**综合实践（实现扫雷游戏程序）**

**一．成员和分工如下：**

2024015004吐尔洪·努尔买买提-1，2024013318龚超-2， 2024013165张曦-3，2024015002刘梓桐-4，2024015003袁茗清-5。

1.菜单，设置雷，整理Word，

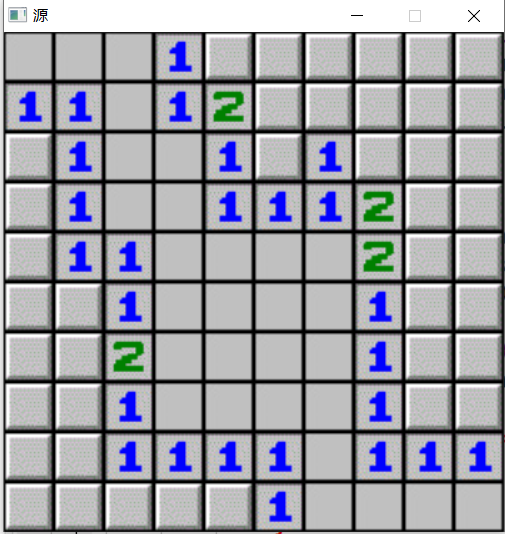
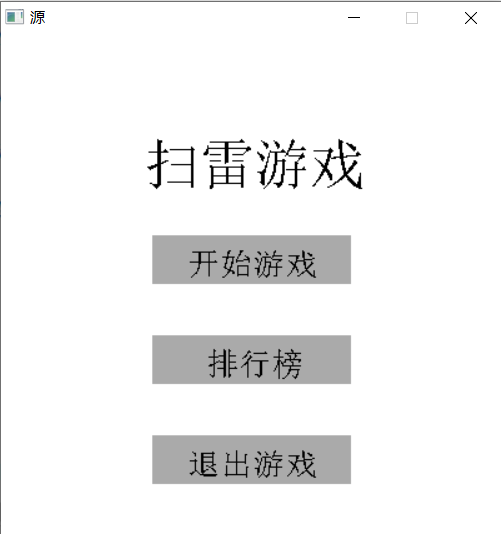
2.雷区状况，贴图，

3.点击打开格子，标记功能，

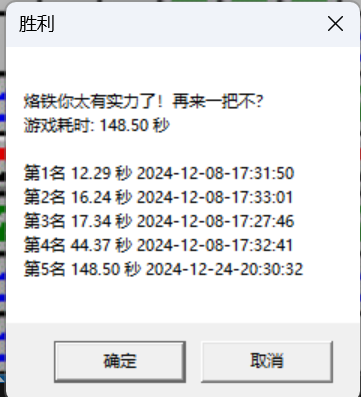
4.输赢判断，反复执行，

5.计时，排行。

**二．运行图**



（2-1菜单选项） （2-2游戏页面）



（2-3赢输）



（2-4排行榜）

**三，运行代码以及各个功能**

下面是各个函数的功能描述：

1. init(int map[][COL])：初始化雷区。这个函数首先加载资源，然后将地图数组`map`的所有元素初始化为0。接着随机放置`MineNum`个雷（-1表示雷），并更新周围格子的数字（表示周围雷的数量）。最后，对格子进行加密（加上20）。

2. show(int map[][COL])：雷区数据显示。这个函数用于在控制台打印出当前雷区的状态，每个格子的数字表示周围雷的数量。

3. loadResource()：加载图。这个函数用于加载游戏所需的图像资源，并将它们存储在`imgs`数组中。

4. draw(int map[][COL])：绘制贴图。根据`map`数组中的数据，将对应的图像绘制到图形界面上。

5. mouseMsg(ExMessage\* msg, int map[][COL])：鼠标操作数据。处理鼠标事件，左键用于打开格子，右键用于标记雷。

6. boomBlank(int map[][COL], int row, int col)：点击空白格子连环爆开周围所有空白格子还有数字。如果点击的格子是空白（0），则递归地打开周围所有未打开的格子。

7. judge(int map[][COL], int row, int col)：判断输赢。如果点击到了雷，则返回-1表示输了；如果所有非雷格子都被成功打开，则返回1表示赢了；否则返回0表示游戏继续。

