓** | **界面、素材制作及碰撞检测判断** | **33.3%** |

## 2．开发背景

雷电（Thunder）是一款电视游戏机和掌上游戏机游戏，《雷电》是一款由西武开发的射击游戏；也是全球弹幕游戏的鼻祖之一，游戏《雷电1》在1990年在日本上市，受到许多街机迷的热爱，随后在1993年推出雷电1的续作《雷电2》把弹幕游戏推向世界，在2005年推出的革新大作《雷电3》取代了传统的2D画面变更为3D画面，并在各种游戏平台，街机，电脑上出现，全新设计和玩法使玩家络绎不绝；2007年推出的《雷电4》支持PS2，XBOX，欧美玩家也能玩到新游，并且支持联网，2015年推出《雷电4 OverKill》更是把游戏推向了高潮，并出现新角色和新模式。本此课程设计是基于雷电的模型，自主研发并修改了一些细节而进行的一款名为airwar的游戏。

在本次课程设计中，要求支持键盘操作，并且可以记录玩家的即时得分和选择是否保存最终得分。随着游戏的进行，分数越高，游戏难度越大，即敌人的数量会越来越多。

## 系统功能设计

### 系统功能模块图



图1 系统功能模块图

2.2 系统业务流程图

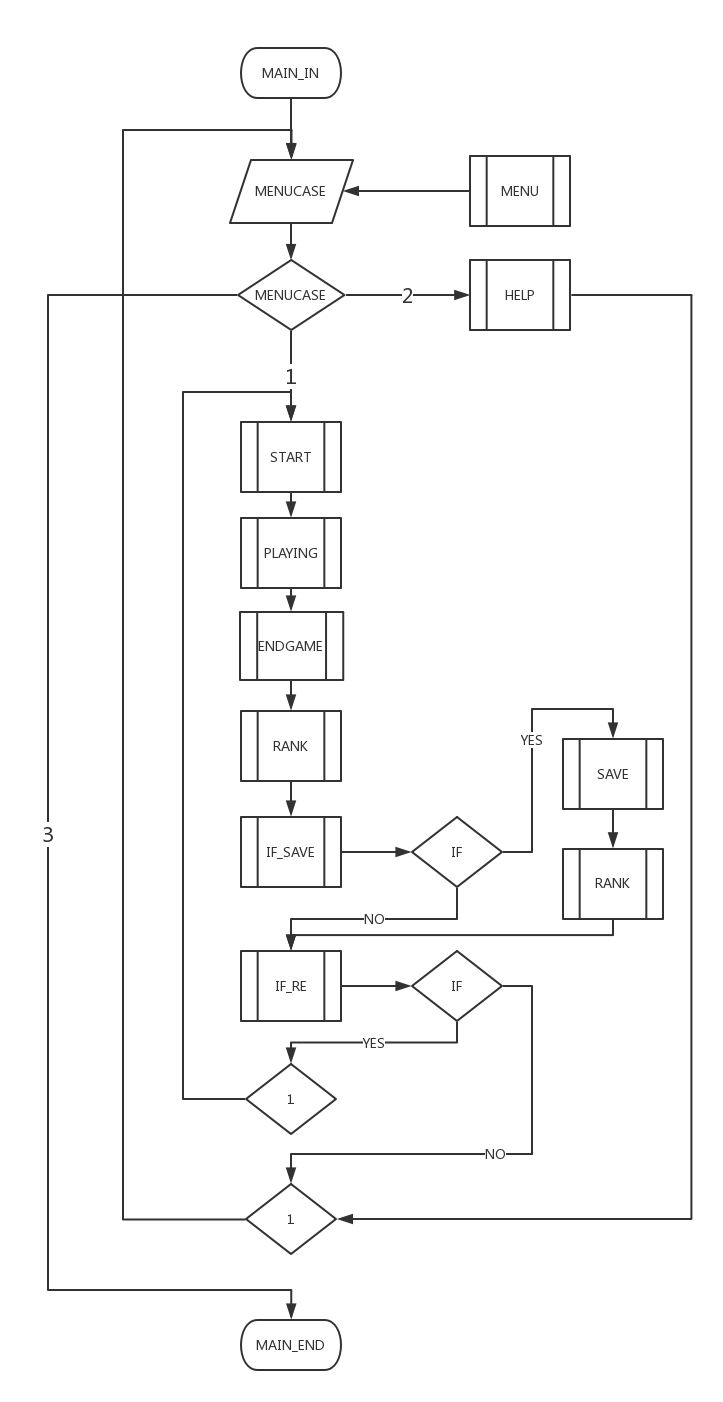


图2 业务流程图

## 项目创建

### 系统开发环境要求

本项目的开发及运行环境要求：

操作系统：windows10

开发工具：visual stdio 2017

开发语言：c++语言

### 项目创建过程

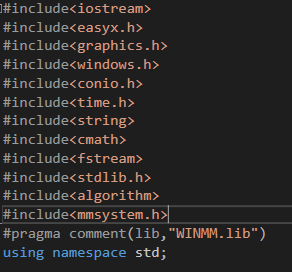
本次项目创建主要分了以下六个过程

1. 项目的创建，选择新建项目，设置保存路径
2. 主体框架的构建，将主函数、函数定义、全局变量定义等确定
3. 细化框架的构建，具体相关函数的制作，放入相关图片
4. 碰撞检测以及游戏功能完善
5. 文件输入输出及其他功能完善，设置txt文档
6. 最终定稿

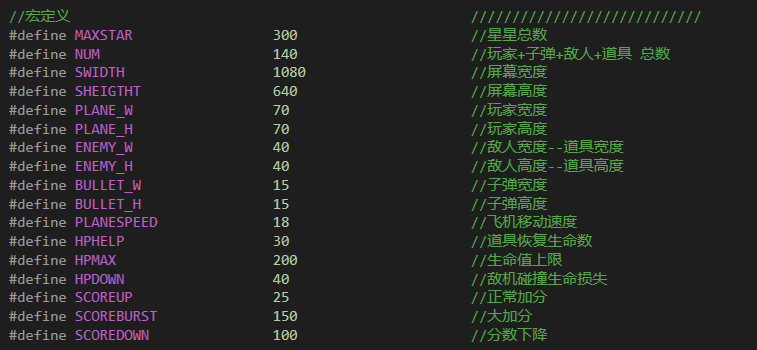
（具体图片见文档外文件夹《项目创建过程》）

## 预处理模块设计

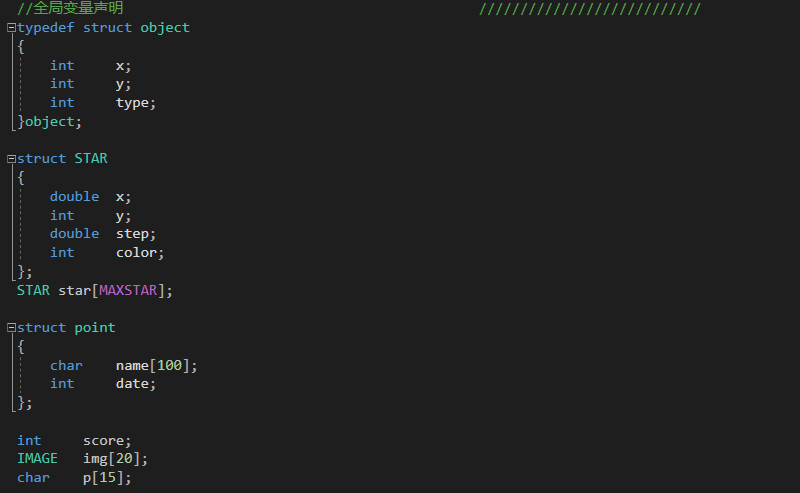
### 文件引用



### 宏定义



### 定义全局变量



### 函数声明

1. void menu( )

输入参数：无

输出参数：无

实现功能：绘制菜单

1. bool box( )

输入参数：无

输出参数：yes返回1，no返回0

实现功能：弹出对话框

1. void ifsave( )

输入参数：无

输出参数：无

实现功能：绘制保存得分/是否保存界面

1. bool ifrestart( )

输入参数：无

输出参数：yes返回1，no返回0

实现功能：绘制是否重新开始界面

1. void readrank( )

输入参数：无

输出参数：无

实现功能：绘制得分查询界面

1. void load( )

输入参数：无

输出参数：无

实现功能：加载图片资源

1. void InitStar(int)

输入参数：一个int类型的i，代表第i个星星

输出参数：无

实现功能：背景星空绘制

1. void MoveStar(int)

输入参数：一个int类型的i，代表第i个星星

输出参数：无

实现功能：移动星星

1. char \*numtostr(int)

