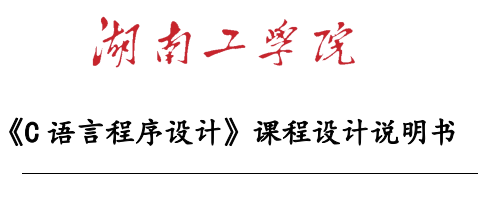
**迷宫数独游戏：小黄鸡逃脱**

学　　院： 湖南工学院

学　　号： 19020140203 姓名： 郑源昕

指导老师： 赵巾帼

专　　业： 计算机科学与技术

班　　级： 计科 1902

完成时间： 2020.6.20

成 　绩：

目录

[1. 概述 1](#_Toc43746445)

[1.1设计目的 1](#_Toc43746446)

[1.2任务描述与内容 1](#_Toc43746447)

[2. 分析与设计 2](#_Toc43746448)

[2.1函数功能描述 2](#_Toc43746449)

[3. 详细设计与实现 4](#_Toc43746450)

[3.1数据结构设计 4](#_Toc43746451)

[3.2游戏设计 6](#_Toc43746452)

[3.3界面设计 7](#_Toc43746453)

[4. 测试及运行结果 8](#_Toc43746454)

[5. 总结 11](#_Toc43746455)

[6. 参考 12](#_Toc43746456)

[7. 附录 13](#_Toc43746457)

# 概述

## 1.1设计目的

熟悉对图形库的使用，多线程创建以及对部分算法的应用。

## 1.2任务描述与内容

1、涉及算法与外来库：深度优先搜索（dfs）、路径压缩、带有权值的并查集、进制转换以及Kruskal算法。窗口、按钮等控件的实现则是依靠ege图形库。

2、游戏内容概述：探索迷宫并通过完成数独打败boss。除了最后一个Boss位置固定，其余两个都是随机放置在必经之路上的。

3、游戏过程流程图：

密码错误三次

走迷宫

完成数独打败boss

遇到boss

密码正确

管理员登陆成功

管理员用户名

输入用户名

验证密码

普通用户名

普通玩家登录

进入游戏

游戏结束，进入最后结算界面，将显示游戏用时和达成的结局

打败最终boss进入普通结局

选择留下进入特殊结局

遇到最终boss

重复两次后

6、管理员账号：Suolk

7、管理员密码：tgb951

# 分析与设计

## 2.1函数功能描述

1、并查集类（class Merge\_Find）内部函数：

Merge\_Find(int Size)：并查集初始化。

~Merge\_Find()：析构函数，用于释放初始化时申请的内存空间。

void init()：并查集数据初始化，在Merge\_Find(int Size)中调用。

int Find\_Root(int Index)：查找并返回该点所属的根节点。

int Path\_Find(int Index)：路径压缩，令所有子节点直接指向根节点。

bool Set\_Union(int set1,int set2)：集合的查找与合并。查找操作的实现是靠直接调用上述函数

int Set\_Quantity()：返回剩余的集合数目，用于判断合并是否完成。

2、子线程函数：

DWORD WINAPI MouseControl(LPVOID lpParameter)：用于处理迷宫中的鼠标信息。

3、其他自定义函数：

void oneselection(char s[100])：用于弹窗提示，只能选择确定，传入的字符串是提示内容。

bool twoselection(char s[100])：用于弹窗选择，有确定和取消两个选择，玩家选择确定则返回真，选择取消返回假，传入的字符串是提示内容。

void Mouse\_Click()：检测鼠标单击。

bool Cheat\_Judge()：若玩家选择作弊则进行判断，最后登录了管理员身份的返回真，否则返回假。

bool Exit\_Game() ：弹窗询问是否退出游戏，是则返回真，否返回假。

void Create\_Maze()：生成迷宫并将其存入二维数组。

void Button\_Draw() ：绘制两个全程通用的按钮

void Maze\_Draw() ：绘制迷宫

bool Find\_Path(int x,int y)：查找迷宫最短出路，传入入口坐标，找到出路返回真，碰壁则返回假。

void Boss\_pos(int pos1,int pos2)：在必经之路上放置boss，传入两个boss的坐标，坐标由随机数生成。

void Path\_Draw()：对于在迷宫阶段选择了作弊的玩家，绘制出迷宫路线。

void Story\_Start()：游戏开始时的故事背景交代。

void Erase(int x,int y)：用于迷宫阶段玩家移动时擦除原位置玩家形象，传入玩家原位置坐标。

void sd(int randnum)：用于生成数独小游戏。

void Maze\_Movement()：用于检测迷宫阶段的玩家移动。

void calculate\_PL()：计算迷宫最短路线的长度。

4、ege图形库函数见文末参考部分。

# 详细设计与实现

## 3.1数据结构设计

class Merge\_Find //并查集类

{

private:

int \*set; //集合

int \*value; //集合的权重

int size; //集合的大小，用于生成集合

int cnt; //集合数量

public:

Merge\_Find(int Size) //初始化

{

this->size=Size; //获得集合数量

set=new int[Size]; //申请内存

value=new int[Size];

init(); //数据初始化

}

~Merge\_Find() //析构函数

{

delete[] set; //释放内存

delete[] value;

}

void init() //数据初始化

{

cnt=size;

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

set[i]=i,value[i]=1;

}

int Find\_Root(int Index) //查找根节点

{

while(set[Index]!=Index)

Index=set[Index];

return Index;

}

int Path\_Find(int Index) //路径压缩

{

int Root=Find\_Root(Index);

while(set[Index]!=Root) //令子节点直接指向根节点

{

int id=Index;

Index=set[Index];

set[id]=Root;

}

return Root;

}

bool Set\_Union(int set1,int set2) //合并集合，返回值代表是否合并集合

{

int i=Path\_Find(set1),j=Path\_Find(set2);

if(i==j) //说明是同一个集合无需合并

return false;

cnt--; //若合并则集合数减少1

if(value[i]<value[j]) //将权重小的集合合并至权重大的集合中

{

value[j]+=value[i];

set[i]=j;

}

else

{

value[i]+=value[j];

set[j]=i;

}

return true;

}

int Set\_Quantity() //返回剩余集合数

{

return cnt;

}

};

struct Player

{

char Name[20]={0};

int highest\_record=86400; //最快通关时间

bool stay=false; //检测是否达成特殊结局

bool chosen[Chosen\_Len]={0}; //检查该数组矩阵用过没有

}player;

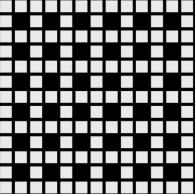
## 3.2游戏设计

1、游戏按键：在迷宫部分有两个按键，分别是“退出游戏”和“Cheat”。“Cheat”是作弊按键，如果在一开始登录游戏时输入的是管理员用户名，并且密码输入正确，那么就能使用作弊功能。在数独部分，除了退出和作弊，还新增了“提交”和“清空输入”两个按键。“提交”是用于完成数独后的提交检查，“清空输入”则是用于清空输入的全部数字，玩家可自行斟酌使用。“Cheat”按键在迷宫部分的作用是显示最短出路和boss所在位置，在数独部分是自动填入所有内容。