8. startTimer()：开始计时。记录游戏开始的时间。

9. stopTimer()：停止计时并计算时间差。记录游戏结束的时间，并计算游戏耗时。

10. displayTime()：返回游戏耗时字符串。将游戏耗时格式化为字符串返回。

11. getRankString()：获取排行榜字符串。将排行榜信息格式化为字符串返回。

12. updateRank(double time)：更新排行榜。将新的成绩插入到排行榜中，并保存到文件。

13. saveRankings()：保存排行榜到文件。将当前的排行榜数据写入到文件中。

14. loadRankings()：从文件加载排行榜。从文件中读取排行榜数据。

15. drawMenu()：绘制菜单。绘制游戏的主菜单界面。

16. handleMenuClick(int x, int y)：处理菜单点击事件。根据鼠标点击的位置，返回相应的操作码。

17. drawRanking()：绘制排行榜。绘制排行榜界面，并显示当前的排行榜数据。

18. handleRankingClick(int x, int y)：处理排行榜点击事件。处理排行榜界面的点击事件，返回相应的操作码。

main函数是程序的入口点，它初始化图形界面，加载排行榜，显示菜单，并进入主循环处理鼠标事件和更新游戏状态。游戏的主循环中，会根据鼠标事件更新雷区状态，判断游戏是否结束，并在适当的时候更新排行榜和显示游戏结果。

**四，运行代码**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <easyx.h>

#include <windows.h> // 包含Windows特定的头文件

#define ROW 10 // 定义行列的常量

#define COL 10

#define MineNum 10 // 雷的数量

#define ImgSize 40

#define TOP\_RANKS 10 // 排行榜数量

IMAGE imgs[12];

char title[50];

clock\_t startTime, endTime; // 计时变量

struct RankEntry {

double time;

char date[30];

};

RankEntry topRanks[TOP\_RANKS]; // 存储前十名的时间和日期

bool gameOver = false; // 标志变量，表示游戏是否已经结束

void init(int map[][COL]); // 初始化雷区

void show(int map[][COL]); // 雷区数据显示

void loadResource(); // 加载图

void draw(int map[][COL]); // 贴图

void mouseMsg(ExMessage\* msg, int map[][COL]); // 传鼠标坐标 标记 点开

void boomBlank(int map[][COL], int row, int col); // 炸开空白地区

int judge(int map[][COL], int row, int col); // 判断输赢

void startTimer(); // 开始计时

void stopTimer(); // 停止计时并计算时间差

char\* displayTime(); // 返回游戏耗时字符串

char\* getRankString(); // 获取排行榜字符串

void updateRank(double time); // 更新排行榜

void saveRankings(); // 保存排行榜到文件

void loadRankings(); // 从文件加载排行榜

void drawMenu(); // 绘制菜单

int handleMenuClick(int x, int y); // 处理菜单点击事件

void drawRanking(); // 绘制排行榜

int handleRankingClick(int x, int y); // 处理排行榜点击事件

// 初始化雷区

void init(int map[][COL]) {

loadResource();

// 把map全部初始化为0

memset(map, 0, sizeof(int) \* ROW \* COL);

for (int i = 0; i < MineNum;) {

int r = rand() % ROW;

int c = rand() % COL;

if (map[r][c] == 0) {

map[r][c] = -1;

i++;

}

}

// 把雷形成的九宫格除雷以外都加1；

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

if (map[i][j] == -1) {

for (int r = i - 1; r <= i + 1; r++) {

for (int c = j - 1; c <= j + 1; c++) {

if ((r >= 0 && r < ROW && c >= 0 && c < COL) // 没越界

&& map[r][c] != -1)

++map[r][c];

}

}

}

}

}

// 加密格子

for (int i = 0; i < ROW; i++) {

for (int k = 0; k < COL; k++) {

map[i][k] += 20;

}

}

}

// 雷区数据显示

void show(int map[][COL]) {

for (int i = 0; i < ROW; i++) {

for (int k = 0; k < COL; k++) {

printf("%2d ", map[i][k]);

}

printf("\n");

}

}

// 加载图

void loadResource() {

for (int i = 0; i < 12; i++) {

char imgPath[50] = { 0 };

sprintf(imgPath, "./images/%d.jpg", i);

loadimage(&imgs[i], imgPath, ImgSize, ImgSize);

}

}

// 绘制 贴图

void draw(int map[][COL]) {

// 贴图

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

if (map[i][j] >= 0 && map[i][j] <= 8)

putimage(j \* ImgSize, i \* ImgSize, &imgs[map[i][j]]);

else if (map[i][j] == -1)

putimage(j \* ImgSize, i \* ImgSize, &imgs[9]);

else if (map[i][j] >= 19 && map[i][j] <= 28) {

putimage(j \* ImgSize, i \* ImgSize, &imgs[10]);

}

else if (map[i][j] >= 39) {

putimage(j \* ImgSize, i \* ImgSize, &imgs[11]);

}

}

}

}

// 鼠标操作数据

void mouseMsg(ExMessage\* msg, int map[][COL]) {

// 先根据鼠标点击的坐标求出对应的数组的下标

int r = msg->y / ImgSize;

int c = msg->x / ImgSize;