输入参数：一个int类型的score/players[0].type代表分数/生命值

输出参数：无

实现功能：数字转字符串，用于打印得分/生命值

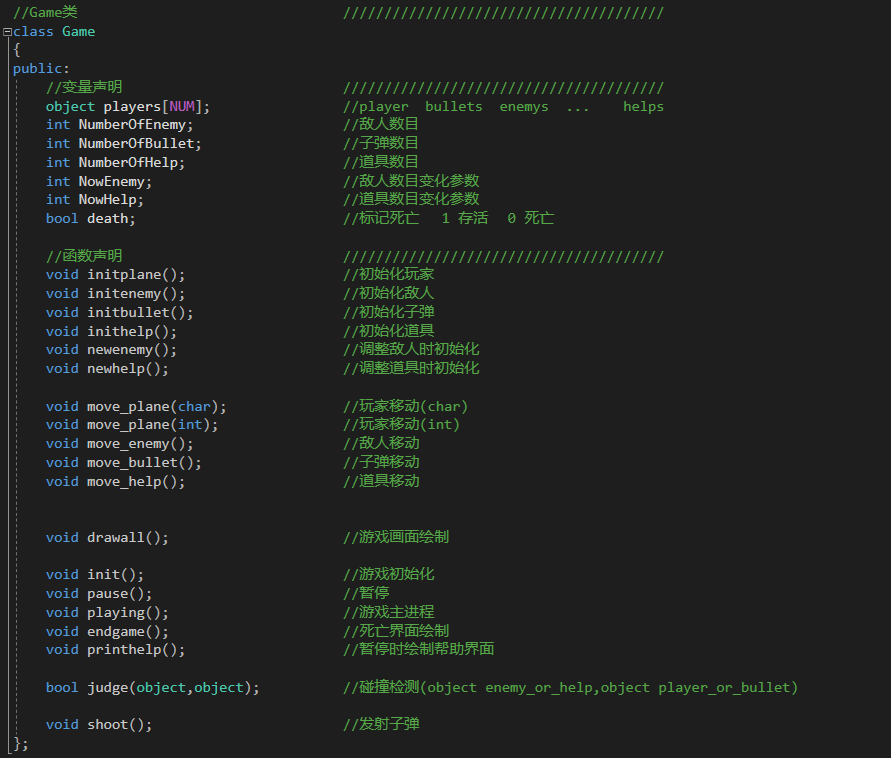
1. int cmp(struct point,struct point)