2、游戏操作：通过w、a、s、d移动角色探索迷宫；通过鼠标点击继续对话和触发按钮；通过键入数字完成数独。

3、多线程应用：由于迷宫环节既需要实时检测键入和鼠标点击，单线程无法满足线程，因此建立了一个独立的子线程用于检测迷宫环节的鼠标点击。

4、游戏内容（数独部分）：数独部分的生成基于游戏项目文件夹下的sd01~sd15这15个批处理文件，它们分别存储了15个数独矩阵的数据。鉴于通关全游戏只需要解三次数独，15个足以满足一定程度上的随机性。

5、游戏内容（迷宫部分）：迷宫部分是采取随机生成的模式，利用算法是并查集与kruskal，核心想法是，迷宫初始状态为，由一个空格加上周围8个位置都被墙包围的单位组成（如图，白为空格黑为墙）。那么要生成一个迷宫只需要确保每个空格之间都是连通的状态，因此想到了此前学习过的最小生成树算法。

## 3.3界面设计

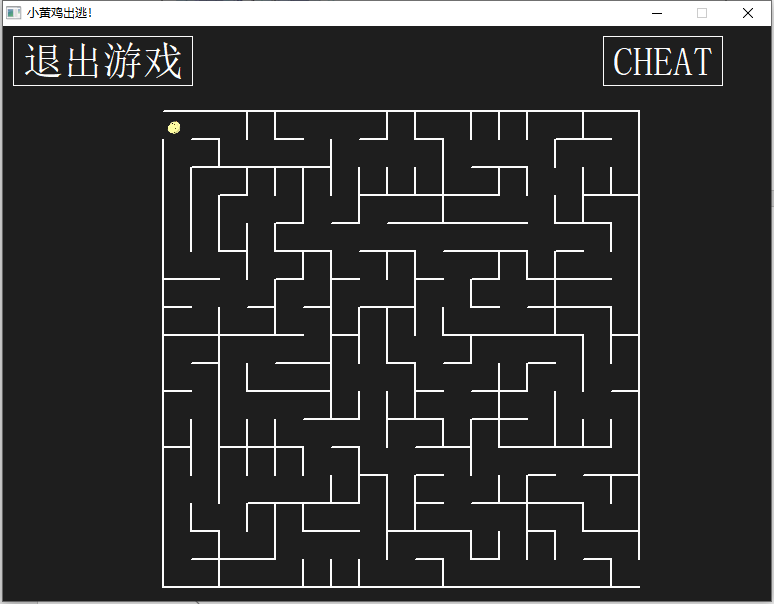
故事背景就是逃离Q版世界，因此游戏背景均为Q版人物和动物图像，界面实现主要是基于ege图形库的函数。音乐板块则是调用了api中的mcisendstring来实现。

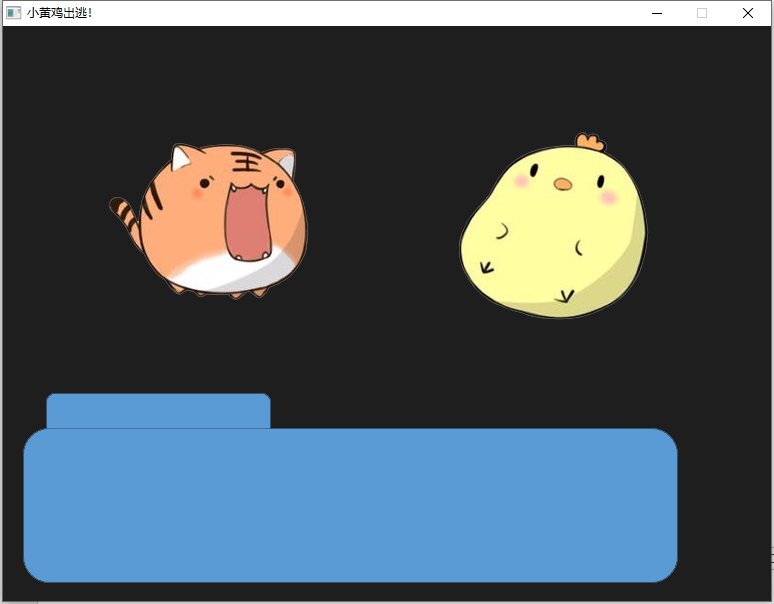
# 测试及运行结果

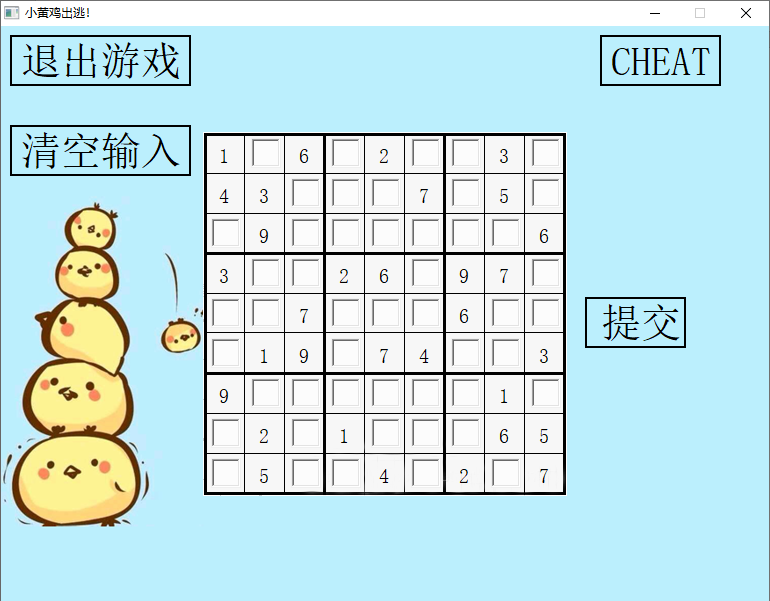
1. 测试运行环境：Windows 7
2. 编译器：Visual Studio Code
3. 程序测试运行：

登录界面：

游戏开始界面：

迷宫界面：

与游戏boss的对话界面（之一）：

数独界面：

# 总结

这次课程设计我受益颇多，一开始的想法是打算编写一个完整的剧情游戏，拥有完整的存档/读档功能，剧情更丰富圆满，同时玩法更多样，同时还有多次通关游戏会触发一些特殊结局。

但理想很丰满，现实很骨感。尽管一开始就规划好了游戏框架，但在图形界面、窗口控件等初步实现上就频频碰壁，在进行多方的资料查找之后，选择了导入ege图形库来完成实现了界面设计，在完成了Vs Code针对ege的重新配置后，解决了第一个问题。但接踵而来的问题只会多不会少，例如，游戏的趣味性很大程度来源于“随机”，但是实现随机性绝不是一个简单的随机数可以解决的，等等。最后由于一个人能力有限只实现了迷宫与数独两个模块，故事剧情也很单调且短暂。

尽管还有很多不足与遗憾，但我也从这次课程设计学到了许多，增强了自学能力和独立解决问题的能力，如果将来有时间与机会，一定会继续完善这个游戏的功能并增强其趣味性，同时丰富剧本，将这个游戏做的更好！

