**西南大学**

**计算机与信息科学学院**

课程设计报告

《程序设计综合实践》

**题 目：** 欢乐扫雷

**年 级 、 专 业 ：** 20 2 1 **级** 计 科 中 外 **专 业** 3 **班**

**学 生 学 号 ：** 22 2 0 2 1 3 2 11 0 2 0 7 7

**学 生 姓 名 ：** 赵 浡 屹

**指 导 教 师 ：** 王 柯

**提 交 日 期 ：** 20 2 2 **年** 7 **月** 16 **日**

**目录**

目录

中文摘要及关键词英文摘要及关键词

1. 问题描述
2. 功能需求分析
   1. 主菜单
   2. 扫雷功能

相邻无雷区域一键翻开周围九宫格地雷数显示插旗问号标记功能

* 1. 计时器功能

游戏实时计时游戏时间记录

* 1. 游戏记录的存储和展示游戏记录存储

游戏记录展示

* 1. 音乐播放

1. 总体设计
   1. 主控模块设计
   2. **主要模块设计**
      1. 开始菜单模块
      2. 计时器模块
      3. 游戏记录存储模块
      4. 游戏记录展示模块
      5. 游戏贴图载入模块
      6. 游戏初始化模块
      7. 贴图模块
      8. 无雷区域一键翻开模块
      9. 炸雷模块
      10. 地雷数展示模块
      11. 鼠标操作接收模块
   3. 模块间调用关系设计
2. 详细设计
   1. 关键数据结构设计
   2. 主控模块main()流程图
   3. 主要模块详细设计
3. 开始菜单模块

MainMenu(HWND window)

实现流程

1. 计时器模块 TimeCounting(void \*none) 实现流程
2. 游戏记录存储模块 Recording(int pass) 实现流程
3. 游戏记录展示模块 ShowRecording() 实现流程
4. 游戏初始化模块 InitGame() 实现流程
5. 无雷区域一键展开模块(7)鼠标操作接收模块

BlankOpen(int r, int c)

play()

1. 系统开发

软件环境：

1. 开发环境
2. 第三方库3.Windows规格

硬件环境：

* 1. 主功能界面实现及代码
  2. 游戏界面实现及代码
  3. 计时器模块实现及代码
  4. 游戏记录存储模块实现及代码
  5. 游戏记录展示模块实现及代码

实现流程

实现流程

* 1. 无雷区域一键展开模块实现及代码

1. 测试
2. 问题与解决方案
3. 扫雷游戏中如何实现让用户只靠鼠标点击图形化界面完成游戏操作？
4. 用二维数组模拟不同状态的方格，如何让方格在点击前看上去一样？
5. 如何实现经典扫雷中的点击到无雷格子一键翻开周围连续无雷区域的功能？
6. 如何实现按钮悬停特效功能?
7. 如何实现音乐播放功能？
8. 游戏进行时程序一直在等待接收鼠标的信息，如何实现计时器实时计时并显示的功能？
9. 计时器线程如何将时间传递给游戏记录存储模块？
10. 为什么使用EasyX库中的outtextxy()等函数时VS提示没有与参数列表匹配的重载函数，无法通过编译？
11. 为什么在 Visual Studio 中一些常用的函数会无法通过编译，VS 会提示你使用另外的函数代替？
12. 参考文献

# 中文摘要及关键词

本项目使用主要使用C语言，结合第三方库 ，以经典扫雷为蓝本开发了一个新的扫雷游戏。

EasyX

游戏完全实现了周围九宫格地雷数显示，左键点击相邻无雷区域一键翻开，右键插旗和问号两种标记等经典扫雷的几乎全部功能，同时 在该基础上进行创新，设计了供玩家选择游戏模式的图形主界面，并且在游戏中加入了计时器，最近游戏记录展示，和自动保存游戏记录的 功能，还精心挑选了若干优美有趣的背景音乐和音效，增加玩家的游戏体验。

**关键词：**C语言，EasyX，扫雷

# 英文摘要及关键词

This project uses mainly C language, combined with the third-party library EasyX , to develop a new minesweeper game based on the classic minesweeper.

The game fully implements almost all the functions of classic minesweeper, such as displaying the number of mines in the

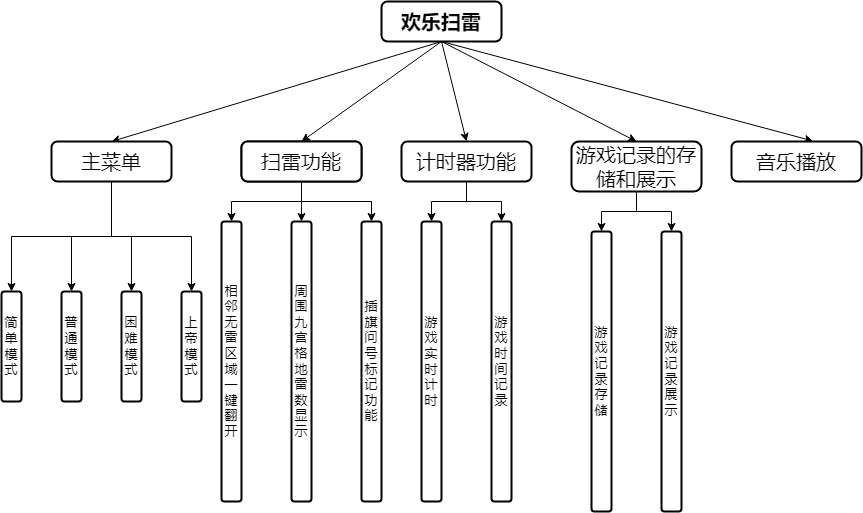
surrounding nine-box grid, left-clicking on adjacent mines-free areas to open them with one click, and right-clicking on both flags and question marks. A number of beautiful and interesting backgrou-nd music and sound effects have been carefully selected to increase players' gaming experience.

**Keywords:** C language, EasyX, minesweeper

**1. 问题描述**

使用C语言开发一个扫雷游戏，配合第三方图形库 EasyX ，开发一个包含图形界面的扫雷游戏。

**2. 功能需求分析**



* 1. **主菜单**

主菜单为图形化界面，有四个模式供玩家选择，点击按钮选择模式，前三个模式难度依次递增，第四个模式上帝模式需要输入正确的作 弊码才能进入（作弊码为本人完整学号 222021321102077 ），该模式下所有地雷将会被特殊标记。

* 1. **扫雷功能**

**相邻无雷区域一键翻开**

当左键点击到无雷的格子时，若该格子相邻区域的方块内也没有地雷，则所有相邻的无雷格子将被一键翻开，以免玩家做多余的点击。

**周围九宫格地雷数显示**

当无雷的格子被翻开时，翻开的格子上会显示以该格子为中心，周围九宫格内藏有的地雷数量。

**插旗问号标记功能**

当未翻开的格子被右键点击一次时，该格子将被红旗标记，此时该格子不可被左键点击翻开，被右键点击两次时，该格子将被问号标 记，此时左键点击该格子可被正常翻开，右键点击三次格子回到初始状态。