输入参数：两个point类型的结构体

输出参数：大于返回1，小于返回0

实现功能：用于排行榜的排序

### Game类声明



## 游戏欢迎界面设计

### 游戏欢迎界面设计概述



首先加载一张背景图，然后按下回车键进入欢迎界面



此时，右上角矩形框内显示为游戏提示

右下角矩形框内显示为菜单选项

菜单选项分为：开始游戏、查看排名READ、退出游戏三部分

### 设置文字颜色



用settextstyle函数设置文字字体和大小，颜色默认

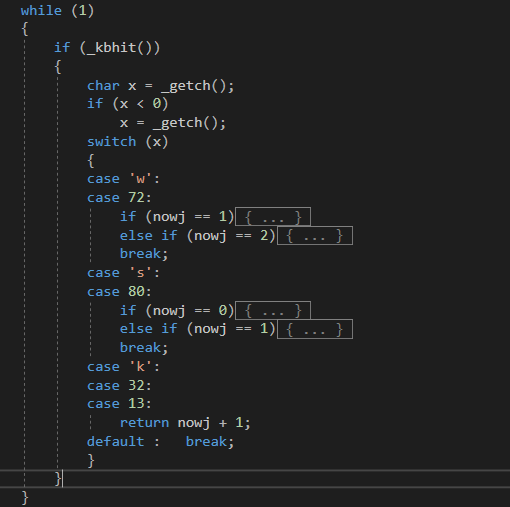
### 设置文字显示位置





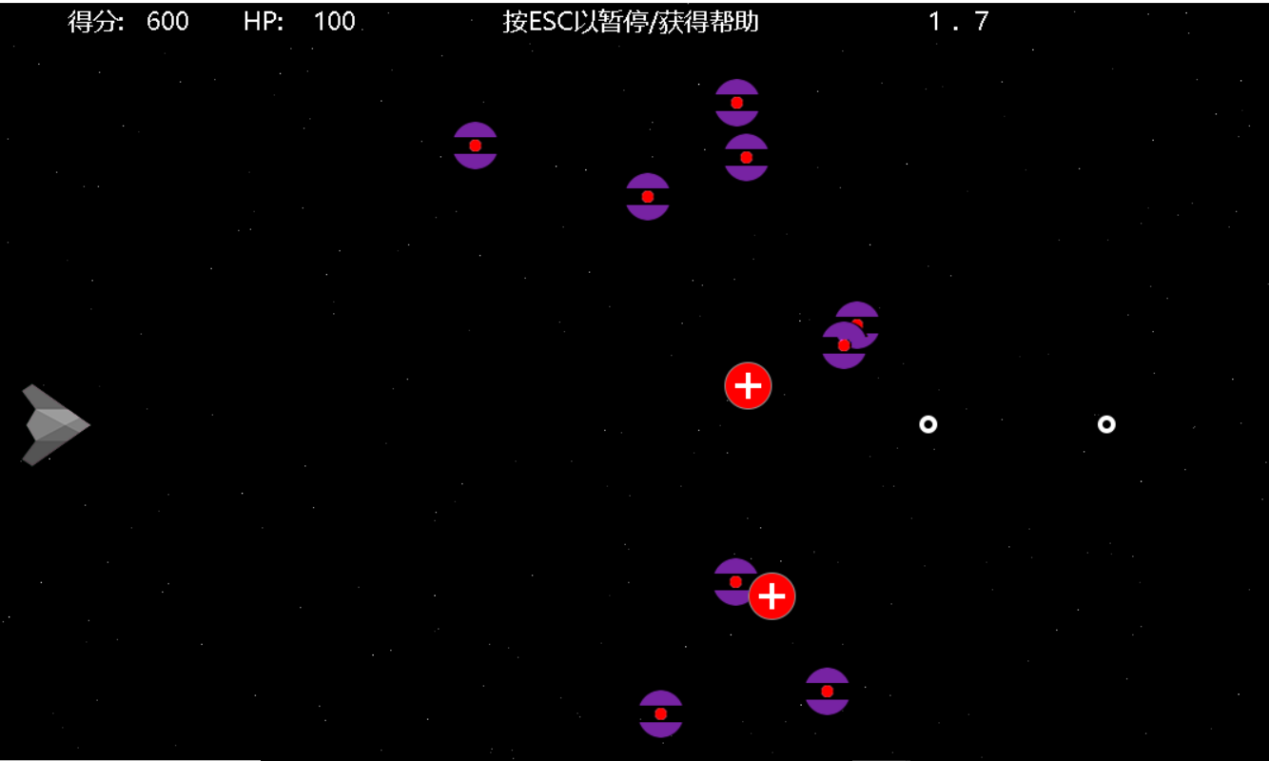
先绘制矩形框选定位置，再通过drawtext函数在绘制的矩形框内各自显示文字

### 菜单选项的移动



利用nowj记录当前选项框所在位置右下角纵坐标，读取键盘输入的值通过nowj的变化来控制选项的移动

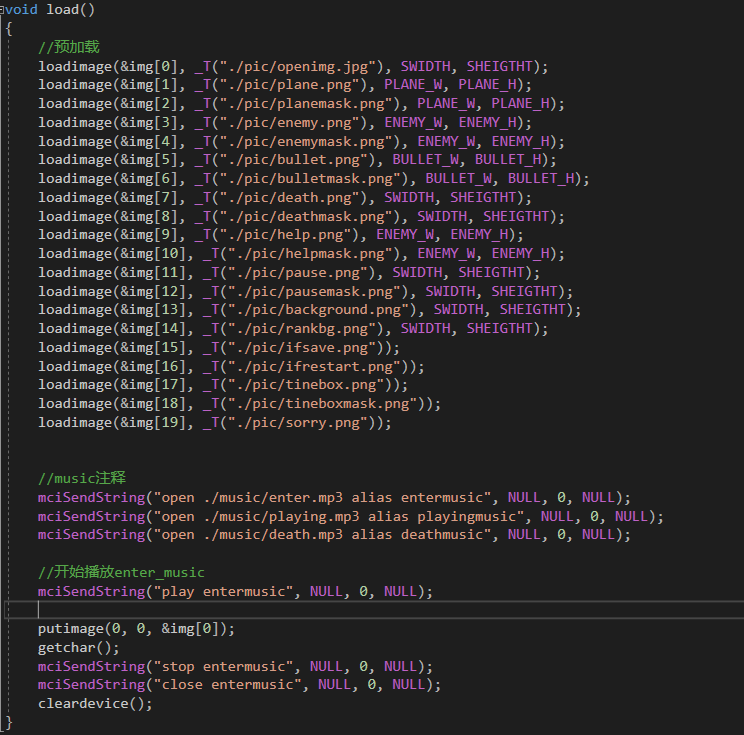
## 游戏主窗体设计



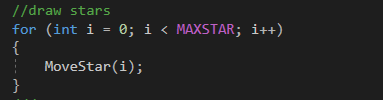
我们绘制了飞机，敌人，医疗包和星空背景，在左上角会显示当前得分和HP值



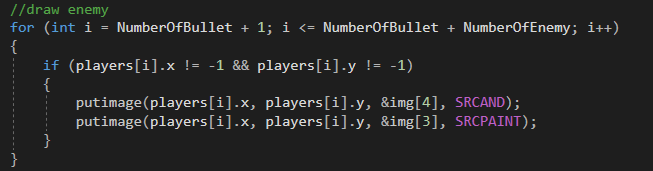
游戏过程中按下esc键可以使游戏暂停，同时弹出游戏帮助界面



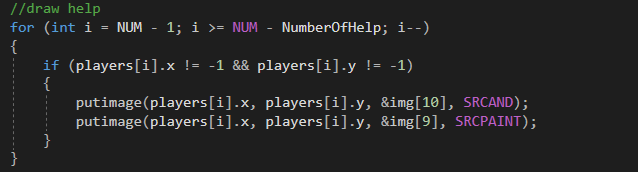
这是载入必要音乐和图片素材的代码



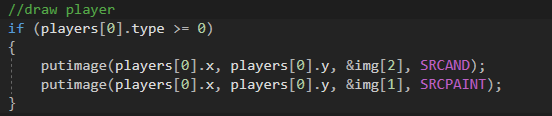
绘制星星的代码



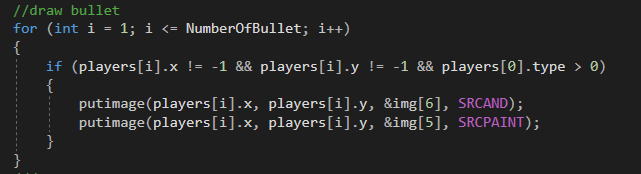
绘制敌人的代码



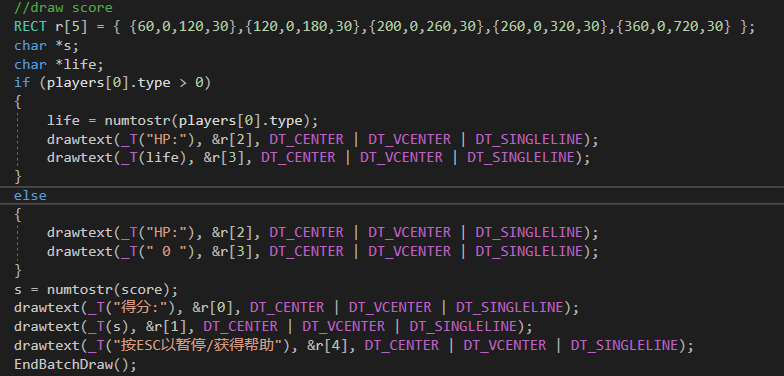
绘制道具的代码



绘制飞机的代码



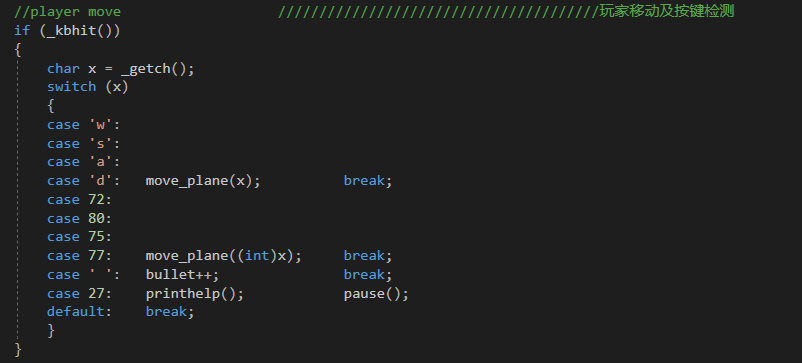
绘制子弹的代码



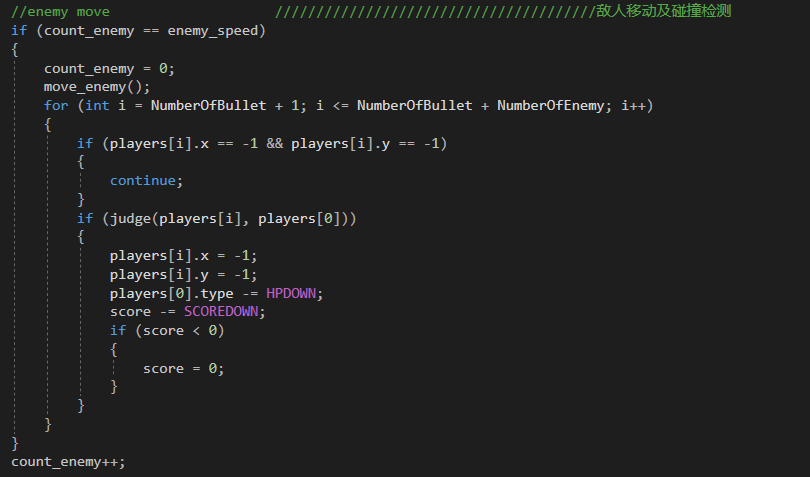
绘制分数的代码

## 游戏逻辑设计

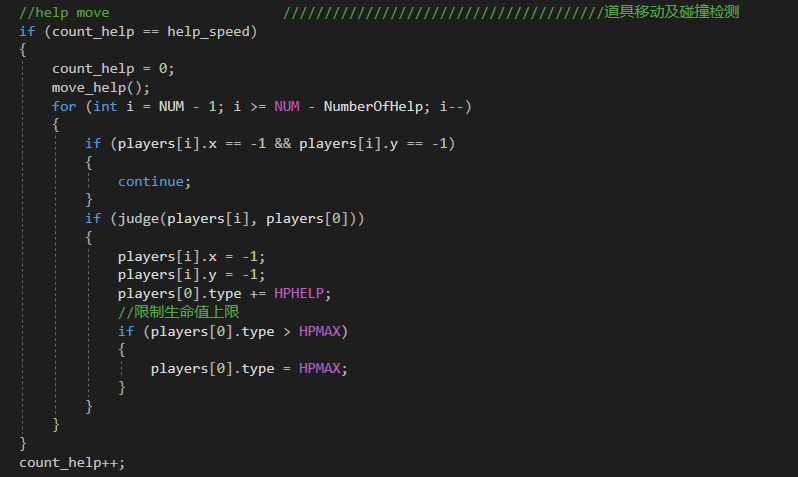
本游戏通过键盘控制飞机的移动，发射子弹消灭敌人获取分数，与敌人相撞时减少HP，HP降为0时游戏结束。并且加入了医疗包道具，增加了游戏的可玩性