# 参考

1. ege图形库：

窗口实现

<https://blog.csdn.net/qq_39151563/article/details/104342295>

图像操作

<https://blog.csdn.net/qq_39151563/article/details/104342530>

文字打印

https://blog.csdn.net/qq\_39151563/article/details/104342672

键盘消息

https://blog.csdn.net/qq\_39151563/article/details/104342706

鼠标消息

https://blog.csdn.net/qq\_39151563/article/details/104342730

基础绘图

https://blog.csdn.net/qq\_39151563/article/details/104342390

2、音乐来源（主要）：炼狱庭园音乐包

3、数独矩阵： sudoku

# 附录

1. 程序源码链接：<https://paste.ubuntu.com/p/Xk2TvrgWCx/>
2. 程序源码（由于编译器与word文档页宽不同，排版变形严重，如有需要可直接点击上方链接阅读源码）：

#include <time.h>

#include <random>

#include <vector>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#include <mmsystem.h>

#include <algorithm>

#include <graphics.h>

#include <sys\_edit.h>

#define Chosen\_Len 15 //数独矩阵个数

#define Maze\_Size 17 //迷宫一行的空格数

#define Wall\_Num 544 //迷宫最多能有多少面墙

#define Maze\_Max\_Num 35 //一行墙和空格的总数

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

bool S\_login=false,Exit=false,Path\_View=false;

//按顺序分别为检测管理员登录与否，是否选择退出游戏，是否显示迷宫出路

int password\_count=0,X\_Player=1,Y\_Player=1;

//按顺序分别为管理员密码输入错误次数，迷宫的玩家x，y坐标

int maze[Maze\_Max\_Num+1][Maze\_Max\_Num+1];

//迷宫状态储存矩阵

int dire[4][2]={{-1,0},{1,0},{0,1},{0,-1}};

//搜索路径时的方向数组

int Path\_Length=-2;

//迷宫出路的长度,初始化为-2确保生成的boss位置不超过终点

double Start\_Time,End\_Time; //计时器

enum Maze\_Mark{Blank=0,Wall,Path,Boss};

//枚举，用于迷宫状态存储

class Merge\_Find //并查集类

{

private:

int \*set; //集合

int \*value; //集合的权重

int size; //集合的大小，用于生成集合

int cnt; //集合数量

public:

Merge\_Find(int Size) //初始化

{

this->size=Size; //获得集合数量

set=new int[Size]; //申请内存

value=new int[Size];

init(); //数据初始化

}

~Merge\_Find() //析构函数

{

delete[] set; //释放内存

delete[] value;

}

void init() //数据初始化

{

cnt=size;

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

set[i]=i,value[i]=1;

}

int Find\_Root(int Index) //查找根节点

{

while(set[Index]!=Index)

Index=set[Index];

return Index;

}

int Path\_Find(int Index) //路径压缩

{

int Root=Find\_Root(Index);

while(set[Index]!=Root) //令子节点直接指向根节点

{

int id=Index;

Index=set[Index];

set[id]=Root;

}

return Root;

}

bool Set\_Union(int set1,int set2) //合并集合，返回值代表是否合并集合

{

int i=Path\_Find(set1),j=Path\_Find(set2);

if(i==j) //说明是同一个集合无需合并

return false;

cnt--; //若合并则集合数减少1

if(value[i]<value[j]) //将权重小的集合合并至权重大的集合中

{

value[j]+=value[i];

set[i]=j;

}

else

{

value[i]+=value[j];

set[j]=i;

}

return true;

}

int Set\_Quantity() //返回剩余集合数

{

return cnt;

}

};

struct Player

{

char Name[20]={0};

int highest\_record=86400; //最快通关时间

bool stay=false; //检测是否达成特殊结局

bool chosen[Chosen\_Len]={0}; //检查该数组矩阵用过没有

}player;

void oneselection(char s[100]) //弹出只有一个确定选项的弹窗

{

MessageBox(0,s,"hint",0); //弹窗内容取决于传过来的字符串

return ;

}

bool twoselection(char s[100]) //弹出有确定和取消两个选项的弹窗

{

int a=MessageBox(0,s,"hint",1); //弹窗内容取决于传过来的字符串

if(a==1) //选择确定则返回真

return true;

else //选择取消返回假

return false;

}

void Mouse\_Click() //检测鼠标点击

{

mouse\_msg MOUSEMSG; //定义鼠标类变量

for(;is\_run();delay\_fps(180)) //进入循环，循环频率为每秒180次

{

MOUSEMSG=getmouse(); //获取鼠标信息

if(MOUSEMSG.is\_down()&&MOUSEMSG.is\_left())

break; ///检测到鼠标左键点击，退出循环

}

}

Bool Cheat\_Judge() //管理员作弊判断

{

if(S\_login) //已登录管理员

return true;

else if(password\_count<3&&!S\_login&&strcmp(player.Name,"Suolk")==0)

{//登录用户名为管理员用户名且密码错误少于三次

if(twoselection((char\*)"检测到您还未登录管理员身份，是否现在登录"))

{//选择登录管理员身份则生成密码输入框

sys\_edit PassWord; //定义一个输入框类

key\_msg S\_password\_msg; //定义一个键盘类

char Password[20]={0}; //定义一个字符串数组储存输入的密码

bool password\_right=false;

//定义一个“开关”用于判断输入密码是否正确，初始化状态代表不正确

PassWord.create(false);

//创建单行输入框（输入框变量被定义时输入框仍然不存在，需要进行创建操作）

PassWord.move(440,10); //设置输入框左上角坐标

PassWord.size(150,30); //设置输入框的大小

PassWord.setmaxlen(10); //设置最大输入长度

PassWord.setbgcolor(LIGHTBLUE); //设置输入框颜色

PassWord.setcolor(WHITE); //设置输入的文字颜色

PassWord.setfont(20,0,"宋体"); //设置输入的文字大小与字体

PassWord.settext("请输入密码"); //键入文字说明该输入框的目的

PassWord.visable(true); //设置输入框可见

PassWord.setfocus(); //设置输入框的光标

flushkey(); //清空此前的键盘信息

for (; is\_run(); delay\_fps(60)) //设置循环，循环频率是每秒60次

{

char hint[50]={0};

//定义字符串数组用于传送内容不确定的错误提示

S\_password\_msg=getkey(); //获取键盘信息

if(S\_password\_msg.msg==key\_msg\_down&&S\_password\_msg.key==key\_enter)

{//检测到键入enter键

password\_count++; //密码输入次数+1

PassWord.gettext(10,Password);

//获取输入在输入框内的文本

if(strcmp(Password,"tgb951")==0) //检测是否输入正确

{

oneselection((char\*)"密码正确！");

//输入正确则弹出该提示框

password\_right=true;

//将开关状态标记为正确

S\_login=true;

//将代表是否登录的全局变量标记为真

}

else if(password\_count<3)

{//密码输入错误，但是密码输入次数还有剩

sprintf(hint,"密码错误！你还有%d次机会",3-password\_count); //将错误提示写入hint数组

PassWord.settext(""); //清空输入框内容

PassWord.setfocus(); //放置光标

oneselection(hint); //弹出提示框

}

flushkey();

//清空键盘信息防止下次检测检测到无用的键盘信息

memset(Password,0,sizeof(Password));