* 1. **计时器功能**

**游戏实时计时**

当玩家从主界面进入游戏后，游戏界面右侧将实时显示当前游戏进行的时间。

**游戏时间记录**

每一次游戏结束后，该局游戏所用时间将会被自动保存到相应的文件中，以供后续展示。

* 1. **游戏记录的存储和展示**

**游戏记录存储**

每一次游戏结束后，当局游戏的游玩记录（游戏时间，通关状态，游戏耗时）将被自动记录到文件内。

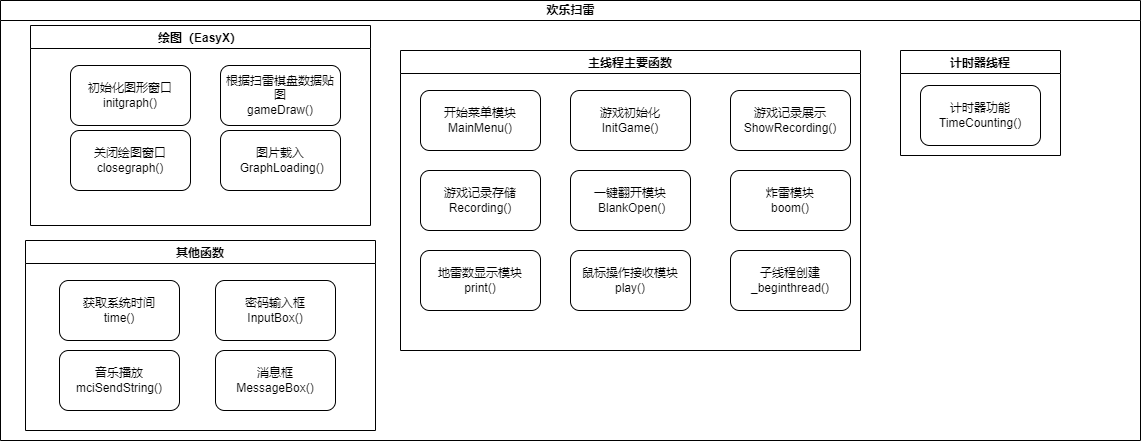
**游戏记录展示**

当玩家从主界面进入游戏后，游戏界面右侧将展示出最近三次的游玩记录。

* 1. **音乐播放**

当玩家执行程序时，主界面将播放专属的BGM；进入游戏后，游戏界面也将播放专属于自己的BGM；当玩家点击到地雷时游戏失败 时，界面将播放地雷爆炸音效；当用户成功通关时，界面将播放胜利音效。

**3. 总体设计**



## 主控模块设计

int main() /\*主控函数\*/

创建计时器线程，初始化绘图界面，调用执行开始菜单操作函数，加载所需的图片文件，初始化游戏，游戏结束后关闭图形窗口。

* 1. **主要模块设计**

1. **开始菜单模块**

/\*进入游戏开始菜单界面\*/

void MainMenu(HWND window)

播放开始菜单背景音乐，提供四个游戏模式，包含按钮悬停特效，点击不同的按钮进入不同的游戏模式。

1. **计时器模块**

/\*计时器\*/

void TimeCounting(void \*none)

记录当前游戏时间，并将耗时实时输出到游戏界面右侧，游戏结束时将游戏耗时写入全局变量timer，交给相应模块进行保存。

1. **游戏记录存储模块**

/\*游戏记录存储\*/

void Recording(int pass)

游戏结束后调用，将精确到秒的日历时间，本次游戏通关状态，游戏用时存储到文件。

1. **游戏记录展示模块**

/\*游戏记录展示\*/

void ShowRecording()

每次开始游戏时，将存储到文件的最近三次游戏记录输出到游戏界面右侧。

1. **游戏贴图载入模块**

/\*游戏贴图载入\*/

void GraphLoading()

将进行游戏时每个方格需要的贴图的编号顺序载入。

1. **游戏初始化模块**

/\*游戏初始化\*/

int InitGame()

初始化游戏数据，在扫雷棋盘内随机生成地雷，将棋盘进行加密。

1. **贴图模块**

/\*对棋盘进行贴图\*/

void gameDraw()

根据每个扫雷棋盘格子的状态载入相应的贴图。

1. **无雷区域一键翻开模块**

/\*无雷区域一键翻开\*/

void BlankOpen(int r, int c)