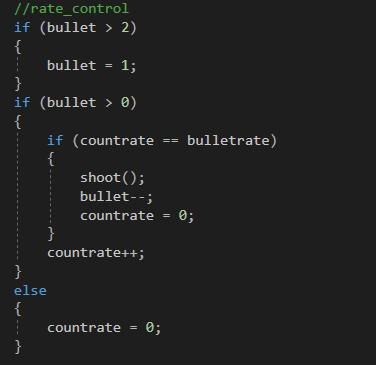
这是实现飞机移动的代码



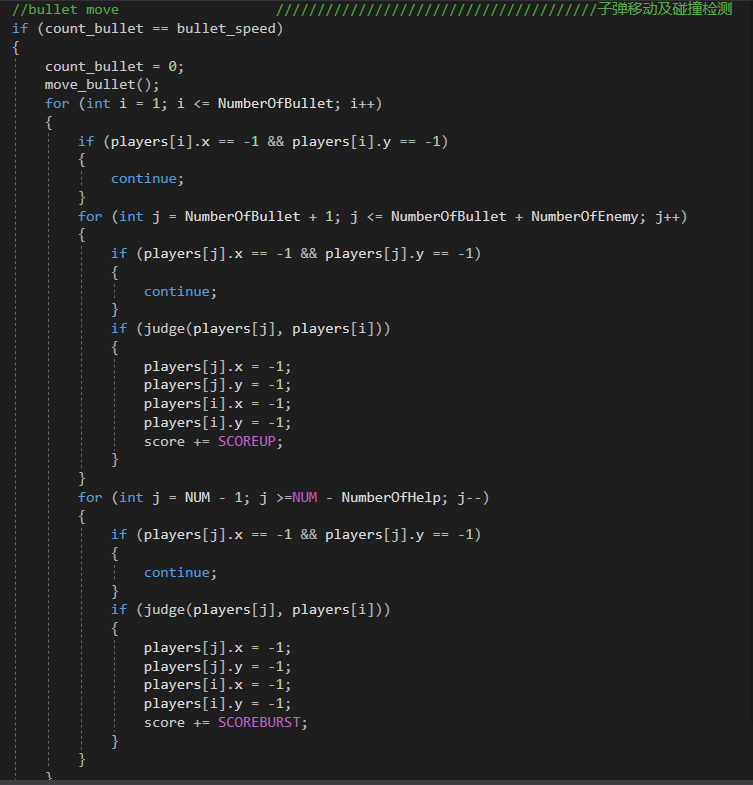
这是实现敌人移动及其碰撞检测的代码



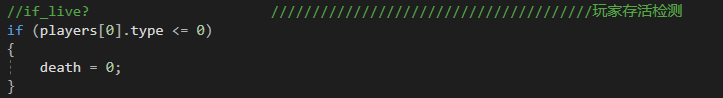
这是实现道具移动及其碰撞检测的代码



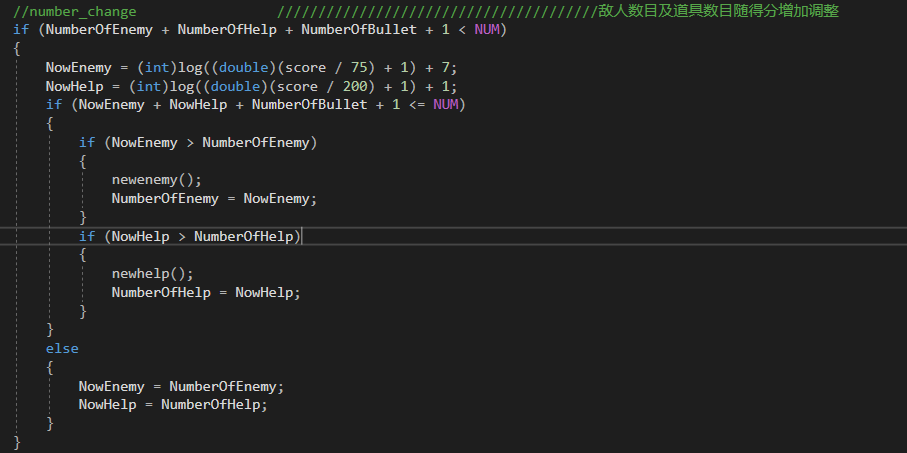
这是控制子弹速度的代码（连续按下键盘按键时匀速发射）



这是控制子弹移动及其碰撞检测的代码



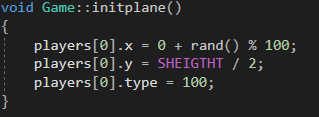
这是判断玩家是否存活的检测



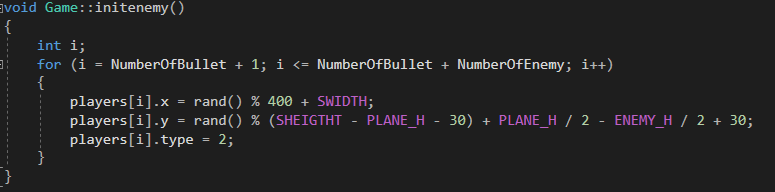
这是使敌人和道具数量随着得分增加而增加的代码

## 开始游戏

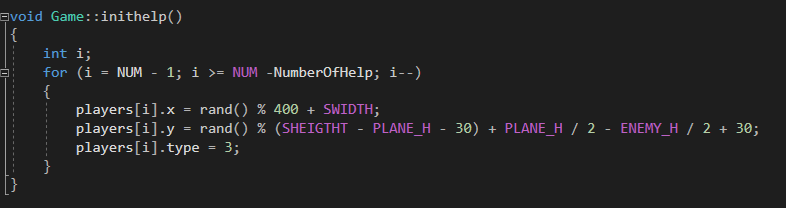
在菜单界面玩家选择了“开始游戏play”选项后，会初始化界面到游戏主窗体

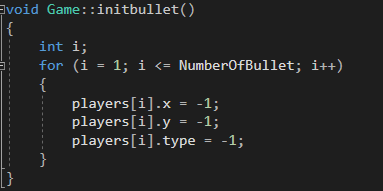


这是初始化飞机的代码



这是初始化敌人的代码

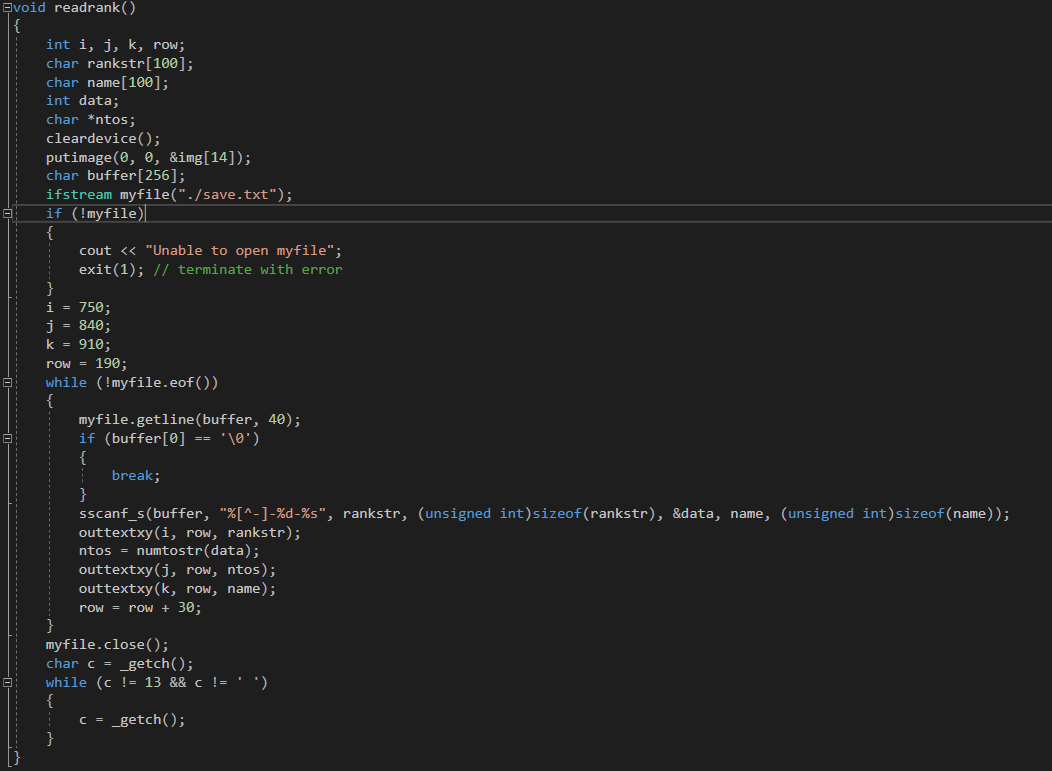
这是初始化道具的代码



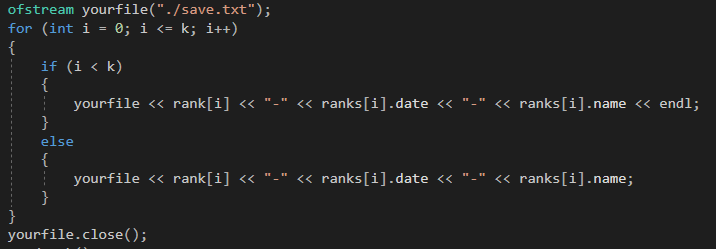
这是初始化子弹的代码

## 排行榜

我们采用了文件读写的方式来记录分数，并制作成了一张排行榜，其中分数的排序运用对结构体进行sort函数的操作来实现。



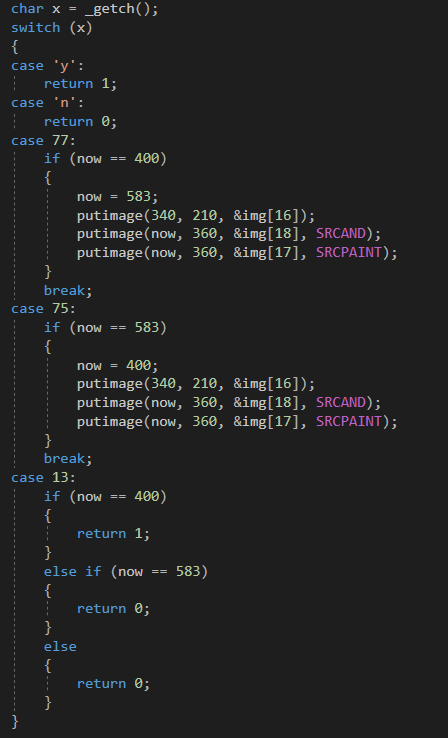
这是查看排行榜的代码



这是把当前分数记录进排行榜的代码

## 是否重新开始

在一局游戏结束时，可以选择是否重新开始游戏，这段代码用自定义函数ifrestart来实现。



这是ifrestart的主要代码

## 项目创新点

1. 使用了easyX的图形化界面
2. 使用多个while循环嵌套优化游戏界面选择
3. 主要游戏进行函数采用先进行数值修改再整体绘制
4. 文件读写采用了ifstream输入流和ofstream输出流，从而实现了本地排行榜的存储
5. 运用了Github来进行项目的合作，提高了信息互通的效率。

## 收获和建议

**沈建鑫**:我们小组这次的作业是《airwar》，这是一款模拟雷电而编写的飞机大战游戏。通过这次学习和实践，我深入了解了easyX中各个函数的作用，对c++中ifstream输入流和ofstream输出流有了更加熟练的应用，并且在图形绘制界面上的操作也更加成熟。复习了循环结构、选择结构、对结构体进行sort排序等操作，也对程序的格式规范有了一定的标准。

在程序设计开始前我们毫无头绪，之后我们通过组内会议的方法明确了实践的方向，并且商量了程序的大致架构和明确了分工，这让我深刻地体会到了团队合作的重要性。在之后的设计中，发现了问题我们会交流讨论，然后一起学习，由先学会的人给不了解的人进行讲解，保证团队的共同进步，大大地提高了学习的效率。在这期间，我们也学会了使用流程图和伪代码配合进行讲解的方式，提高了自身的表达能力。

在进行文件操作的时候我碰到了经常会出现乱码的问题，通过上网查阅资料发现主要是由字符串最后没有'\0'、文件打开后没有关闭、排序时多排了一组数据等问题所导致的，经过debug和仔细查看程序后，我成功解决了问题。这点让我明白了多利用学习资源自学的重要性，教科书不可能交给我们所有知识，一旦发现未知情况后就要通过自学去解决问题。