//清空储存输入密码的字符串数组，防止下次检测时混入额外字符导致检测出错

}

else if(password\_right) //如果密码输入正确，则返回真

{

return true;

}

if(password\_count>=3)

{//如果密码输入次数达到上限且没有正确过，返回假

return false;

}

}

}

else //选择不登录管理员身份

{

oneselection((char\*)"作弊功能只能登录后使用！");

//弹出错误提示

return false;

}

}

else if(password\_count==3) //在此之前密码错误次数已经达到了三次

{

oneselection((char\*)"密码输入错误次数已达上限！无法登录！");

return false;

}

else //非管理员

{

oneselection((char\*)"作弊功能只能管理员使用！");

return false;

}

}

bool Exit\_Game() //弹窗询问是否退出游戏

{

if(twoselection((char\*)"确定退出游戏？"))

{

Exit=true; //将开关状态修改为退出

return true;

}

else

return false;

}

void Create\_Maze()

{//原理是用最小生成树生成联通道路，这里用的是并查集+kruskal

for(int i=0;i<=Maze\_Max\_Num;i++) //先将迷宫全部初始化为墙

for(int j=0;j<=Maze\_Max\_Num;j++)

maze[i][j]=Wall;

for(int i=1;i<Maze\_Max\_Num;i+=2) //再将必需的空格填入

for(int j=1;j<Maze\_Max\_Num;j+=2)

maze[i][j]=Blank;

std::vector<int> Random\_Sequence;

for(int i=0;i<Wall\_Num;i++) //设置一个vector数组用于生成随机序列

Random\_Sequence.push\_back(i);

shuffle(Random\_Sequence.begin(),Random\_Sequence.end(),std::default\_random\_engine(time(NULL))); //生成一个随机序列用于随机拆墙生成迷宫

Merge\_Find bcj(Maze\_Size\*Maze\_Size); //调用并定义一个并查集类

for(int i=0;i<Wall\_Num;i++)

{

if(bcj.Set\_Quantity()<=1) //只剩少于个集合说明道路已全部联通

break;

int num,Horizon=0,Vertical=0,Rand\_Wall\_Num,x\_Wall,y\_Wall;

if(Random\_Sequence[i]<Wall\_Num/2)

{//序列中编号为0~Wall\_Num/2-1的为横向墙

Rand\_Wall\_Num=Random\_Sequence[i];

Horizon=1;

num=Maze\_Size-1;

}

Else //序列中编号为Wall\_Num/2~Wall\_Num的为纵向墙

{

Rand\_Wall\_Num=Random\_Sequence[i]-Wall\_Num/2;

Vertical=1;

num=Maze\_Size;

}

x\_Wall=Rand\_Wall\_Num%num;

y\_Wall=Rand\_Wall\_Num/num;

int x1=x\_Wall,y1=y\_Wall,x2=x\_Wall+Horizon,y2=y\_Wall+Vertical;

if (bcj.Set\_Union(y1\*Maze\_Size+x1,y2\*Maze\_Size+x2))

{ //合并了集合说明需要拆墙

maze[y\_Wall+y2+1][x\_Wall+x2+1]=Blank;

}

}

}

void Button\_Draw() //绘制两个全程通用按钮

{

rectangle(600,10,720,60); //画一个空心的长方形

rectangle(10,10,190,60);

setfont(40,0,"宋体"); //设置字体大小

setbkmode(TRANSPARENT); //将文字背景设置为透明

xyprintf(20,15,"退出游戏"); //键入文字

xyprintf(610,15,"CHEAT");

setfont(20,0,"宋体"); //恢复成常用的字体大小

}

void Maze\_Draw() //绘制迷宫

{

PIMAGE BG\_Pic=newimage(768,575); //定义一个图片类型的指针

ege\_enable\_aa(true,BG\_Pic); //抗锯齿

getimage(BG\_Pic,"p\_Maze.png"); //获取图片

cleardevice(); //清屏

putimage(0,0,BG\_Pic); //放置图片

delimage(BG\_Pic); //销毁图片释放内存

BG\_Pic=NULL; //图片被销毁时指针仍然存在，但所指向应为空，为防止后面可能再调用该指针时出错，手动令其指向NULL

Button\_Draw(); //调用绘制按钮函数

setlinewidth(2.0); //设置线宽为2.0

setcolor(WHITE); //设置线为白色

for(int i=0;i<Maze\_Max\_Num;i+=2)

for(int j=1;j<Maze\_Max\_Num;j+=2)

if(maze[i][j]==Wall)

line(160+(j-1)\*14,85+i\*14,160+14\*(j+1),85+14\*i); //画横线

for(int i=1;i<Maze\_Max\_Num;i+=2)

{

for(int j=0;j<Maze\_Max\_Num;j+=2)

{

if((i==1&&j==0)||(i==33&&j==34))

continue; //限定入口在左上角和右下角因此这两个位置不画线

else if(maze[i][j]==Wall)

line(160+14\*j,85+14\*(i-1),160+14\*j,85+14\*(i+1)); //画竖线

}

}

}

bool Find\_Path(int x,int y) //用dfs回溯法来搜索离开迷宫的最短路线

{

if (maze[x][y]==Blank) //检测到空格

{

maze[x][y] =Path; //先标记为路线

if (x==Maze\_Max\_Num-2&&y==Maze\_Max\_Num-2)

return true; //到达出口，说明路线正确，返回真

for (int i=0;i<4;i++) //dire数组分别代表了四个方向

{

if(Find\_Path(x+dire[i][0],y+dire[i][1])) //递归调用

return true; //路线正确，返回真

}

maze[x][y]=Blank; //若四个方向都没有正确路线则说明这一格不是出迷宫的必经之路，重新标记为空格

}

return false; //进行到这一步说明遇到了路线不正确，返回假

}

void Boss\_pos(int pos1,int pos2) //在必经之路上随机放置两个boss

{

int cnt=0; //pos1和pos2代表Boss在路线的第几格

for(int i=0;i<Maze\_Max\_Num;i++)

{

for(int j=0;j<Maze\_Max\_Num;j++)

{

if(maze[i][j]==Path)

{

cnt++; //计算当前格是路线上的第几格

if(cnt==pos1) //放置boss1

maze[i][j]=Boss;

else if(cnt==pos2)

maze[i][j]=Boss; //放置boss2

else if(cnt==Path\_Length+2)

{

maze[i][j]=Boss; //放置最后一个boss

X\_LB=i,Y\_LB=j;

return ;

}

}

}

}

}

void Path\_Draw()

{//当管理员在迷宫模块选择作弊时将离开迷宫的路线画出来并标记boss的位置

setfillcolor(LIGHTBLUE); /路线的颜色是浅蓝色（实际上更类似于紫色）

for(int i=1;i<Maze\_Max\_Num;i++)

{

for(int j=1;j<Maze\_Max\_Num;j++)

{

if(i==X\_Player&&j==Y\_Player)

continue;

else if(maze[i][j]==Path) //检测到路线则画出一个小方块覆盖

bar(j\*14+150,i\*14+80,(j+1)\*14+150,(i+1)\*14+80);

else if(maze[i][j]==Boss) //检测到boss则更换颜色画个小方块覆盖

{

setfillcolor(RED); //boss是红色小方块

bar(j\*14+150,i\*14+80,(j+1)\*14+150,(i+1)\*14+80);

setfillcolor(LIGHTBLUE); //恢复颜色为路线颜色

}

}

}

}

void Story\_Start() //游戏开始时的故事背景交代

{

PIMAGE BG=newimage(768,575),p=newimage(203,197);