当左键点击到无雷格子时，通过递归将相邻的无雷格子的状态全部切换为翻开。

1. **炸雷模块**

void boom() /\*将地雷状态切换为爆炸\*/

当左键点击到无雷格子时，将所有现存地雷的状态设置为爆炸。

1. **地雷数展示模块**

int print() /\*展示现存地雷数\*/

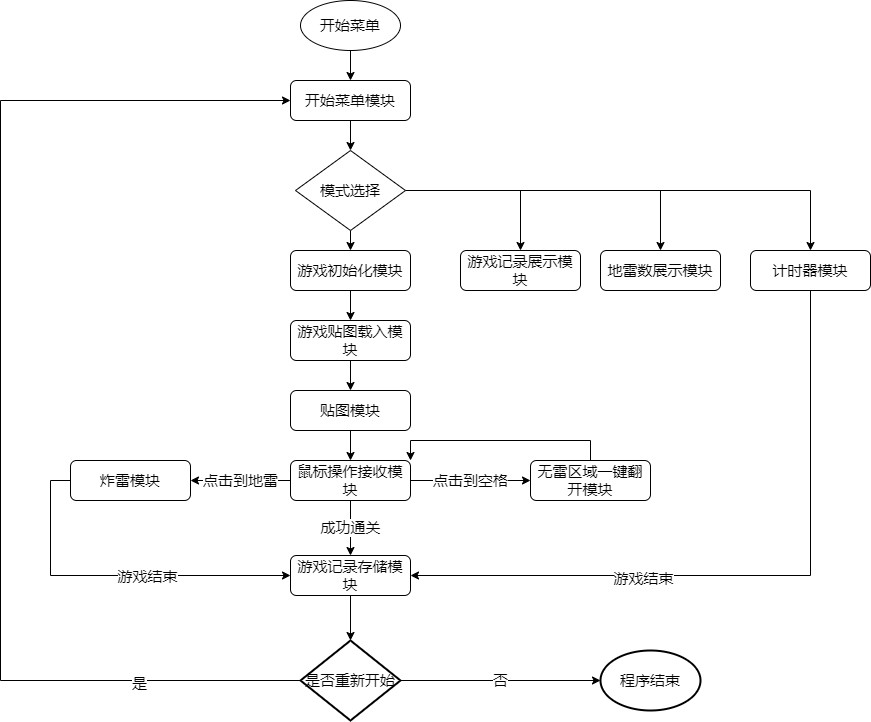
将现存的地雷数展示到游戏界面右侧。

1. **鼠标操作接收模块**

int play() /\*接收用户鼠标操作\*/

接收用户左键翻开格子，右键标记和取消标记等鼠标操作的信息。

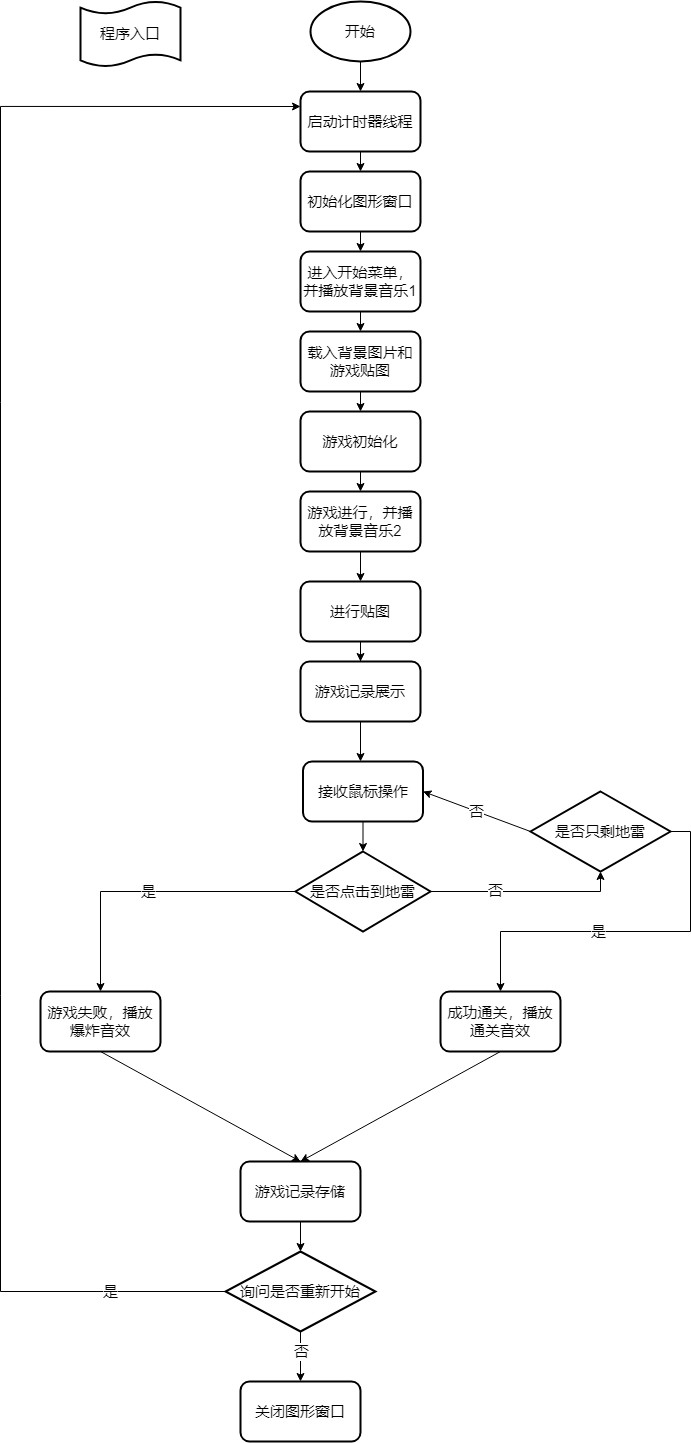
* 1. **模块间调用关系设计**



**4. 详细设计**

## 关键数据结构设计

1. 贴图所使用的图片： EasyX 中的图片类IMAGE 。
2. 扫雷格子不同位置和不同状态的模拟： int 型的二维数组 map 。
3. 点开格子数量与每局生成地雷数还有计时秒数： int 型的变量 count 和 mineNum 还有 timer 。
4. 控制计时开始与特殊模式上帝模式开始： bool 型的变量 godMode 和 timingStart 。
5. 存储计时器记录的时间字符串： char 类型的一维数组 timeRecord 。
   1. **主控模块main()流程图**

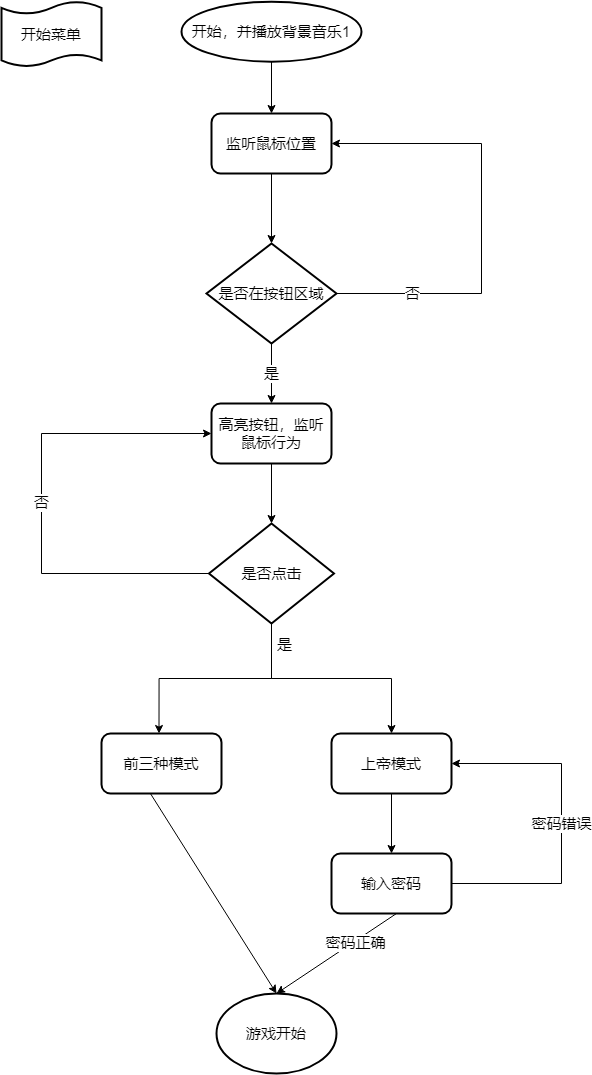


## 主要模块详细设计

### 开始菜单模块

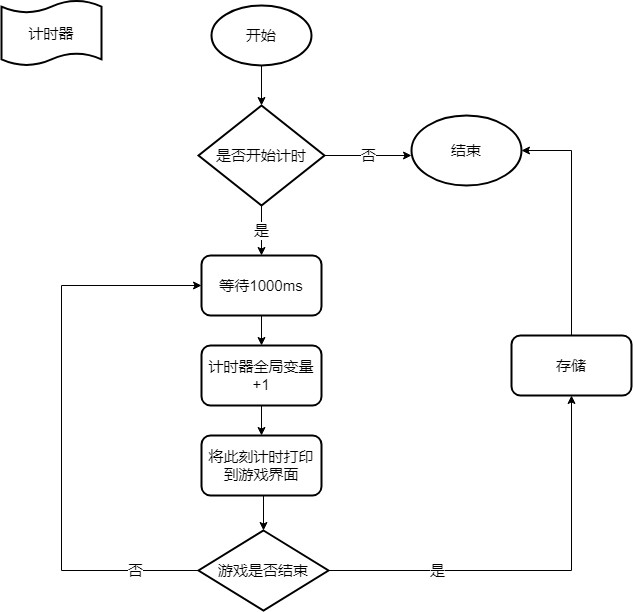
MainMenu(HWND window)