总结一下，此次实践设计培养了我们的团队合作能力、自学能力、整体程序规划能力、模块化程序设计能力，使我们对本学期所有的知识有了一定的巩固，也学到了更多新的知识。本次实践作业还有一点不足之处是只能在本地运行，不能在云端上使所有人都能参与并且记录分数。我们应该把视野放宽，在未来的学习中多学习一些关于服务器和数据库的知识，使该程序能在云端上进行。

**秦嘉珩**:我们小组这次的作业是《airwar》，一款基于雷电而编写的飞机大战游戏。通过这次学习和实践，我对easyX中各个函数有了了解，也对easyx中图形化界面和游戏逻辑设计有了进一步的认识，并且在文件输入输出上的操作也更加规范。在实践过程中我巩固了控制结构相关应用以及stl应用等操作，也培养了我规范程序格式的习惯。

在本次实践过程中，我们遇到问题能够相互帮助，通过讨论解决问题。我认为这是一个团队应该具备的能力，不仅可以使我们之间的默契度提升，也可以让相关程序段之间的联系更加紧密。

在进行游戏逻辑设计的时候，我遇到了运行不流畅以及输出画面不连贯等问题，后通过多次测试以及修改逻辑框架解决问题。虽然并没有完全解决适配性卡顿问题，但在一定性能保证的前提下可以流畅运行，且相关数值设置符合正常游戏体验。这让我明白了构建游戏框架的重要程度，以及软件适配的困难。未来的学习中，我也要不断培养寻找问题并解决问题的能力。

在这次实践中，一些问题也暴露了出来，首先，该程序在低性能计算机上运行则会出现卡顿掉帧等现象。我们应该多学习一些优化方式，使得游戏可以更好地适配硬件。其次，我们的排行榜只能在本地保存，这大大降低了程序的广泛性，我们应该更加深入地学习，在云端上实现数据的存储和读写。

**倪梓皓**:我们小组这次的作业是《airwar》，这是一款模仿雷电而编写的飞机大战游戏。通过这次学习和实践，我深入了解了如何使用第三方绘图函数库，也对第三方函数库与集成编译环境的链接有了更加熟练的应用。与此同时，我对本学期程序设计课上学到的相关知识作了巩固，并且将许多理论上的知识运用到了实践中。

尽管在实践项目真正敲定前我们小组的工作毫无条理，但在之后我们通过组内会议的方法明确了实践的方向，并且商量了程序的大致架构，也明确了分工，此后的工作开始有条不紊，进展迅速。这让我深刻地体会到了团队成员之间互相了解、互相沟通的重要性。在之后的设计中，发现了问题我们会及时交流讨论，然后一起学习，并合作解决，保证团队的共同进步，以及总体进程的稳步进行，大大地提高了学习的效率。在这期间，多次修改程序流程图也提升了我们使用流程图来绘制程序整体架构的能力，加强了对程序的理解。

在进行碰撞检测代码架构的时候我经常碰到出现奇特的碰撞判定区的问题，通过大量的Debug操作，以及网上查阅相关公式，进而思索出现问题的原因，从而优化计算公式，构建出了合理的碰撞模型，最终解决了飞机会在奇特的位置强制死亡的问题。这点让我明白了多思考以及构建高效的Debug方法的重要性。

总结来说，此次实践设计使我们的各个方面都有了长足的进步，使我们对本学期学习的C语言知识有了一定的巩固，同时也初步了解了C++的部分特性。但也正如老师所指导的那样，本次实践作业还有一点不足之处，那就是该游戏只能在本地运行，不能使网络上的玩家们一起参与游戏并且记录排行榜。我们应该把视野放宽，在未来的学习中也可以留心数据库与网络方面的知识，以期能够使我们的作品在网络上供人联机使用。

## 附：源代码清单

#include<iostream>

#include<easyx.h>

#include<graphics.h>

#include<windows.h>

#include<conio.h>

#include<time.h>

#include<string>

#include<cmath>

#include<fstream>

#include<stdlib.h>

#include<algorithm>

#include<mmsystem.h>

#pragma comment(lib,"WINMM.lib")

using namespace std;

//宏定义                                                 ////////////////////////////

#define MAXSTAR                 300                     //星星总数

#define NUM                     140                     //玩家+子弹+敌人+道具 总数

#define SWIDTH                  1080                    //屏幕宽度

#define SHEIGTHT                640                     //屏幕高度

#define PLANE\_W                 70                      //玩家宽度

#define PLANE\_H                 70                      //玩家高度

#define ENEMY\_W                 40                      //敌人宽度--道具宽度

#define ENEMY\_H                 40                      //敌人高度--道具高度

#define BULLET\_W                15                      //子弹宽度

#define BULLET\_H                15                      //子弹高度

#define PLANESPEED              18                      //飞机移动速度

#define HPHELP                  30                      //道具恢复生命数

#define HPMAX                   200                     //生命值上限

#define HPDOWN                  40                      //敌机碰撞生命损失

#define SCOREUP                 25                      //正常加分

#define SCOREBURST              150                     //大加分

#define SCOREDOWN               100                     //分数下降

//全局变量声明                                           ///////////////////////////

typedef struct object

{

    int     x;

    int     y;

    int     type;

}object;

struct STAR

{

    double  x;

    int     y;

    double  step;

    int     color;

};

STAR star[MAXSTAR];

struct point

{

    char    name[100];

    int     date;

};

int     score;

IMAGE   img[20];

char    p[15];

//函数声明                               ///////////////////////////////////////

int menu();                             //绘制菜单界面

bool box();                             //绘制问询界面

void ifsave();                          //绘制保存得分/是否保存界面

bool ifrestart();                       //绘制是否重新开始界面

void readrank();                        //绘制得分查询界面

void load();                            //加载图片资源

void InitStar(int);                     //背景星空绘制

void MoveStar(int);                     //背景星空绘制

char \*numtostr(int);                    //数字转字符串，用于打印得分/生命值

int cmp(point, point);                  //比较函数

//Game类                                ///////////////////////////////////////

class Game

{

public:

    //变量声明                           ///////////////////////////////////////

    object players[NUM];                //player  bullets  enemys  ...    helps

    int NumberOfEnemy;                  //敌人数目

    int NumberOfBullet;                 //子弹数目

    int NumberOfHelp;                   //道具数目

    int NowEnemy;                       //敌人数目变化参数

    int NowHelp;                        //道具数目变化参数

    bool death;                         //标记死亡  1 存活  0 死亡

    //函数声明                           ///////////////////////////////////////

    void initplane();                   //初始化玩家

    void initenemy();                   //初始化敌人

    void initbullet();                  //初始化子弹

    void inithelp();                    //初始化道具

    void newenemy();                    //调整敌人时初始化

    void newhelp();                     //调整道具时初始化

    void move\_plane(char);              //玩家移动(char)

    void move\_plane(int);               //玩家移动(int)

    void move\_enemy();                  //敌人移动

    void move\_bullet();                 //子弹移动

    void move\_help();                   //道具移动

    void drawall();                     //游戏画面绘制

    void init();                        //游戏初始化

    void pause();                       //暂停

    void playing();                     //游戏主进程

    void endgame();                     //死亡界面绘制

    void printhelp();                   //暂停时绘制帮助界面

    bool judge(object,object);          //碰撞检测(object enemy\_or\_help,object player\_or\_bullet)

    void shoot();                       //发射子弹

};

//函数定义                               ///////////////////////////////////////

void Game::playing()

{

    int bullet\_speed = 1;               //子弹速度

    int enemy\_speed = 2;                //敌人速度

    int help\_speed = 2;                 //道具速度

    int bullet = 0;                     //将要发出的子弹数目

    int bulletrate = 30;                //子弹频率

    //count计数变量                      ////

    int countrate = 0;

    int count\_bullet = 0;

    int count\_enemy = 0;

    int count\_help = 0;

    ////////////////////////

    while (death)

    {

        //player move                   ////////////////////////////////玩家移动及按键检测

        if (\_kbhit())

        {

            char x = \_getch();

            switch (x)

            {

            case 'w':

            case 's':

            case 'a':

            case 'd':   move\_plane(x);          break;

            case 72:

            case 80:

            case 75:

            case 77:    move\_plane((int)x);     break;

            case ' ':   bullet++;               break;

            case 27:    printhelp();            pause();

            default:    break;

            }

        }

        //enemy move                    ////////////////////////////////敌人移动及碰撞检测

        if (count\_enemy == enemy\_speed)

        {

            count\_enemy = 0;

            move\_enemy();

            for (int i = NumberOfBullet + 1; i <= NumberOfBullet + NumberOfEnemy; i++)

            {

                if (players[i].x == -1 && players[i].y == -1)

                {

                    continue;

                }

                if (judge(players[i], players[0]))

                {

                    players[i].x = -1;

                    players[i].y = -1;

                    players[0].type -= HPDOWN;

                    score -= SCOREDOWN;

                    if (score < 0)

                    {

                        score = 0;

                    }

                }

            }

        }

        count\_enemy++;

        //help move                     ////////////////////////////////道具移动及碰撞检测

        if (count\_help == help\_speed)

        {

            count\_help = 0;

            move\_help();

            for (int i = NUM - 1; i >= NUM - NumberOfHelp; i--)

            {

                if (players[i].x == -1 && players[i].y == -1)

                {

                    continue;

                }

                if (judge(players[i], players[0]))