// 左键打开格子

if (msg->message == WM\_LBUTTONDOWN) {

// 什么时候点击有效能够打开（没有打开时可以打开）

if (map[r][c] >= 19 && map[r][c] <= 28) {

map[r][c] -= 20;

boomBlank(map, r, c); // 检测是否空白格子，是即炸开，不是即退出

if (startTime == 0) {

startTimer(); // 游戏开始时启动计时器

}

}

}

// 右键标记格子

else if (msg->message == WM\_RBUTTONDOWN) {

// 是否能够标记，如果没有打开就能够标记

if (map[r][c] >= 19 && map[r][c] <= 28) {

map[r][c] += 20;

}

else if (map[r][c] >= 39) {

map[r][c] -= 20;

}

}

}

// 点击空白格子连环爆开周围所有空白格子还有数字,row和col是当前点击的格子的下标

void boomBlank(int map[][COL], int row, int col) {

// 判断row col位置是不是空白格子

if (map[row][col] == 0) {

for (int r = row - 1; r <= row + 1; r++) {

for (int c = col - 1; c <= col + 1; c++) {

if ((r >= 0 && r < ROW && c >= 0 && c < COL) // 没越界

&& map[r][c] >= 19 && map[r][c] <= 28) // 没有打开

{

map[r][c] -= 20;

boomBlank(map, r, c);

}

}

}

}

}

// 游戏结束条件 输了返回-1，没结束返回0，赢了返回1

int judge(int map[][COL], int row, int col) {

// 点到了雷，结束 输了

if (map[row][col] == -1 || map[row][col] == 19) {

return -1;

}

// 点完了格子，结束 赢了 点开了90个格子

int cnt = 0;

for (int i = 0; i < ROW; i++) {

for (int k = 0; k < COL; k++) {

// 统计打开的格子的数量

if (map[i][k] >= 0 && map[i][k] <= 8) {

++cnt;

}

}

}

if (ROW \* COL - MineNum == cnt) {

return 1;

}

return 0;

}

// 开始计时

void startTimer() {

startTime = clock();

}

// 停止计时并计算时间差

void stopTimer() {

endTime = clock();

}

// 返回游戏耗时字符串

char\* displayTime() {

static char timeStr[50];

double duration = (double)(endTime - startTime) / CLOCKS\_PER\_SEC;

sprintf(timeStr, "游戏耗时: %.2f 秒\n", duration);

return timeStr;

}

// 获取排行榜字符串

char\* getRankString() {

static char rankStr[500];

strcpy(rankStr, "");

for (int i = 0; i < TOP\_RANKS; i++) {

if (topRanks[i].time != 0) {

char temp[60];

sprintf(temp, "第%d名 %.2f 秒 %s\n", i+1,topRanks[i].time, topRanks[i].date);

strcat(rankStr, temp);

}

}

return rankStr;

}

// 更新排行榜

// 更新排行榜

void updateRank(double ti) {

time\_t now = time(NULL);

struct tm\* tinfo;

tinfo = localtime(&now);

char formattedDate[30];

strftime(formattedDate, sizeof(formattedDate), "%Y-%m-%d-%H:%M:%S", tinfo);

// 找到新成绩应该插入的位置

int i;

for (i = 0; i < TOP\_RANKS; i++) {

if (topRanks[i].time == 0 || topRanks[i].time > ti) {

break;

}

}

// 如果找到了合适的位置，插入新成绩并移动后续元素

if (i < TOP\_RANKS) {

for (int j = TOP\_RANKS - 1; j > i; j--) {

topRanks[j] = topRanks[j - 1];

}

topRanks[i].time = ti;

strcpy(topRanks[i].date, formattedDate);

}

saveRankings(); // 更新后保存排行榜到文件

}

// 保存排行榜到文件

void saveRankings() {

FILE\* file = fopen("rankings.txt", "w");

if (file == NULL) {

MessageBox(GetHWnd(), "无法保存排行榜文件！", "错误", MB\_OK);

return;

}

for (int i = 0; i < TOP\_RANKS; i++) {

fprintf(file, "%.2f %s\n", topRanks[i].time, topRanks[i].date);

}

fclose(file);

}

// 从文件加载排行榜

void loadRankings() {

FILE\* file = fopen("rankings.txt", "r");

if (file == NULL) {

MessageBox(GetHWnd(), "无法加载排行榜文件，创建新排行榜！", "提示", MB\_OK);

memset(topRanks, 0, sizeof(topRanks));

return;

}

for (int i = 0; i < TOP\_RANKS; i++) {

fscanf(file, "%lf %s", &topRanks[i].time, topRanks[i].date);

}

fclose(file);

}

// 绘制菜单

void drawMenu() {

setbkcolor(WHITE);

cleardevice();