**实现流程**



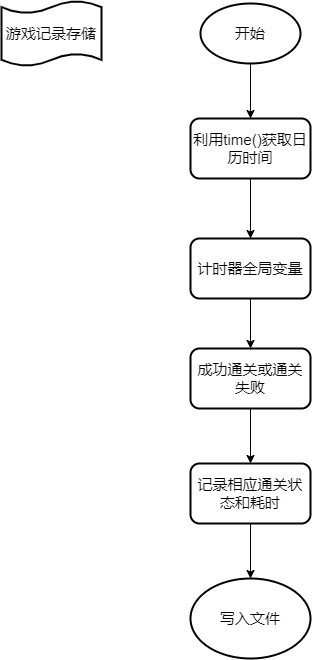
1. **计时器模块 实现流程**

TimeCounting(void \*none)



1. **游戏记录存储模块 实现流程**

Recording(int pass)



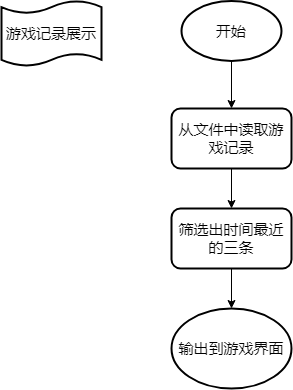
1. **游戏记录展示模块**

ShowRecording()

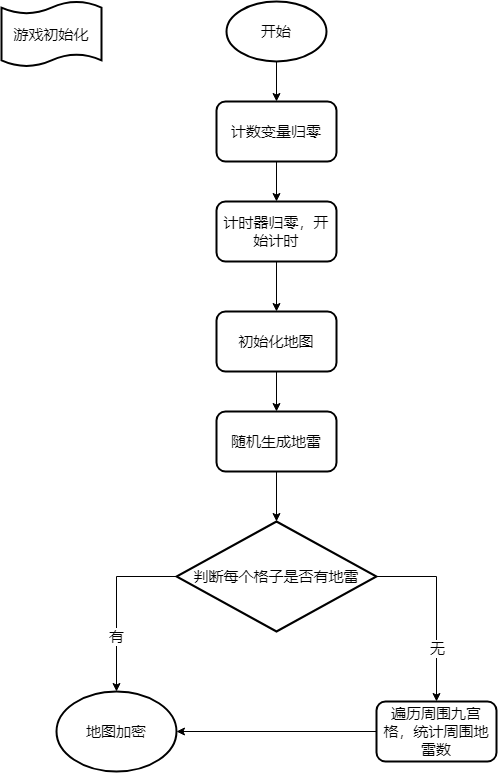
1. **游戏初始化模块**

InitGame()

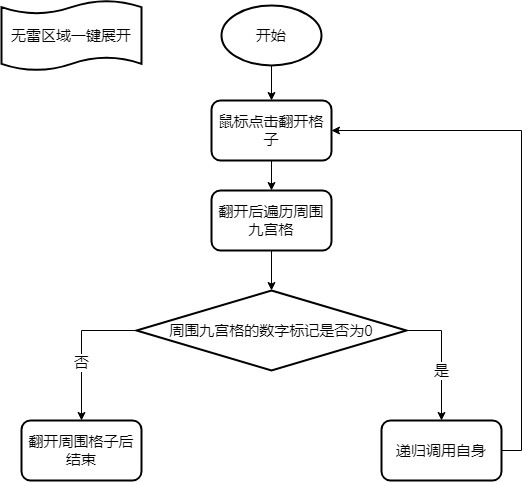
**实现流程**



**实现流程**

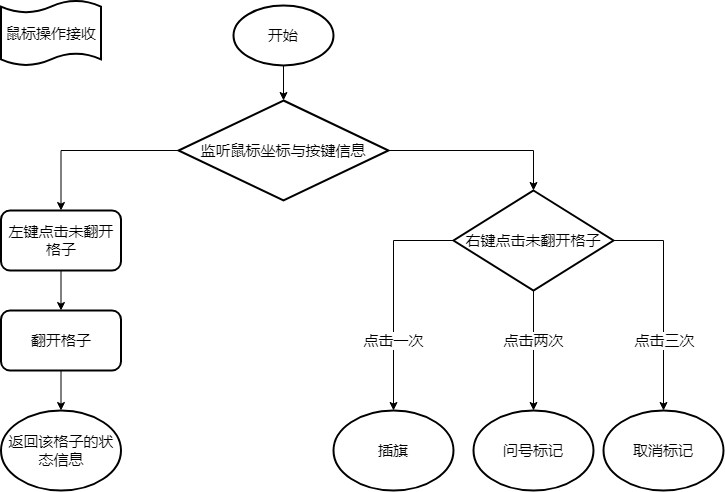


BlankOpen(int r, int c)



**(7)鼠标操作接收模块 实现流程**

play()



**5.系统开发**

## 软件环境：

### 开发环境

本次开发在Visual Studio 2022下进行。

1. **第三方库**

在进行游戏的图形化时使用到了第三方图形库 EasyX。

1. **Windows规格**

版本 Windows 10 家庭中文版版本号 21H2

安装日期 2021/8/2

操作系统内部版本 19044.1766

体验 Windows Feature Experience Pack 120.2212.4180.0

**硬件环境：**

设备名称LAPTOP-BT295J44

处理器 AMD Ryzen 7 5800U with Radeon Graphics 1.90 GHz

机带 RAM 16.0 GB (15.4 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

* 1. **主功能界面实现及代码**



**核心代码：**

void MainMenu(HWND window)

{

mciSendString("open ./BGM.MP3 alias BGM", NULL, 0, NULL); // 向多媒体设备接口(mci)发送(send)一个字符串(string) mciSendString("play BGM repeat", NULL, 0, NULL); // 播放音乐

BeginBatchDraw(); settextcolor(BLACK); setbkmode(TRANSPARENT);

loadimage(&backgroud, "./background.png", COL \* SIZE + 220, ROW \* SIZE);

putimage(0, 0, &backgroud); //加载图片

fillrectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6 + 30);

fillrectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 50, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6 + 80);

fillrectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 100, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6 + 130);

fillrectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 150, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6 + 180);

outtextxy((COL \* SIZE + 200) / 2 - 100 + 65, SIZE \* 6 + 7, "简单模式");

outtextxy((COL \* SIZE + 200) / 2 - 100 + 65, SIZE \* 6 + 57, "普通模式");

outtextxy((COL \* SIZE + 200) / 2 - 100 + 65, SIZE \* 6 + 107, "困难模式");

outtextxy((COL \* SIZE + 200) / 2 - 100 + 65, SIZE \* 6 + 157, "上帝模式"); EndBatchDraw();

ExMessage msg; int flag = 1; while (flag)

{

again: //密码输入错误

if (peekmessage(&msg, EM\_MOUSE))

{

//第一个按键

if (msg.x > (COL \* SIZE + 200) / 2 - 100 && msg.x < (COL \* SIZE + 200) / 2 + 100 && msg.y > SIZE \*

6 && msg.y < SIZE \* 6 + 30) //第一个按键

{

30);

BeginBatchDraw(); // 悬停特效

setlinecolor(BLACK);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6 +

EndBatchDraw();

switch (msg.message)