                {

                    players[i].x = -1;

                    players[i].y = -1;

                    players[0].type += HPHELP;

                    //限制生命值上限

                    if (players[0].type > HPMAX)

                    {

                        players[0].type = HPMAX;

                    }

                }

            }

        }

        count\_help++;

        //rate\_control

        if (bullet > 2)

        {

            bullet = 1;

        }

        if (bullet > 0)

        {

            if (countrate == bulletrate)

            {

                shoot();

                bullet--;

                countrate = 0;

            }

            countrate++;

        }

        else

        {

            countrate = 0;

        }

        //bullet move                   ////////////////////////////////子弹移动及碰撞检测

        if (count\_bullet == bullet\_speed)

        {

            count\_bullet = 0;

            move\_bullet();

            for (int i = 1; i <= NumberOfBullet; i++)

            {

                if (players[i].x == -1 && players[i].y == -1)

                {

                    continue;

                }

                for (int j = NumberOfBullet + 1; j <= NumberOfBullet + NumberOfEnemy; j++)

                {

                    if (players[j].x == -1 && players[j].y == -1)

                    {

                        continue;

                    }

                    if (judge(players[j], players[i]))

                    {

                        players[j].x = -1;

                        players[j].y = -1;

                        players[i].x = -1;

                        players[i].y = -1;

                        score += SCOREUP;

                    }

                }

                for (int j = NUM - 1; j >=NUM - NumberOfHelp; j--)

                {

                    if (players[j].x == -1 && players[j].y == -1)

                    {

                        continue;

                    }

                    if (judge(players[j], players[i]))

                    {

                        players[j].x = -1;

                        players[j].y = -1;

                        players[i].x = -1;

                        players[i].y = -1;

                        score += SCOREBURST;

                    }

                }

            }

        }

        count\_bullet++;

        //if\_live?                      ////////////////////////////////////玩家存活检测

        if (players[0].type <= 0)

        {

            death = 0;

        }

        //number\_change                 ////////////////////敌人数目及道具数目随得分增加调整

        if (NumberOfEnemy + NumberOfHelp + NumberOfBullet + 1 < NUM)

        {

            NowEnemy = (int)log((double)(score / 75) + 1) + 7;

            NowHelp = (int)log((double)(score / 200) + 1) + 1;

            if (NowEnemy + NowHelp + NumberOfBullet + 1 <= NUM)

            {

                if (NowEnemy > NumberOfEnemy)

                {

                    newenemy();

                    NumberOfEnemy = NowEnemy;

                }

                if (NowHelp > NumberOfHelp)

                {

                    newhelp();

                    NumberOfHelp = NowHelp;

                }

            }

            else

            {

                NowEnemy = NumberOfEnemy;

                NowHelp = NumberOfHelp;

            }

        }

        //draw                          ///////////////////////////////////////屏幕绘制

        drawall();

        Sleep(1);

    }

}

void Game::drawall()

{

    cleardevice();

    BeginBatchDraw();

    //draw stars

    for (int i = 0; i < MAXSTAR; i++)

    {

        MoveStar(i);

    }

    //draw enemy

    for (int i = NumberOfBullet + 1; i <= NumberOfBullet + NumberOfEnemy; i++)

    {

        if (players[i].x != -1 && players[i].y != -1)

        {

            putimage(players[i].x, players[i].y, &img[4], SRCAND);

            putimage(players[i].x, players[i].y, &img[3], SRCPAINT);

        }

    }

    //draw help

    for (int i = NUM - 1; i >= NUM - NumberOfHelp; i--)

    {

        if (players[i].x != -1 && players[i].y != -1)

        {

            putimage(players[i].x, players[i].y, &img[10], SRCAND);

            putimage(players[i].x, players[i].y, &img[9], SRCPAINT);

        }

    }

    //draw player

    if (players[0].type >= 0)

    {

        putimage(players[0].x, players[0].y, &img[2], SRCAND);

        putimage(players[0].x, players[0].y, &img[1], SRCPAINT);

    }

    //draw bullet

    for (int i = 1; i <= NumberOfBullet; i++)

    {

        if (players[i].x != -1 && players[i].y != -1 && players[0].type > 0)

        {

            putimage(players[i].x, players[i].y, &img[6], SRCAND);

            putimage(players[i].x, players[i].y, &img[5], SRCPAINT);

        }

    }

    //draw score

    RECT r[5] = { {60,0,120,30},{120,0,180,30},{200,0,260,30},{260,0,320,30},{360,0,720,30} };

    char \*s;

    char \*life;

    if (players[0].type > 0)

    {

        life = numtostr(players[0].type);

        drawtext(\_T("HP:"), &r[2], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

        drawtext(\_T(life), &r[3], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    }

    else

    {

        drawtext(\_T("HP:"), &r[2], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

        drawtext(\_T(" 0 "), &r[3], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    }

    s = numtostr(score);

    drawtext(\_T("得分:"), &r[0], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    drawtext(\_T(s), &r[1], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    drawtext(\_T("按ESC以暂停/获得帮助"), &r[4], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    EndBatchDraw();

}

void Game::init()

{

    int i;

    //初始化星星

    for (i = 0; i < MAXSTAR; i++)

    {

        InitStar(i);

        star[i].x = rand() % SWIDTH;

    }

    //游戏初始化数据

    NumberOfEnemy = 7;

    NumberOfBullet = 25;

    NumberOfHelp = 1;

    NowEnemy = 7;

    NowHelp = 1;

    score = 0;

    initplane();

    initbullet();

    initenemy();

    inithelp();

    death = 1;

    mciSendString("play playingmusic repeat from 0", NULL, 0, NULL);

}

void Game::initplane()

{

    players[0].x = 0 + rand() % 100;

    players[0].y = SHEIGTHT / 2;

    players[0].type = 100;

}

void Game::initenemy()

{

    int i;

    for (i = NumberOfBullet + 1; i <= NumberOfBullet + NumberOfEnemy; i++)

    {

        players[i].x = rand() % 400 + SWIDTH;

        players[i].y = rand() % (SHEIGTHT - PLANE\_H - 30) + PLANE\_H / 2 - ENEMY\_H / 2 + 30;

        players[i].type = 2;

    }

}

void Game::inithelp()

{

    int i;

    for (i = NUM - 1; i >= NUM -NumberOfHelp; i--)

    {

        players[i].x = rand() % 400 + SWIDTH;

        players[i].y = rand() % (SHEIGTHT - PLANE\_H - 30) + PLANE\_H / 2 - ENEMY\_H / 2 + 30;

        players[i].type = 3;

    }

}

void Game::initbullet()

{

    int i;

    for (i = 1; i <= NumberOfBullet; i++)

    {

        players[i].x = -1;

        players[i].y = -1;

        players[i].type = -1;

    }

}

void Game::newenemy()

{

    for (int i = NumberOfBullet + NumberOfEnemy + 1; i <= NumberOfBullet + NowEnemy; i++)

    {

        players[i].x = rand() % 400 + SWIDTH;

        players[i].y = rand() % (SHEIGTHT - PLANE\_H - 30) + PLANE\_H / 2 - ENEMY\_H / 2 + 30;

        players[i].type = 2;

    }

}

void Game::newhelp()

{

    for (int i = NUM - NumberOfHelp - 1; i >= NUM - NowHelp; i--)

    {

        players[i].x = rand() % 400 + SWIDTH;

        players[i].y = rand() % (SHEIGTHT - PLANE\_H - 30) + PLANE\_H / 2 - ENEMY\_H / 2 + 30;

        players[i].type = 3;

    }

}

void Game::move\_plane(char x)

{

    int dir[4][2] = { {0,-PLANESPEED}   //w 0

                     ,{0,PLANESPEED}    //s 1

                     ,{-PLANESPEED,0}   //a 2

                     ,{PLANESPEED,0} }; //d 3

    int dirx=0;

    switch (x)

    {

    case 'w':   dirx = 0;   break;

    case 's':   dirx = 1;   break;

    case 'a':   dirx = 2;   break;

    case 'd':   dirx = 3;   break;

    }

    players[0].x += dir[dirx][0];

    players[0].y += dir[dirx][1];

    if (players[0].x > SWIDTH - PLANE\_W)

    {

        players[0].x = SWIDTH - PLANE\_W;

    }

    if (players[0].x < 0 )

    {

        players[0].x = 0;

    }

    if (players[0].y > SHEIGTHT - PLANE\_H)

    {

        players[0].y = SHEIGTHT - PLANE\_H;

    }

    if (players[0].y < 30 )

    {

        players[0].y = 30;

    }

}

void Game::move\_plane(int x)

{

    int dir[4][2] = { {0,-PLANESPEED}   //w 0

                     ,{0,PLANESPEED}    //s 1

                     ,{-PLANESPEED,0}   //a 2

                     ,{PLANESPEED,0} }; //d 3

    int dirx = 0;

    switch (x)

    {

    case 72:    dirx = 0;   break;

    case 80:    dirx = 1;   break;

    case 75:    dirx = 2;   break;

    case 77:    dirx = 3;   break;

    }

    players[0].x += dir[dirx][0];

    players[0].y += dir[dirx][1];

    if (players[0].x > SWIDTH - PLANE\_W)

    {

        players[0].x = SWIDTH - PLANE\_W;

    }

    if (players[0].x < 0)