//定义图片类型的指针并获取背景图片和玩家形象图片

getimage(BG,"p\_Maze.png");

getimage(p,"p\_player.png");

putimage(0,0,BG); //放置背景图片

setfont(20,0,"宋体"); //设置字体和大小

setbkmode(TRANSPARENT); //将文字背景设置为透明

setcolor(WHITE); //设置文字颜色

xyprintf(100,205,"点击鼠标来进行剧情,按w、a、s、d来移动");

//在背景上打印文字

xyprintf(100,230,"移动的时候如果没反应，就切换成英文输入法");

Mouse\_Click(); //检测鼠标点击

cleardevice(); //清屏

putimage(0,0,BG); //再次铺上背景图片

delimage(BG); //背景图片不会再用到了，因此销毁图片释放内存

BG=NULL;

//图片被销毁时指针仍然存在，但所指向应为空，为防止后面可能再调用该指针时出错，手动令其指向NULL

xyprintf(100,205,"这一天，当你醒过来时，你照了照镜子想看看自己帅气的脸……");

Mouse\_Click();

putimage(175,0,p); //放置玩家形象图片

delimage(p);

p=NULL;

Mouse\_Click();

xyprintf(100,230,"?!你变成了一只鸡！还是Q版小黄鸡!");

Mouse\_Click();

xyprintf(100,255,"原来你误入了一个Q版世界，这里所有东西都是Q版的");

Mouse\_Click();

xyprintf(100,280,"而想要回去必须通过一个迷宫………");

Mouse\_Click();

}

void Erase(int x,int y) //角色在迷宫移动时擦除原角色位置，并在原角色位置绘制一个黑色小方块，用于标记走过的路

{

setfillcolor(BLACK); //设置颜色黑色

bar(x,y,x+14,y+14); //画一个实心方块

}

void sd(int randnum)

{//进入数独小游戏（数独并非随机生成，但randnum是随机数）

char sd[20]={0}; //定义字符串数组读取文件名，由于文件名不确定因此需要通过字符串数组中转

sprintf(sd,"sd%02d.bat",randnum); //写入字符串数组

FILE\* fp; //定义一个文件类型的指针

fp=fopen(sd,"r"); //读取数独矩阵存储文件

if(fp==NULL) //错误提示，说明缺少对应的数独文件

{

char hint[14]={0}; //同样是为了应对不确定的字符串而作为中转

sprintf(hint,"Can't Find %s",sd);

oneselection(hint); //弹窗

return ;

}

int sd\_square[9][9],sd\_mark[9][9]; //两个数组分别用于 储存数独矩阵，储存该位的数字是否要由程序打印（需要为1，不需要为0）

for(int i=0;i<9;i++) //将是否填入的状态解压存进mark中

{//由于将状态存入文件时用的是二进制转换为十进制的存入方式，因此读入需要解压

int a;

fscanf(fp,"%d",&a);

for(int j=8;j>=0;j--) //一个数字代表一行的状态

{ //由于位运算计算快于普通运算，因此采用位运算计算解压

sd\_mark[i][j]=(a&1); //位运算，相当于a%2

a>>=1; //位运算，相当于a/2

}

}

for(int i=0;i<9;i++) //将文件中的数独矩阵转存到square中

{

int a;

for(int j=0;j<9;j++)

{

fscanf(fp,"%d",&a);

sd\_square[i][j]=a;

}

}

//接下来是生成数独游戏界面

cleardevice(); //清屏

PIMAGE BG\_Pic=newimage(768,575); //定义图片指针

ege\_enable\_aa(true,BG\_Pic); //设置抗锯齿

getimage(BG\_Pic,"p\_sdBG.png"); //获取背景图片

putimage(0,0,BG\_Pic); //放置背景图片

delimage(BG\_Pic); //图片不会重复使用，因此直接销毁图片释放内存

BG\_Pic=NULL; //手动令其指向NULL

PIMAGE Square\_Pic=newimage(364,364);

sys\_edit Input[9][9]; //定义输入框数组变量

mouse\_msg MOUSEMSG; //定义鼠标变量

bool Wrong\_ans=false;

//定义用于标记错误答案的开关，初始状态代表没有错误答案

getimage(Square\_Pic,"p\_数独.png"); //获得数独格子图片

ege\_enable\_aa(true,Square\_Pic);

setcolor(BLACK); //设置文字颜色黑色

putimage(202,106,Square\_Pic); //放置数独格子图片

delimage(Square\_Pic); //销毁图片释放内存

Square\_Pic=NULL;

for(int i=0;i<9;i++) //对输入框进行设置

{

for(int j=0;j<9;j++)

{

if(sd\_mark[i][j]) //说明这个位置应该由程序填入

{

xyprintf(218+40\*j,120+40\*i,"%d",sd\_square[i][j]); //打印文字

}

else //说明这位置应该由玩家输入

{

Input[i][j].create(false);

//创建单行输入框（输入框变量被定义时输入框仍然不存在，需要进行创建操作）

Input[i][j].move(210+40\*j,112+40\*i); //设置输入框位置

Input[i][j].size(30,30); //设置输入框大小

Input[i][j].setbgcolor(WHITE); //设置输入框颜色

Input[i][j].setcolor(BLACK); //设置输入框文字颜色

Input[i][j].setmaxlen(2); //设置输入框最大输入长度

Input[i][j].settext(""); //设置输入框内容为空

Input[i][j].setfont(20,0,"宋体"); //设置输入框输入字体大小

Input[i][j].visable(true); //设置输入框可见

}

}

}

//绘制几个数独所需的按钮

setfont(40,0,"宋体"); //设置文字字体

setcolor(BLACK); //设置颜色

rectangle(10,100,190,150); //画一个空心的矩形

rectangle(585,272,685,322);

setbkmode(TRANSPARENT); //设置文字背景透明

xyprintf(20,105,"清空输入"); //打印文字

xyprintf(600,277,"提交");

setfont(20,0,"宋体"); //恢复为常用字体

Button\_Draw(); //绘制两个通用按钮

flushmouse(); //清空鼠标信息

for(;is\_run;delay\_fps(180))//检查点击

{

while(mousemsg()) //当检测到鼠标信息

{

MOUSEMSG=getmouse(); //获取鼠标信息

if(MOUSEMSG.is\_down()&&MOUSEMSG.is\_left())

{//当检测到鼠标左键按下

int x,y; //用于储存鼠标点击的坐标

char num[3]={0}; //用于储存输入的字符，由于输入的数字是以字符串形式读取，因此建立一个用于中转的数组

mousepos(&x,&y); //获取鼠标点击坐标

if(x>=585&&x<=685&&y>=272&&y<=322) /

{/点击位置为提交按钮所在范围

for(int i=0;i<9;i++)

{

for(int j=0;j<9;j++)

{

if(!sd\_mark[i][j]) //该位置应为玩家输入

{

Input[i][j].gettext(2,num); //获取玩家输入

if(num[0]-48!=sd\_square[i][j]) //输入不正确

{

Wrong\_ans=true;