{

case WM\_LBUTTONDOWN:

flag = 0;

mineNum = 10; break;

}

}

else //移开清除悬停特效

{

30);

BeginBatchDraw(); setlinecolor(WHITE);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6 +

EndBatchDraw();

}

//第二个按键

if (msg.x > (COL \* SIZE + 200) / 2 - 100 && msg.x < (COL \* SIZE + 200) / 2 + 100 && msg.y > SIZE \* 6 + 50 && msg.y < SIZE \* 6 + 80) //第二个按键

{

+ 80);

BeginBatchDraw(); setlinecolor(BLACK);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 50, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6

EndBatchDraw();

switch (msg.message)

{

case WM\_LBUTTONDOWN:

flag = 0;

mineNum = 30; break;

}

}

else //移开清除悬停特效

{

+ 80);

BeginBatchDraw(); setlinecolor(WHITE);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 50, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6

EndBatchDraw();

}

if (msg.x > (COL \* SIZE + 200) / 2 - 100 && msg.x < (COL \* SIZE + 200) / 2 + 100 && msg.y > SIZE \* 6 + 100 && msg.y < SIZE \* 6 + 130) //第三个按键

{

+ 130);

BeginBatchDraw(); setlinecolor(BLACK);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 100, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6

EndBatchDraw();

switch (msg.message)

{

case WM\_LBUTTONDOWN:

flag = 0;

mineNum = 50;

break;

}

}

else //移开清除悬停特效

{

+ 130);

BeginBatchDraw(); setlinecolor(WHITE);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 100, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6

EndBatchDraw();

}

if (msg.x > (COL \* SIZE + 200) / 2 - 100 && msg.x < (COL \* SIZE + 200) / 2 + 100 && msg.y > SIZE \* 6 + 150 && msg.y < SIZE \* 6 + 180) //第四个按键

{

+ 180);

BeginBatchDraw(); setlinecolor(BLACK);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 150, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6

EndBatchDraw();

switch (msg.message)

{

case WM\_LBUTTONDOWN:

char passWord[50];

char rightWord[50] = "222021321102077"; InputBox(passWord, 50, "请输入激活密码"); if (strcmp(passWord, rightWord) == 0)

{

flag = 0;

mineNum = 30;

godMode = 1; break;

}

else

{

MessageBox(window, "密码错误！", "", MB\_OK); goto again;

}

}

}

else //移开清除悬停特效

{

+ 180);

BeginBatchDraw(); setlinecolor(WHITE);

rectangle((COL \* SIZE + 200) / 2 - 100, SIZE \* 6 + 150, (SIZE \* COL + 200) / 2 + 100, SIZE \* 6

EndBatchDraw();

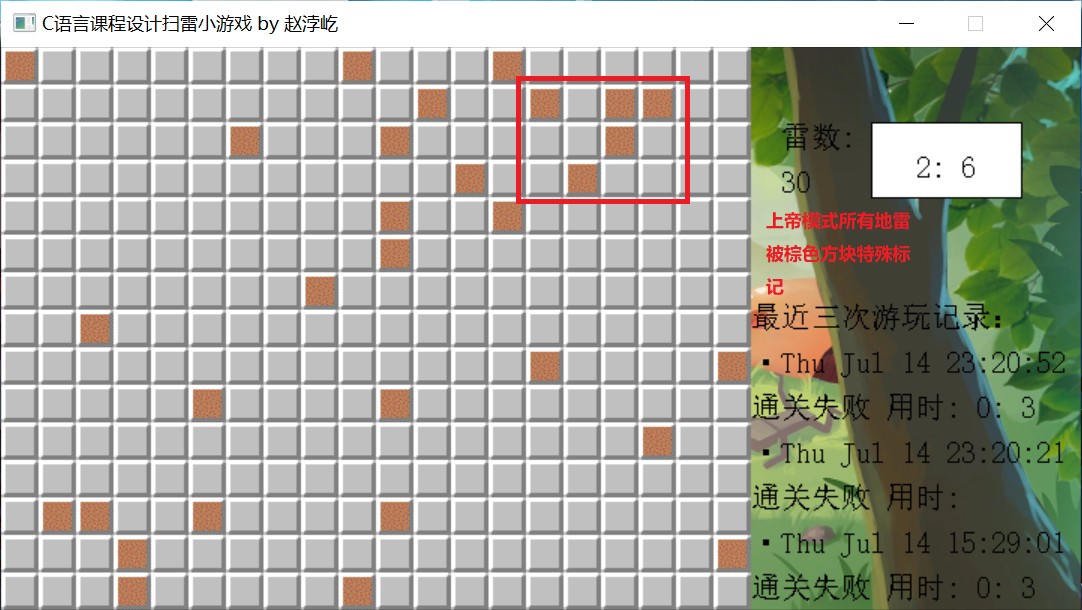
}

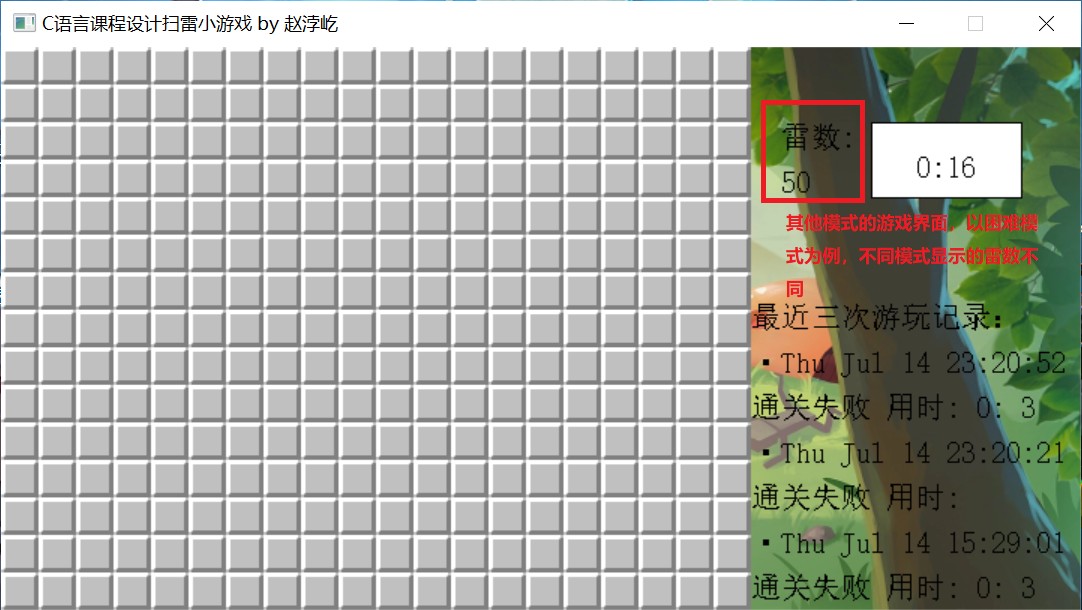
}

}

}

## 游戏界面实现及代码







**核心代码：**

int main()