    {

        players[0].x = 0;

    }

    if (players[0].y > SHEIGTHT - PLANE\_H)

    {

        players[0].y = SHEIGTHT - PLANE\_H;

    }

    if (players[0].y < 30)

    {

        players[0].y = 30;

    }

}

void Game::move\_bullet()

{

    for (int i = 1; i <= NumberOfBullet; i++)

    {

        if (players[i].x == -1 && players[i].y == -1)

        {

            continue;

        }

        players[i].x += 5;

        if (players[i].x > SWIDTH)

        {

            players[i].x = -1;

            players[i].y = -1;

        }

    }

}

void Game::move\_enemy()

{

    for (int i = NumberOfBullet + 1; i <= NumberOfBullet + NumberOfEnemy; i++)

    {

        if (players[i].x == -1 && players[i].y == -1 && players[i].type == 2)

        {

            players[i].x = rand() % 400 + SWIDTH;

            players[i].y = rand() % (SHEIGTHT - PLANE\_H - 30) + PLANE\_H / 2 - ENEMY\_H / 2 + 30;

        }

        else

        {

            players[i].x -= 4;

            if (players[i].x < -ENEMY\_W)

            {

                players[i].x = -1;

                players[i].y = -1;

            }

        }

    }

}

void Game::move\_help()

{

    for (int i = NUM - 1; i >= NUM - NumberOfHelp; i--)

    {

        if (players[i].x == -1 && players[i].y == -1 && players[i].type == 3)

        {

            players[i].x = rand() % 400 + SWIDTH;

            players[i].y = rand() % (SHEIGTHT - PLANE\_H - 30) + PLANE\_H / 2 - ENEMY\_H / 2 + 30;

        }

        else

        {

            players[i].x -= 3;

            if (players[i].x < -ENEMY\_W)

            {

                players[i].x = -1;

                players[i].y = -1;

            }

        }

    }

}

void Game::shoot()

{

    for (int i = 1; i <= NumberOfBullet; i++)

    {

        if (players[i].x == -1 && players[i].y == -1 && players[i].type == -1 )

        {

            players[i].x = players[0].x + PLANE\_W;

            players[i].y = players[0].y + PLANE\_H / 2-BULLET\_H/2-1;

            break;

        }

    }

}

void Game::pause()

{

    char c = \_getch();

    while (c != 27 && c != ' ' && c != 101)

    {

        c = \_getch();

    }

    if (c == 101)

    {

        death = 0;

    }

}

bool Game::judge(object enemy, object porb)

{

    if (porb.type >= 0)

    {

        int enemy\_x = enemy.x + ENEMY\_W / 2;

        int enemy\_y = enemy.y + ENEMY\_H / 2;

        int plane\_x = porb.x;

        int plane\_y = porb.y;

        if (enemy\_x - plane\_x <= ENEMY\_W / 2 && enemy\_x - plane\_x >= -(ENEMY\_W / 2) && enemy\_y >= plane\_y - ENEMY\_W / 2 && enemy\_y <= plane\_y + PLANE\_H - 1 + ENEMY\_W / 2)

        {

            return 1;

        }

        else if (enemy\_y >= plane\_y && enemy\_y <= (plane\_y + PLANE\_H / 2 - 1) && enemy\_x - ENEMY\_W <= plane\_x + 2 \* (enemy\_y - plane\_y) && enemy\_x + ENEMY\_W >= plane\_x + 2 \* (enemy\_y - plane\_y))

        {

            return 1;

        }

        else if (enemy\_y >= plane\_y + PLANE\_H / 2 - 1 && enemy\_y <= (plane\_y + PLANE\_H - 1) && enemy\_x - ENEMY\_W <= plane\_x + 2 \* (plane\_y + PLANE\_H - 1 - enemy\_y) && enemy\_x + ENEMY\_W >= plane\_x + 2 \* (plane\_y + PLANE\_H - 1 - enemy\_y))

        {

            return 1;

        }

        else

        {

            return 0;

        }

    }

    else if (porb.type == -1)

    {

        int bullet\_x = porb.x+BULLET\_W/2;

        int bullet\_y = porb.y+BULLET\_H/2;

        int enemy\_x = enemy.x + ENEMY\_W / 2;

        int enemy\_y = enemy.y + ENEMY\_H / 2;

        if ((bullet\_x - enemy\_x)\*(bullet\_x - enemy\_x) + (bullet\_y - enemy\_y)\*(bullet\_y - enemy\_y) <= (BULLET\_W / 2 + ENEMY\_W / 2)\*(BULLET\_W / 2 + ENEMY\_W / 2))

        {

            return 1;

        }

        else

        {

            return 0;

        }

    }

    else

    {

        return 0;

    }

}

void Game::endgame()

{

    int times = 30;

    drawall();

    mciSendString("stop playingmusic", NULL, 0, NULL);

    mciSendString("play deathmusic", NULL, 0, NULL);

    while (times--)

    {

        Sleep(100);

        drawall();

    }

    BeginBatchDraw();

    putimage(0, 0, &img[8], SRCAND);

    putimage(0, 0, &img[7], SRCPAINT);

    EndBatchDraw();

    while (getchar() != '\n');

    mciSendString("stop deathmusic", NULL, 0, NULL);

}

void Game::printhelp()

{

    putimage(0, 0, &img[12], SRCAND);

    putimage(0, 0, &img[11], SRCPAINT);

}

int menu()

{

    int nowj;   //当前选项框所在位置右下角纵坐标

    RECT r[3] = { {661, 300, SWIDTH, 370},{661, 370, SWIDTH, 440},{661, 440, SWIDTH, 510} };

    putimage(0, 0, &img[13]);

    settextstyle(28, 0, \_T("微软雅黑"));

    nowj = 0;

    drawtext(\_T("> 开始游戏PLAY <"), &r[0], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    drawtext(\_T("查看排名READ"), &r[1], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    drawtext(\_T("退出游戏EXIT"), &r[2], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

    while (1)

    {

        if (\_kbhit())

        {

            char x = \_getch();

            if (x < 0)

                x = \_getch();

            switch (x)

            {

            case 'w':

            case 72:

                if (nowj == 1)

                {

                    nowj = 0;

                    drawtext(\_T("> 开始游戏PLAY <"), &r[0], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                    drawtext(\_T("     查看排名READ     "), &r[1], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                }

                else if (nowj == 2)

                {

                    nowj = 1;

                    drawtext(\_T("> 查看排名READ <"), &r[1], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                    drawtext(\_T("     退出游戏EXIT     "), &r[2], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                }

                break;

            case 's':

            case 80:

                if (nowj == 0)

                {

                    nowj = 1;

                    drawtext(\_T("     开始游戏PLAY     "), &r[0], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                    drawtext(\_T("> 查看排名READ <"), &r[1], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                }

                else if (nowj == 1)

                {

                    nowj = 2;

                    drawtext(\_T("     查看排名READ     "), &r[1], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                    drawtext(\_T("> 退出游戏EXIT <"), &r[2], DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE);

                }

                break;

            case 'k':

            case 32:

            case 13:

                return nowj + 1;

            default :   break;

            }

        }

    }

}

void load()

{

    //预加载

    loadimage(&img[0], \_T("./pic/openimg.jpg"), SWIDTH, SHEIGTHT);

    loadimage(&img[1], \_T("./pic/plane.png"), PLANE\_W, PLANE\_H);

    loadimage(&img[2], \_T("./pic/planemask.png"), PLANE\_W, PLANE\_H);

    loadimage(&img[3], \_T("./pic/enemy.png"), ENEMY\_W, ENEMY\_H);

    loadimage(&img[4], \_T("./pic/enemymask.png"), ENEMY\_W, ENEMY\_H);

    loadimage(&img[5], \_T("./pic/bullet.png"), BULLET\_W, BULLET\_H);

    loadimage(&img[6], \_T("./pic/bulletmask.png"), BULLET\_W, BULLET\_H);

    loadimage(&img[7], \_T("./pic/death.png"), SWIDTH, SHEIGTHT);

    loadimage(&img[8], \_T("./pic/deathmask.png"), SWIDTH, SHEIGTHT);

    loadimage(&img[9], \_T("./pic/help.png"), ENEMY\_W, ENEMY\_H);

    loadimage(&img[10], \_T("./pic/helpmask.png"), ENEMY\_W, ENEMY\_H);

    loadimage(&img[11], \_T("./pic/pause.png"), SWIDTH, SHEIGTHT);

    loadimage(&img[12], \_T("./pic/pausemask.png"), SWIDTH, SHEIGTHT);

    loadimage(&img[13], \_T("./pic/background.png"), SWIDTH, SHEIGTHT);

    loadimage(&img[14], \_T("./pic/rankbg.png"), SWIDTH, SHEIGTHT);

    loadimage(&img[15], \_T("./pic/ifsave.png"));

    loadimage(&img[16], \_T("./pic/ifrestart.png"));

    loadimage(&img[17], \_T("./pic/tinebox.png"));

    loadimage(&img[18], \_T("./pic/tineboxmask.png"));

    loadimage(&img[19], \_T("./pic/sorry.png"));

    //music注释

    mciSendString("open ./music/enter.mp3 alias entermusic", NULL, 0, NULL);

    mciSendString("open ./music/playing.mp3 alias playingmusic", NULL, 0, NULL);

    mciSendString("open ./music/death.mp3 alias deathmusic", NULL, 0, NULL);