//修改开关状态为有错误答案

break; //跳出循环

}

}

}

if(Wrong\_ans)

break; //有错误答案则不再检测下一行跳出循环

}

if(Wrong\_ans)

{

oneselection((char\*)"有错误答案！"); //弹出提示

Wrong\_ans=false; //恢复开关状态以进行下一次检测

continue; //跳过接下来的内容等待下一次提交

}

else

{

oneselection((char\*)"恭喜你完成了！"); //答案正确

player.chosen[randnum-1]=true;

//该数组矩阵已被完成，则标记为用过

return ; //离开函数

}

}

else if(x>=600&&x<=720&&y>=10&&y<=60)

{//选择作弊，管理员选项

if(Cheat\_Judge())

{

for(int i=0;i<9;i++) //自动填入答案

{

for(int j=0;j<9;j++)

{

if(!sd\_mark[i][j])

{

char t[1];

t[0]=sd\_square[i][j]+48;

Input[i][j].settext(t);

}

}

}

}

}

else if(x>=10&&x<=190&&y>=10&&y<=60)//选择退出

{

if(Exit\_Game())

{

return ;

}

}

else if(x>=10&&x<=190&&y>=100&&y<=150&&twoselection((char\*)"确定清空输入？"))//选择清空输入

{

for(int i=0;i<9;i++)

{

for(int j=0;j<9;j++)

{

if(!sd\_mark[i][j])

Input[i][j].settext("");

}

}

}

}

}

}

cleardevice();

}

DWORD WINAPI MouseControl(LPVOID lpParameter);//线程函数声明

void Maze\_Movement()

{

HANDLE h1;

PIMAGE bp=newimage(203,197),sp=newimage(14,14),boss;

//定义图像指针，分别为玩家形象（大），玩家形象（小），boss形象

PIMAGE temp1,talk=newimage(696,218),BG=newimage(768,575);

//中转用图片，对话框，背景

key\_msg KEYMSG; //定义键盘类

int cnt=0; //用于记录第几个boss

getimage(bp,"p\_player.png"); //获取玩家形象

putimage(sp,0,0,14,14,bp,0,0,203,197); //缩小玩家形象

putimage(164,94,sp); //在迷宫入口放置小玩家

getimage(talk,"p\_talk.png"); //获取对话框图片文件

getimage(BG,"p\_Maze.png"); //获取迷宫背景图片文件

Start\_Time=fclock(); //开始计时

flushkey(); //清空键盘信息

flushmouse(); //清空鼠标信息

for(;is\_run;delay\_fps(180)) //循环频率1秒180次

{

h1=CreateThread(NULL,0,MouseControl,NULL,0,NULL); //创建线程

CloseHandle(h1); //关闭线程句柄

while(kbmsg()) //检测到键盘信息产生

{

KEYMSG=getkey(); //获取键盘信息

if(KEYMSG.msg==key\_msg\_down) //检测到键盘按下

{

if(KEYMSG.key=='A'&&maze[X\_Player][Y\_Player-1]!=Wall)

{//键入a，且左边不是墙

Erase(Y\_Player\*14+150,X\_Player\*14+80);

//擦除原位置玩家图片

Y\_Player--; //玩家坐标更改

}

else if(KEYMSG.key=='D'&&maze[X\_Player][Y\_Player+1]!=Wall)

{//键入d，且右边不是墙

Erase(Y\_Player\*14+150,X\_Player\*14+80);

Y\_Player++;

}

else if(KEYMSG.key=='W'&&maze[X\_Player-1][Y\_Player]!=Wall)

{ //键入w，且上方不是墙

Erase(Y\_Player\*14+150,X\_Player\*14+80);

X\_Player--;

}

else if(KEYMSG.key=='S'&&maze[X\_Player+1][Y\_Player]!=Wall)

{//键入s，且下方不是墙

Erase(Y\_Player\*14+150,X\_Player\*14+80);

X\_Player++;

}

else//不知道玩家又瞎键入了什么奇怪的东西

continue;

putimage(150+Y\_Player\*14,80+X\_Player\*14,sp);

//在更新了的玩家位置上画出玩家形象

if(maze[X\_Player][Y\_Player]==Boss) //遇到了boss

{

cnt++,maze[X\_Player][Y\_Player]=Blank;

//将boss改为空格，防止有人干倒退回来出错

temp1=newimage(768,575); //创建图片大小

getimage(temp1,0,0,768,575);

//获取当前屏幕内容（即截屏后将图片存进temp1里）

if(cnt==1) //遇到第一个boss

{

boss=newimage(208,182); //创建图片大小

getimage(boss,"p\_boss1.png"); //获取一号boss形象

putimage(0,0,BG); //铺上背景

setcolor(WHITE); //设置字体颜色白色

setfont(20,0,"宋体"); //设置字体大小

setbkmode(TRANSPARENT); //设置字体背景透明

putimage(450,100,bp); //放置玩家形象（大）

putimage(10,355,talk); //放置对话框

xyprintf(60,375,player.Name); //打印玩家姓名

xyprintf(60,410,"啊迷宫里好黑，好像撞到了什么…");

//打印对话内容

Mouse\_Click(); //检测鼠标点击

putimage(100,100,boss); //放置boss形象

putimage(10,355,talk);

//再次放置对话框以覆盖掉前面打印的文字

Mouse\_Click(); //下同

putimage(10,355,talk);

xyprintf(100,375,"很凶的猛虎");

xyprintf(60,410,"什么人敢挡我的路！已经做好被我吃掉了的准备了吗！");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(60,375,player.Name);

xyprintf(60,410,"咦，这，这是小猫？好可爱");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(100,375,"很凶的猛虎");

xyprintf(60,410,"你在胡说什么！看不见我头上的王字吗！");

xyprintf(60,435,"我可是猛虎大人！");

xyprintf(60,460,"竟然敢说我是猫！决斗吧你这只可恶的小黄鸡！");

Mouse\_Click();

putimage(0,0,BG);

xyprintf(100,205,"完成数独来击败对手吧");

Mouse\_Click();

cleardevice(); //清屏

srand(time(NULL)); //引入时间作为随机数种子

while(true) //用于防止同一数独矩阵出现两次的循环

{

int rand\_num=rand()%Chosen\_Len+1;

//生成随机数

if(!player.chosen[rand\_num])

//若该随机数未出现过

{

sd(rand\_num);

//生成该数独矩阵并进行数独游戏

break;

}

}

if(Exit) //玩家在数独中选择了退出游戏

return ; //离开函数

putimage(0,0,BG); //放置背景图片

putimage(450,100,bp); //放置玩家形象（大）

putimage(100,100,boss); //放置boss

putimage(10,355,talk); //放置对话框

xyprintf(100,375,"很凶的猛虎"); //打印对话与名称

xyprintf(60,410,"这不可能…");

xyprintf(60,435,"无敌的猛虎大人居然输给了一只鸡…");

xyprintf(60,460,"呜哇啊啊啊啊啊啊啊");

Mouse\_Click(); //检测鼠标点击

putimage(0,0,BG);