{

\_beginthread(TimeCounting, 0, NULL); restart:

timingStart = 0;// 计时关闭，防止在开始菜单显示计时

HWND window = initgraph(COL \* SIZE + 220, ROW \* SIZE); SetWindowText(window, "C语言课程设计扫雷小游戏 by 赵浡屹"); MainMenu(window);

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

mciSendString("open ./BGM1.MP3 alias BGM", NULL, 0, NULL); //向多媒体设备接口(mci)发送(send)一个字符串(string) mciSendString("play BGM repeat", NULL, 0, NULL);

putimage(0, 0, &backgroud); //重新载入背景图GraphLoading();

InitGame(); //生成雷，初始化，加密while (1)

{

gameDraw(); //执行一次play后重复进入循环反复贴图print(); //打印出当前剩余雷的数量ShowRecording();

if (play() == -1) //play函数自身会一直运行接收点击格子的信息，点击到格子后才返回

{

boom();

gameDraw(); //踩到雷后全解密重新贴图展示所有雷mciSendString("close BGM ", 0, 0, 0);

mciSendString("open ./爆炸音效.wav alias BGM", 0, 0, 0);

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

Recording(0); //通关失败并记录

int is\_ok = MessageBox(window, "踩到雷啦，游戏结束!\n别灰心啦，要再来一把试试看？", "", MB\_OKCANCEL); if (is\_ok == IDOK)

{

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

godMode = 0; goto restart;

}

break;

}

else if (ROW \* COL - mineNum == count) //找到了所有雷或者标记了所有雷

{

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

mciSendString("open ./通关音效.wav alias BGM", 0, 0, 0);

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

Recording(1); //通关成功并记录

int is\_ok = MessageBox(window, "恭喜你一命通关！\n这么厉害要不要再来一把呀？", "", MB\_OKCANCEL); if (is\_ok == IDOK)

{

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

godMode = 0; goto restart;

}

break;

}

}

closegraph();

return 0;

}

//鼠标点击开玩int play()

{

ExMessage msg; int r = 0, c = 0; while (1)

{

if (peekmessage(&msg, EM\_MOUSE))

{

switch (msg.message)

{

case WM\_LBUTTONDOWN:

r = msg.y / SIZE + 1; c = msg.x / SIZE + 1;

if (r >= 1 && r <= ROW && c >= 1 && c <= COL) //保证点击的是游戏区内的格子

{

if (map[r][c] >= 19 && map[r][c] <= 28) // 对未点开的格子(非红旗和问号标记)

{

if (map[r][c] == 20) // 点击到了无雷区空格

{

BlankOpen(r, c); // 周围展开

return map[r][c]; // 返回非-1值

}

else

{

map[r][c] -= 20; count++;

return map[r][c]; // 有雷区返回-1

}

}

if (map[r][c] >= 59 && map[r][c] <= 68) // 对问号标记的格子

{

if (map[r][c] == 60)

{

BlankOpen(r, c); return map[r][c];

}

else

{

map[r][c] -= 60; count++;

return map[r][c];

}

}

}

break;

//插旗子，拔旗子case WM\_RBUTTONDOWN:

r = msg.y / SIZE + 1; c = msg.x / SIZE + 1;

if (r >= 1 && r <= ROW && c >= 1 && c <= COL) //保证点击的是游戏区内的格子

{

if (map[r][c] >= 19 && map[r][c] <= 28) //第一次点击 插旗子

{

map[r][c] += 20;

}

else if (map[r][c] >= 39 && map[r][c] <= 48) //第二次点击 变问号

{

map[r][c] += 20;

}

else if (map[r][c] >= 59 && map[r][c] <= 68) //第三次点击 变回未标记状态

{

map[r][c] -= 40;

}

return map[r][c];

}

break;

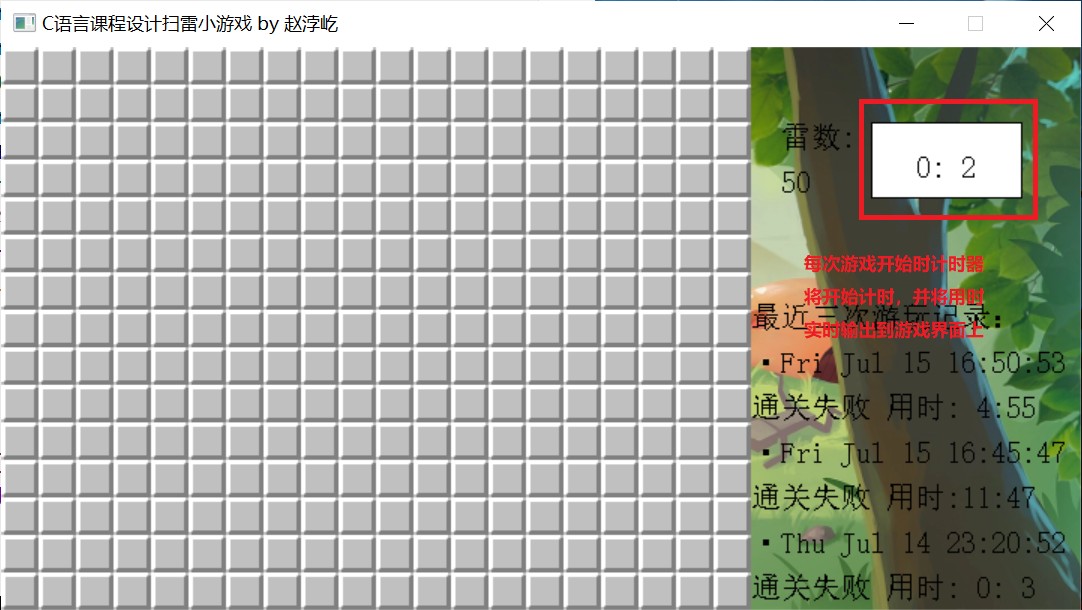
}

}

}

}

## 计时器模块实现及代码



**核心代码：**

}

}

}

//settextstyle(&f);// 设置字体样式

settextcolor(BLACK);

sprintf(timeRecord, "%2d:%2d", timer / 60, timer % 60); //将时间格式化为字符串fillrectangle(580, 50, 580 + 100, 100);

setlinecolor(BLACK); rectangle(580, 50, 580 + 100, 100);

settextstyle(20, 0, "宋体");

outtextxy(600, 70, timeRecord); Sleep(1000); // 1000ms后计时器自增

timer++;