    //开始播放enter\_music

    mciSendString("play entermusic", NULL, 0, NULL);

    putimage(0, 0, &img[0]);

    getchar();

    mciSendString("stop entermusic", NULL, 0, NULL);

    mciSendString("close entermusic", NULL, 0, NULL);

    cleardevice();

}

void readrank()

{

    int i, j, k, row;

    char rankstr[100];

    char name[100];

    int data;

    char \*ntos;

    cleardevice();

    putimage(0, 0, &img[14]);

    char buffer[256];

    ifstream myfile("./save.txt");

    if (!myfile)

    {

        cout << "Unable to open myfile";

        exit(1); // terminate with error

    }

    i = 750;

    j = 840;

    k = 910;

    row = 190;

    while (!myfile.eof())

    {

        myfile.getline(buffer, 40);

        if (buffer[0] == '\0')

        {

            break;

        }

        sscanf\_s(buffer, "%[^-]-%d-%s", rankstr, (unsigned int)sizeof(rankstr), &data, name, (unsigned int)sizeof(name));

        outtextxy(i, row, rankstr);

        ntos = numtostr(data);

        outtextxy(j, row, ntos);

        outtextxy(k, row, name);

        row = row + 30;

    }

    myfile.close();

    char c = \_getch();

    while (c != 13 && c != ' ')

    {

        c = \_getch();

    }

}

bool box()

{

    int now;

    now = 400;

    //打印界面

    putimage(340, 210, &img[15]);

    putimage(now, 360, &img[18], SRCAND);

    putimage(now, 360, &img[17], SRCPAINT);

    while (1)

    {

        if (\_kbhit())

        {

            char x = \_getch();

            switch (x)

            {

            case 'y':

                return 1;

            case 'n':

                return 0;

            case 77:

                if (now == 400)

                {

                    now = 583;

                    putimage(340, 210, &img[15]);

                    putimage(now, 360, &img[18], SRCAND);

                    putimage(now, 360, &img[17], SRCPAINT);

                }

                break;

            case 75:

                if (now == 583)

                {

                    now = 400;

                    putimage(340, 210, &img[15]);

                    putimage(now, 360, &img[18], SRCAND);

                    putimage(now, 360, &img[17], SRCPAINT);

                }

                break;

            case 13:

                if (now == 400)

                {

                    return 1;

                }

                else if (now == 583)

                {

                    return 0;

                }

                else

                {

                    return 0;

                }

            }

        }

    }

    return 0;

}

void ifsave()

{

    point ranks[11];

    char buffer[256];

    char str[100];

    int k = 0;

    char name[100];

    char rank[10][7] = { {"No.01"},{"No.02"},{"No.03"},{"No.04"},{"No.05"},{"No.06"},{"No.07"},{"No.08"},{"No.09"},{"No.10"} };

    ifstream myfile("./save.txt");

    if (!myfile)

    {

        cout << "Unable to open myfile";

        exit(1); // terminate with error

    }

    while (!myfile.eof())

    {

        myfile.getline(buffer, 40);

        if (buffer[0] == '\0')

        {

            break;

        }

        sscanf\_s(buffer, "%[^-]-%d-%s", str, (unsigned int)sizeof(str), &ranks[k].date, ranks[k].name, (unsigned int)sizeof(ranks[k].name));

        k++;

    }

    myfile.close();

    readrank();

    if (k == 0)

    {

        if (score == 0)

        {

            putimage(340, 210, &img[20]);

            Sleep(2000);

            cout << "sorry low score" << endl;

        }

        else

        {

            if (box())

            {

                InputBox(name, 100, "请输入你的名字");

                if (name[0] == '\0')

                {

                    strcpy\_s(name, "anonymity");

                }

                ofstream yourfile("./save.txt");

                if (!yourfile)

                {

                    cout << "Unable to open myfile";

                    exit(1); // terminate with error

                }

                yourfile << "first" << "-" << score << "-" << name;

                yourfile.close();

                readrank();

            }

        }

    }

    else if(k < 10)

    {

        if (score == 0)

        {

            //分数太低，无法计入

            putimage(340, 210, &img[20]);

            Sleep(2000);

            cout << "sorry low score" << endl;

        }

        else

        {

            if (box())

            {

                InputBox(name, 100, "请输入你的名字");

                if (name[0] == '\0')

                {

                    strcpy\_s(name, "anonymity");

                }

                ranks[k].date = score;

                strcpy\_s(ranks[k].name, name);

                sort(ranks, ranks + k + 1, cmp);

                ofstream yourfile("./save.txt");

                for (int i = 0; i <= k; i++)

                {

                    if (i < k)

                    {

                        yourfile << rank[i] << "-" << ranks[i].date << "-" << ranks[i].name << endl;

                    }

                    else

                    {

                        yourfile << rank[i] << "-" << ranks[i].date << "-" << ranks[i].name;

                    }

                }

                yourfile.close();

                readrank();

            }

        }

    }

    else

    {

        if (score <= ranks[9].date)

        {

            //分数太低，无法计入

            putimage(340, 210, &img[20]);

            Sleep(2000);

            cout << "sorry low score" << endl;

        }

        else

        {

            if (box())

            {

                InputBox(name, 100, "请输入你的名字");

                if (name[0] == '\0')

                {

                    strcpy\_s(name, "anonymity");

                }

                ranks[k].date = score;

                strcpy\_s(ranks[k].name, name);

                sort(ranks, ranks + k + 1, cmp);

                ofstream yourfile("./save.txt");

                for (int i = 0; i < k; i++)

                {

                    if (i < k - 1)

                    {

                        yourfile << rank[i] << "-" << ranks[i].date << "-" << ranks[i].name << endl;

                    }

                    else

                    {

                        yourfile << rank[i] << "-" << ranks[i].date << "-" << ranks[i].name;

                    }

                }

                yourfile.close();

                readrank();

            }

        }

    }

}

bool ifrestart()

{

    int now;

    now = 400;

    putimage(340, 210, &img[16]);

    putimage(now, 360, &img[18], SRCAND);

    putimage(now, 360, &img[17], SRCPAINT);

    while (1)

    {

        if (\_kbhit())

        {

            char x = \_getch();

            switch (x)

            {

            case 'y':

                return 1;

            case 'n':

                return 0;

            case 77:

                if (now == 400)

                {

                    now = 583;

                    putimage(340, 210, &img[16]);

                    putimage(now, 360, &img[18], SRCAND);

                    putimage(now, 360, &img[17], SRCPAINT);

                }

                break;

            case 75:

                if (now == 583)

                {

                    now = 400;

                    putimage(340, 210, &img[16]);

                    putimage(now, 360, &img[18], SRCAND);

                    putimage(now, 360, &img[17], SRCPAINT);

                }

                break;

            case 13:

                if (now == 400)

                {

                    return 1;

                }

                else if (now == 583)

                {

                    return 0;

                }

                else

                {

                    return 0;

                }

            }

        }

    }

    return 0;

}

char \*numtostr(int n)

{

    int m = n;

    int j = 0;

    if (n == 0)

    {

        p[0] = '0';

        p[1] = '\0';

        return p;

    }

    while (n != 0)

    {

        j++;

        n = n / 10;

    }

    j--;

    n = j;

    while (m != 0)

    {

        p[j--] = m % 10+'0';

        m = m / 10;

    }

    p[n + 1] = '\0';

    return p;

}

int cmp(point a, point b)

{

    return a.date > b.date;

}

/////////////////////////stars

// 初始化星星

void InitStar(int i)

{

    star[i].x = SWIDTH;

    star[i].y = rand() % SHEIGTHT;

    star[i].step = (rand() % 5000) / 1000.0 + 1;

    star[i].color = (int)(star[i].step \* 255 / 6.0 + 0.5);  // 速度越快，颜色越亮

    star[i].color = RGB(star[i].color, star[i].color, star[i].color);

}

// 移动星星

void MoveStar(int i)

{

    // 计算新位置

    star[i].x -= star[i].step;

    if (star[i].x < 0)

    {

        InitStar(i);

    }

    // 画新星星

    putpixel((int)star[i].x, star[i].y, star[i].color);

}

//////////////////////////////

/////////////////////////////////////////////////////////////////////

int main()

{

    int choose;

    srand((unsigned)time(NULL));

    Game game;

    initgraph(SWIDTH, SHEIGTHT);

    load();

    while (1)

    {

        choose = menu();

        if (choose == 1)

        {

                while (1)

                {

                    game.init();

                    game.playing();

                    game.endgame();

                    ifsave();

                    if (!ifrestart())

                    {

                        break;

                    }

                }

        }

        else if (choose == 2)

        {

            readrank();

        }

        else if (choose == 3)

        {

            break;

        }

    }

    closegraph();

    mciSendString("close all", NULL, 0, NULL);

    return 0;

}

注释：本次程序设计实践各个函数基本都由讨论合作完成，其中排行榜相关部分及文件调用部分主要由沈建鑫完成，界面绘制及音乐文件的调用主要由秦嘉珩完成，碰撞检测判断主要由倪梓皓完成。

沈建鑫：电话：13777804148 qq：597238857

倪梓皓：电话：13777892703 qq：1151710764

秦嘉珩：电话：13777841404 qq：2454282105