//重新放置背景、对话框和玩家形象，营造出boss跑掉了的效果

putimage(450,100,bp);

putimage(10,355,talk);

xyprintf(60,375,player.Name);

xyprintf(60,410,"啊…跑掉了…");

xyprintf(60,435,"还想rua一下尾巴的…");

Mouse\_Click(); //检测到点击

delimage(boss); //销毁图片释放内存

putimage(0,0,temp1);

//放置此前截屏的画面，做出回到迷宫游戏界面的效果

delimage(temp1);

temp1=boss=NULL;

}

else if(cnt==2) //第二个boss

{

boss=newimage(186,161); //创建图片大小

getimage(boss,"p\_boss2.png");

//获取第二个boss的形象，其余操作均同上

putimage(0,0,BG);

setcolor(WHITE);

setfont(20,0,"宋体");

setbkmode(TRANSPARENT);

putimage(100,100,boss);

putimage(450,100,bp);

putimage(10,355,talk);

xyprintf(60,375,player.Name);

xyprintf(60,410,"啊咧？海豚原来可以在地上生存吗");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(100,375,"大概是蛇");

xyprintf(60,410,"本大爷是蛇啊！！！！是蛇！！！！");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(60,375,player.Name);

xyprintf(60,410,"啊这，可是……");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(100,375,"是蛇呢");

xyprintf(60,410,"你在质疑本大爷吗！！！！！");

xyprintf(60,435,"居然把本和那种水生动物搞混！！！！");

xyprintf(60,460,"本大爷可是爬行动物啊你这个呆头鹅！！！！");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(60,375,player.Name);

xyprintf(60,410,"不…我是鸡啊…");

Mouse\_Click();

srand(time(NULL));

while(true)

{

int rand\_num=rand()%Chosen\_Len+1;

if(!player.chosen[rand\_num])

{

player.chosen[rand\_num]=true;

sd(rand\_num);

break;

}

}

if(Exit)

return ;

putimage(0,0,BG);

putimage(450,100,bp);

putimage(100,100,boss);

putimage(10,355,talk);

xyprintf(100,375,"一条受到打击的蛇");

xyprintf(60,410,"本大爷…本大爷居然…");

xyprintf(60,435,"败给了一只鸭子？！！！一只水生生物？！！！");

xyprintf(60,460,"这是爬行动物的耻辱！！！！");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(60,375,player.Name);

xyprintf(60,410,"虽然但是…我看起来真的那么像水上生物吗");

Mouse\_Click();

delimage(boss);

putimage(0,0,temp1);

delimage(temp1);

boss=temp1=NULL;

}

else if(cnt==3) //最后一个boss

{

boss=newimage(225,200); //创建图片大小

getimage(boss,"p\_boss3.png");

//获取最终boss图片大小，其余操作均同上

putimage(0,0,BG);

setcolor(WHITE);

setfont(20,0,"宋体");

setbkmode(TRANSPARENT);

putimage(100,100,boss);

putimage(450,100,bp);

putimage(10,355,talk);

xyprintf(60,375,"一只无辜的小白兔");

xyprintf(60,410,"为什么一定要回去呢，留在这个世界不好吗……");

xyprintf(60,435,"难道你也讨厌我们吗");

Mouse\_Click();

if(twoselection((char\*)"要留下来吗"))

{//这里是一个小彩蛋

player.stay=true;

//如果选择留下则特殊成就开关标记为达成

putimage(10,355,talk);

//放置对话框和打印文字

xyprintf(60,375,"一只快乐的小白兔");

xyprintf(60,410,"（小白兔没有说话因为它很快乐）");

delimage(boss);

delimage(temp1);

delimage(BG);

boss=temp1=BG=NULL;

Mouse\_Click();

return ; //离开迷宫游戏

}

putimage(10,355,talk); //这是正常游戏流程

xyprintf(60,375,"一只不太妙的小白兔");

xyprintf(60,410,"果然人类都……");

Mouse\_Click();

xyprintf(60,435,"既然你不愿意留下来，那只好由我来帮你了");

Mouse\_Click();

putimage(10,355,talk);

xyprintf(100,375,player.Name);

xyprintf(60,410,"什——你怎么知道——");

Mouse\_Click();

srand(time(NULL));

while(true)

{

int rand\_num=rand()%Chosen\_Len+1;

if(!player.chosen[rand\_num])

{

player.chosen[rand\_num]=true;

sd(rand\_num);

break;

}

}

if(Exit)

return ;

putimage(0,0,BG);

putimage(100,100,boss);

putimage(450,100,bp);

putimage(10,355,talk);

xyprintf(60,375,"一只难过的小白兔");

xyprintf(60,410,"结果还是没有人愿意留下来陪我们……");

Mouse\_Click();

delimage(boss);

delimage(temp1);

delimage(BG);

boss=temp1=BG=NULL;

return ; //游戏结束

}

}

}

}

if(Exit)

return ;

}

}

void calculate\_PL() //计算迷宫路线长度，用于生成boss的位置

{

for(int i=0;i<Maze\_Max\_Num;i++)

for(int j=0;j<Maze\_Max\_Num;j++)

if(maze[i][j]==Path)

Path\_Length++;

}

int main()

{

FreeConsole(); //关闭控制台窗口

srand(time(NULL)); //获取随机数种子

int Name\_Length=0,randnum=0; //玩家姓名长度

char Password[20]={0}; //用于储存密码

key\_msg KEYMSG,S\_password\_msg; //定义键盘类

bool Key\_press\_flag=false,Mouse\_press\_flag=false,password\_right=false;

if(mciSendString(("open m\_Regrets.mp3 alias BGM"), NULL, 0, NULL))

{//打开音乐文件

oneselection((char\*)"OPEN BGM FAILED");

return 0;

}

mciSendString("play BGM", NULL, 0, NULL); //播放音乐

if(MessageBox(0,"Start Game Now?","hint",1)==2)

return 0;

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

//在这里停止播放否则在输入时会出现音乐卡顿

initgraph(400,300,0); //创建窗口

setbkcolor(WHITE); //设置背景颜色

setinitmode(INIT\_NOBORDER,0,0); //设置出口类型为无边框

setcaption("小黄鸡出逃！"); //设置窗口标题

PIMAGE BG\_Pic=newimage(); //定义图片指针

ege\_enable\_aa(true,BG\_Pic); //抗锯齿

getimage(BG\_Pic,"p\_login.png"); //获取登录背景图片

putimage(0,0,BG\_Pic); //放置背景

delimage(BG\_Pic); //销毁图片释放内存

BG\_Pic=NULL; //图片被销毁时指针仍然存在，但所指向应为空，为防止后面可能再调用该指针时出错，手动令其指向NULL

sys\_edit nameinput; //定义输入框类

nameinput.create(false); //创建单行输入框

nameinput.move(100,100); //设置位置

nameinput.size(200,30); //设置大小

nameinput.setmaxlen(10); //设置最大输入长度

nameinput.setbgcolor(BLUE); //设置输入框颜色

nameinput.setcolor(WHITE); //设置键入文字颜色

nameinput.setfont(20,0,"宋体"); //设置字体大小

nameinput.visable(true); //设置输入框可见

nameinput.setfocus(); //设置光标

setcolor(BLACK); //设置字体颜色

setfont(20,0,"宋体"); //设置字体大小

xyprintf(130, 50, "名字长度不超过%d",10);//打印文字

for (; is\_run(); delay\_fps(60))//管理员登录操作和普通玩家进入游戏

{

while(kbmsg())

{

KEYMSG=getkey();

if (KEYMSG.msg==key\_msg\_down&&KEYMSG.key==key\_enter)

{//捕捉到enter输入

nameinput.gettext(10,player.Name); //获取输入框内容

Name\_Length = strlen(player.Name); //获取姓名长度

Key\_press\_flag = true; //标记是否键入姓名

if(strcmp(player.Name,"Suolk")==0)//验证管理员登录

{

if(twoselection((char\*)"确定要以管理员身份登录吗?"))