// 开始计时的判定

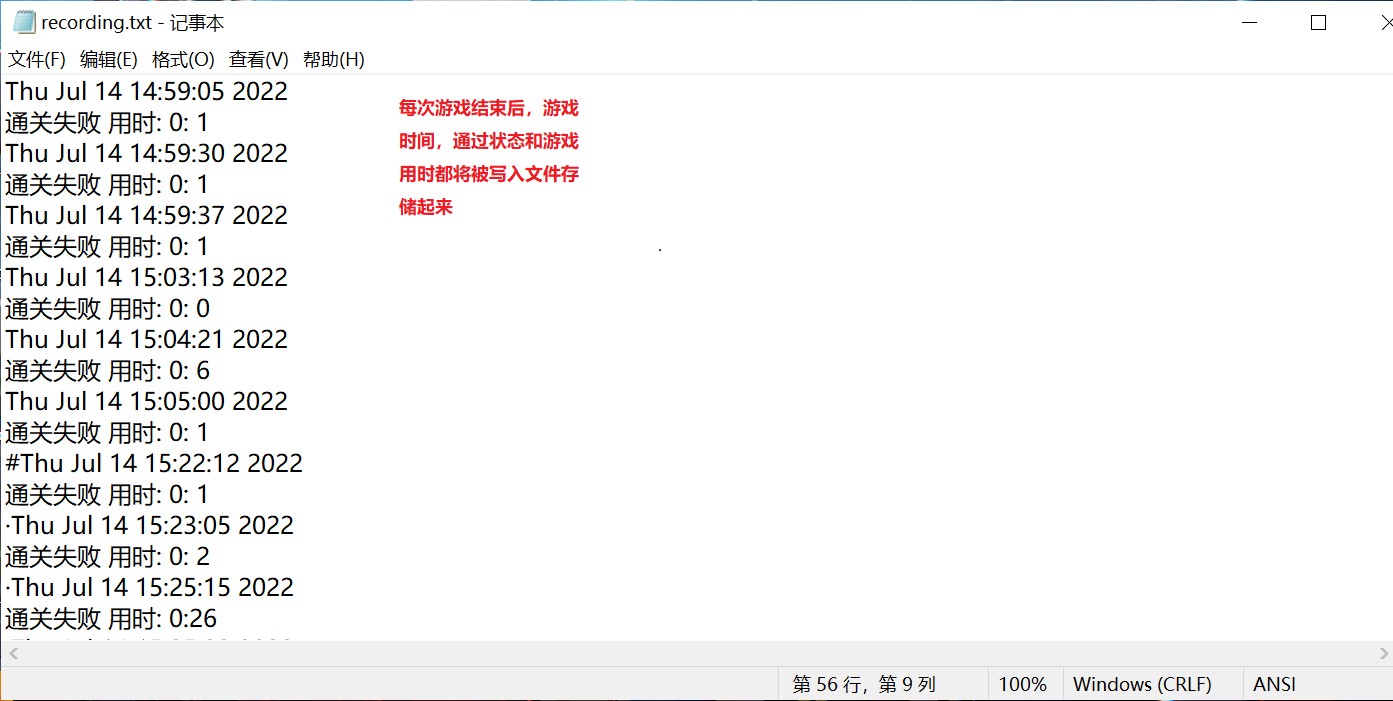
void TimeCounting(void\* none)

{

while (1) {

if (timingStart) {

* 1. **游戏记录存储模块实现及代码**



**核心代码：**

void Recording(int pass) //记录游玩时间，是否通关,游戏耗时，写入文件

的地方倒序输出

{

FILE\* fp;

fp = fopen("./recording.txt", "a"); time\_t endingTime = time(NULL); //日历时间

//记住每次运行程序显示 可读入数组然后判断第一行不为0

fputs("·", fp);

fputs(ctime(&endingTime), fp); //将日历时间转换为字符串写入文件if (pass)

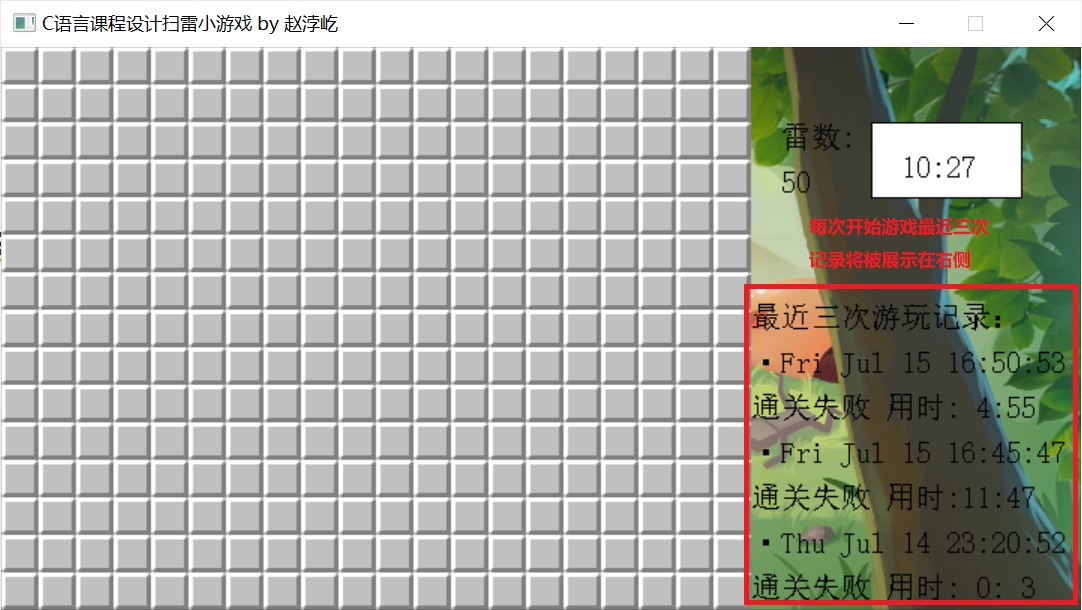
fprintf(fp, "成功通关 用时:%s\n", timeRecord); //写入通关状态换行符，因为fputs不写入换行符else

fprintf(fp, "通关失败 用时:%s\n", timeRecord); //写入通关状态换行符，因为fputs不写入换行符

fclose(fp);

}

* 1. **游戏记录展示模块实现及代码**



**核心代码：**

void ShowRecording()

{

char text[1000][55] = { 0 }; FILE\* fp;

fp = fopen("./recording.txt", "r");

setbkcolor(TRANSPARENT); outtextxy(500, 170, "最近三次游玩记录："); int i;

for (i = 0; i < 1000; i++)

{

fgets(text[i], 50, fp);

}

for (i = 0; i < 1000; i++)

{

if (text[i][0] == 0) break;

}

/\*int j = i - 1; printf("%s",text[j]);\*/ int k = 0;

for (int j = i - 1; j >= i - 6; j = j - 2)

{

outtextxy(500, 200 + 60 \* k, text[j - 1]); outtextxy(500, 230 + 60 \* k, text[j]); k++;

}

fclose(fp);

}

* 1. **无雷区域一键展开模块实现及代码**



**核心代码：**

//当左键点击未点击过的格子时 连续展开

//如果鼠标信息知道了我点击的格子是一个空白格子，那么将他翻开后开始对周围的八个格子进行遍历，如果遍历到空白格子那么就进行下一次遍历void BlankOpen(int r, int c)

{

//打开格子

if (map[r][c] >= 59 && map[r][c] <= 68) //当点开问号标记区域时候

{

map[r][c] -= 60;

}

else

{

map[r][c] -= 20;

}

count++;

//点开后遍历九宫格

for (int m = r - 1; m <= r + 1; m++)

{

for (int n = c - 1; n <= c + 1; n++)

{

if (m >= 1 && m <= ROW && n >= 1 && n <= COL)

{

if (map[m][n] >= 19 && map[m][n] <= 28)