{

nameinput.settext("请输入密码"); //显示输入框作用

nameinput.setfocus(); //设置光标

flushkey(); //清空键盘信息

for (; is\_run(); delay\_fps(60)) //循环频率为每秒60次

{

char hint[40]={0}; //中转用字符串数组

S\_password\_msg=getkey(); //获取键盘信息

if (S\_password\_msg.msg==key\_msg\_down&&S\_password\_msg.key==key\_enter)

{//键入enter

password\_count++; //密码输入次数+1

nameinput.gettext(10,Password);

//获取输入框内容

if(strcmp(Password,"tgb951")==0) {//检测密码是否正确

oneselection((char\*)"密码正确！");

password\_right=true; //密码正确

S\_login=true;

//管理员登陆状态修改为已登录

}

else if(password\_count<3)

{ //密码错误但还有输入机会

sprintf(hint,"密码错误！你还有%d次机会",3-password\_count);

nameinput.settext("");

nameinput.setfocus();

twoselection(hint);

}

else //密码错误且达上限

{

password\_right=false; //密码不正确

nameinput.settext("Suolk");

//将姓名重新置入输入框

}

flushkey();

memset(Password,0,sizeof(Password));

//清空内容防止密码检测出错

}

if(password\_right||password\_count==3)

{//密码正确或输入错误次数三次则离开密码的输入循环

break;

}

}

}

}

if(password\_right) //管理员已登录

{

break;

}

if(Name\_Length==0) //普通玩家进入游戏，但是没有输名字

{

oneselection((char\*)"请输入名称！");

nameinput.setfocus();

Key\_press\_flag=false;

}

else if(!twoselection((char\*)"确定以这个名称开始游戏吗?"))

{

Key\_press\_flag=false;

}

}

}

if(Key\_press\_flag) //说明已键入名称

{

break;

}

}

cleardevice(); //清屏

nameinput.destory(); //销毁输入框

if(mciSendString("open m\_CTS.mp3 alias BGM", NULL, 0, NULL))

{

oneselection((char\*)"OPEN BGM FAILED");

return 0;

}

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

initgraph(768,575); //重置窗口大小

setbkcolor(WHITE); //设置背景颜色

setcaption("小黄鸡出逃！"); //设置标题

BG\_Pic=newimage(768,575); //重新创建图片大小

getimage(BG\_Pic,"p\_开始游戏.png"); //获取图片文件

putimage(0,0,BG\_Pic); //放置背景图片

delay\_ms(1); //刷新一下屏幕

Mouse\_Click(); //检测鼠标点击

delimage(BG\_Pic); //销毁图片

BG\_Pic=NULL; //图片被销毁时指针仍然存在，但所指向应为空，为防止后面可能再调用该指针时出错，手动令其指向NULL

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

cleardevice(); //清屏

if(mciSendString("open m\_bgm.mp3 alias BGM", NULL, 0, NULL))

{

oneselection((char\*)"OPEN BGM FAILED");

return 0;

}

mciSendString("play BGM repeat",NULL,0,NULL); //循环播放背景音乐

Create\_Maze(); //创建迷宫

Story\_Start(); //故事开始界面

Find\_Path(1,1); //dfs搜索最短出路

calculate\_PL(); //计算出路长度

int pos\_a=rand()%Path\_Length+2,pos\_b=rand()%Path\_Length+2;

//通过随机数和路线长度计算boss在路线上的第几格

while(pos\_a==pos\_b) //确保boss的位置不重合

pos\_b=rand()%Path\_Length+2;

Boss\_pos(pos\_a,pos\_b); //放置boss

Maze\_Draw(); //绘制迷宫

Maze\_Movement(); //玩家进入迷宫开始游戏

End\_Time=fclock(); //游戏结束时间

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

if(Exit) //检测玩家是否选择退出游戏

return 0;

BG\_Pic=newimage(768,575); //重新获取背景图片

setcolor(BLACK); //设置文字颜色

setfont(30,0,"宋体"); //设置字体大小

setbkmode(TRANSPARENT); //设置字体背景透明

getimage(BG\_Pic,"p\_sdBG.png"); //获取图片文件

putimage(0,0,BG\_Pic); //放置背景图片

delimage(BG\_Pic); //销毁图片释放内存

BG\_Pic=NULL;

char hint[40]={0}; //中转字符串数组

if(player.stay) //玩家达成特殊结局

{

if(mciSendString("open m\_ding.mp3 alias BGM", NULL, 0, NULL))

{

oneselection((char\*)"OPEN BGM FAILED");

return 0;

}

mciSendString("play BGM",NULL,0,NULL); //特殊结局特殊音效

xyprintf(210,200,"达成特殊结局！");

xyprintf(210,235,"“它们实在太可爱了”");

Mouse\_Click();

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

}

else //达成普通结局

{

xyprintf(210,200,"你回到了现实");

xyprintf(210,235,"再也没有遇到什么特别的事");

xyprintf(210,270,"有时你会想");

xyprintf(210,305,"如果当时留下来，是不是会更好呢");

Mouse\_Click();

xyprintf(210,340,"达成结局：回到日常");

}

player.highest\_record=End\_Time-Start\_Time; //计算通关时间

sprintf(hint,"全程用时：%02d:%02d:%02d",player.highest\_record/3600,(player.highest\_record%3600)/60,player.highest\_record%60);

if(player.stay) //不同结局位置也不同

xyprintf(210,270,hint); //按xx:xx:xx的形式打印通关时间

else

xyprintf(210,375,hint);

Mouse\_Click();

return 0;

}

DWORD WINAPI MouseControl(LPVOID lpParameter)

{ //子线程用于管理迷宫中的鼠标信息，防止走迷宫时因为鼠标移动导致的键盘响应延迟

mouse\_msg MOUSEMSG; //定义鼠标类

while(mousemsg()) //检测到鼠标信息产生

{

MOUSEMSG=getmouse(); //获取鼠标信息

if(MOUSEMSG.is\_down()&&MOUSEMSG.is\_left()) //为鼠标点击

{

int x,y; //用于记录鼠标坐标

char num[2]={0};

mousepos(&x,&y); //获取坐标

if(x>=600&&x<=720&&y>=10&&y<=60) //选择作弊，管理员选项

{

if(Cheat\_Judge())

{

Path\_Draw(); //画出路线图

Path\_View=true; //标记路线状态为可见

}

}

else if(x>=10&&x<=190&&y>=10&&y<=60) //选择退出

{

if(Exit\_Game())

{

return 0L;

}

}

}

}

}