{

//保证是游戏区

//必须为空白格

if (map[m][n] != 20)

{

map[m][n] -= 20; count++;

}

else

{

BlankOpen(m, n);

}

// 周围格子数字标记不为0，展开后递归结束

// 周围格子数字标记为0的话重复调用自身

}

if (map[m][n] >= 59 && map[m][n] <= 68)

{

if (map[m][n] != 60)

{

map[m][n] -= 40; count++;

}

// 周围格子数字标记不为0，展开后递归结束

else // 周围格子数字标记为0的话重复调用自身

{

BlankOpen(m, n);

}

}

}

}

}

}

**6. 测试**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试阶段 | | 〇单元测试 ●集成测试 〇系统测试 〇用户现场测试 | | | | |
| 用例ID | 用例名称 | 测试目的 | 输入描述 | 预期结果 | 实际结果 | 测试数据 |
|  |  |  | 1.点击“简单模式” |  | 成功 |  |
| 001 | 开始菜单功能 | 确保开始菜单功能正常 | 1. 点击“普通模式” 2. 点击“困难模式” 3. 点击“上帝模式”， 并输入密码 | 鼠标悬停时按钮高亮，点击按钮能进入相应模式 | 鼠标点击按钮键盘输入密码 |
| 002 | 左键扫雷功能 | 保证扫雷功能正常 | 左键点击格子 | 点击到格子后格子被翻开，无雷游戏继续，有地雷则游戏结束 | 成功 | 左键点击格子 |
| 003 | 无雷区域一键展开功能 | 保证一键展开功能正常 | 左键点击格子 | 点击到无雷格子后，所有相邻且连续的无雷格子都被翻开 | 成功 | 左键点击格子 |
| 004 | 计时器功能 | 保证计时器功能正常 | 无 | 游戏用时被实时输出到游戏界面右侧 | 成功 | 无 |
| 005 | 游戏记录展示功能 | 保证游戏记录展示功能正常 | 主界面选择任一模式进入游戏 | 游戏界面右侧展示出最近三次游戏记录 | 成功 | 无 |
| 006 | 右键标记功能 | 确保右键标记功能正常 | 右键点击格子一次  右键点击格子两次  右键点击格子三次 | 第一次点击格子被旗子标记，第二次点击格子被问号标记，  第三次点击空格标记取消 | 成功 | 右键点击格子 |
| 007 | 游戏记录存储功能 | 确保游戏记录存储功能正确 | 游戏结束后寻找项目中生成的  “recording.txt”文件 | 文件中记录了之前的游戏记录 | 成功 | “recording.txt”  文件 |
| 008 | 重新开始游戏功能 | 确保重新开始游戏功能正常 | 游戏结束后点击弹出消息框的“确定”按钮 | 点击确定按钮后游戏重新开始 | 成功 | 鼠标点击按钮 |
| 009 | 音乐播放功能 | 确保音乐播放功能正常 | 按通关失败，通关成功两种流程完成游戏 | 在开始菜单界面，播放背景音乐1，在游戏开始时，播放背景音乐2，点击到地雷播放爆炸音 效，游戏通关播放通关音效 | 成功 | 无 |

**7. 问题与解决方案**

## 扫雷游戏中如何实现让用户只靠鼠标点击图形化界面完成游戏操作？

我们使用一个整型二维数组模拟不同状态的方格，再通过图形库 Easyx 接收鼠标信息，当用户点击图形时，对二维数组进行修改，从而改变方格的状态，通过判断方格的状态调用不同的函数模块，从而使用户只靠鼠标在图形化界面下游玩游戏。

1. **用二维数组模拟不同状态的方格，如何让方格在点击前看上去一样？**

我们可以给二维数组存储的每个数据加上同一个整数，完成加密，对该范围内的数据统一贴图。当格子被翻开时，对存储的数据减去该 整数，完成解密，然后根据相应的数据元素的值执行相应操作。

1. **如何实现经典扫雷中的点击到无雷格子一键翻开周围连续无雷区域的功能？**

我们通过编写一个递归函数，当点击到无雷的格子时，函数将遍历以该格子为中心的九宫格，然后将遍历到的连续无雷的格子状态全部 设置为翻开，当遍历到有雷格子时函数结束，从而实现经典扫雷中的点击到无雷格子一键翻开周围连续无雷区域的功能。

1. **如何实现按钮悬停特效功能?**

对于白色按钮而言，我们通过 EasyX 提供的相应函数监听鼠标信息，当鼠标移动到按钮区域时，我们通过调用 rectangle() 函数在按钮边缘画黑色长方体，当鼠标移出按钮区域时，我们通过 rectangle() 函数在按钮边缘画白色长方体覆盖黑色长方体，从而实现按钮悬停的特效。

## 如何实现音乐播放功能？

我们通过包含 mmsystem.h 多媒体设备接口头文件，然后通过 #pragma comment(lib,"winmm.lib") 加载静态库，最后通过调用

mciSendString () 向多媒体接口(mci)发送(send)一个字符串(string)，播放指定路径下的音乐。

## 游戏进行时程序一直在等待接收鼠标的信息，如何实现计时器实时计时并显示的功能？

单线程程序在一个时间只能做一件事情，我们通过 process.h 库为计时器单独创建一个线程，让其在游戏主线程运行时计时器能够实时计时并输出到游戏界面上。

1. **计时器线程如何将时间传递给游戏记录存储模块？**

我们通过设置一个记录时间的全局变量 timer 在计时时自增，游戏结束时再让存储模块使用全局变量的值进行处理还原为需要的时间格式进而完成信息交流，并将其存储到文件中。

1. **为什么使用EasyX库中的outtextxy()等函数时VS提示没有与参数列表匹配的重载函数，无法通过编译？**

字符串常见的编码有两种：MBCS 和 Unicode。VC6 新建的项目默认为 MBCS 编码，VC2008 及高版本的 VC 默认为 Unicode 编码。为了解决问题可在每个传入函数的字符串前加一个大写的"L"。或者点击：项目属性->高级->字符集，将Unicode修改为多字节字符集。

1. **为什么在 Visual Studio 中一些常用的函数会无法通过编译，VS 会提示你使用另外的函数代替？**

scanf() 、 gets() 、 fgets() 、 strcpy() 、 strcat() 等都是 C 语言自带的函数，它们都是标准函数，但是它们都有一个缺陷，就是不安全，可能会导致数组溢出或者缓冲区溢出，让黑客有可乘之机，从而发起“缓冲区溢出”攻击。 scanf\_s() 、 gets\_s() 、

fgets\_s() 、 strcpy\_s() 、 strcat\_s() 是微软自己发明的安全函数，它们仅适用于 VS，在其它编译器下无效。这些安全函数在读取或

操作字符串时要求指明长度，这样一来，过多的字符就会被过滤掉，避免了数组或者缓冲区溢出。

**8. 参考文献**

1. EasyX 官方文档<https://docs.easyx.cn/zh-cn>
2. EasyX基础入门——这一篇就够啦<http://t.csdn.cn/XgTXg>
3. C语言课程设计案例精编(第3版) [M].清华大学出版社,2016.5.1