// 绘制标题

setbkmode(TRANSPARENT);

settextcolor(BLACK);

settextstyle(50, 0, \_T("Arial"));

outtextxy(ImgSize \* 3-5, ImgSize \* 2, \_T("扫雷游戏"));

// 绘制按钮背景

setfillcolor(LIGHTGRAY);

fillrectangle(ImgSize \* 3, ImgSize \* 4, ImgSize \* 7, ImgSize \* 5);

fillrectangle(ImgSize \* 3, ImgSize \* 6, ImgSize \* 7, ImgSize \* 7);

fillrectangle(ImgSize \* 3, ImgSize \* 8, ImgSize \* 7, ImgSize \* 9);

// 绘制按钮文本

settextcolor(BLACK);

settextstyle(30, 0, \_T("Arial"));

outtextxy(ImgSize \* 4-10, ImgSize \* 4 + 10, \_T("开始游戏"));

outtextxy(ImgSize \* 4+5, ImgSize \* 6 + 10, \_T("排行榜"));

outtextxy(ImgSize \* 4-10, ImgSize \* 8 + 10, \_T("退出游戏"));

}

// 处理菜单点击事件

int handleMenuClick(int x, int y) {

if (x >= ImgSize \* 3 && x <= ImgSize \* 7 && y >= ImgSize \* 4 && y <= ImgSize \* 5) {

return 1; // 开始游戏

}

else if (x >= ImgSize \* 3 && x <= ImgSize \* 7 && y >= ImgSize \* 6 && y <= ImgSize \* 7) {

return 2; // 显示排行榜

}

else if (x >= ImgSize \* 3 && x <= ImgSize \* 7 && y >= ImgSize \* 8 && y <= ImgSize \* 9) {

return 0; // 退出游戏

}

return -1; // 未点击按钮

}

// 绘制排行榜

void drawRanking() {

setbkcolor(WHITE);

cleardevice();

// 绘制标题

settextcolor(BLACK);

settextstyle(30, 0, \_T("Arial"));

outtextxy(ImgSize \* 5 - textwidth(\_T("排行榜")) / 2, ImgSize \* 1, \_T("排行榜"));

// 绘制排行榜内容

settextstyle(15, 0, \_T("Arial"));

int lineSpacing = 20; // 调行间距

for (int i = 0; i < TOP\_RANKS; i++) {

if (topRanks[i].time != 0) {

char rankStr[60];

sprintf(rankStr, "第%d名 %.2f 秒 %s", i + 1, topRanks[i].time, topRanks[i].date);

outtextxy(ImgSize \* 5 - textwidth(\_T(rankStr)) / 2, lineSpacing \* (i + 4), \_T(rankStr));

}

}

// 计算返回按钮的位置和大小，确保它居中并位于底部

int buttonWidth = ImgSize \* 2;

int buttonHeight = ImgSize \* 2;

int windowWidth = 10 \* ImgSize;

int windowHeight = 10 \* ImgSize;

int buttonX = (windowWidth - buttonWidth) / 2;

int buttonY = windowHeight - buttonHeight - ImgSize; // 留一些空间给底部边距

// 绘制返回按钮背景

setfillcolor(LIGHTGRAY);

fillrectangle(buttonX, buttonY, buttonX + buttonWidth, buttonY + buttonHeight);

// 绘制返回按钮文本

settextcolor(BLACK);

settextstyle(30, 0, \_T("Arial"));

// 计算文本位置，使其居中

int textX = buttonX + (buttonWidth - textwidth(\_T("返回"))) / 2;

int textY = buttonY + (buttonHeight - textheight(\_T("返回"))) / 2;

outtextxy(textX, textY, \_T("返回"));

}

// 处理排行榜点击事件

int handleRankingClick(int x, int y) {

// 计算返回按钮的位置和大小，确保它居中并位于底部

int buttonWidth = ImgSize \* 2;

int buttonHeight = ImgSize \* 2;

int windowWidth = 10 \* ImgSize;

int windowHeight = 10 \* ImgSize;

int buttonX = (windowWidth - buttonWidth) / 2;

int buttonY = windowHeight - buttonHeight - ImgSize; // 留一些空间给底部边距

if (x >= buttonX && x <= buttonX + buttonWidth && y >= buttonY && y <= buttonY + buttonHeight) {

return 1; // 返回菜单

}

return -1; // 未点击按钮

}

int main() {

initgraph(ImgSize \* 10, ImgSize \* 10);

srand((unsigned)time(NULL)); // 设置随机数种子

int map[ROW][COL] = { 0 };

loadRankings(); // 从文件加载排行榜

//菜单交互

bool inMenu = true;

bool inRanking = false;

drawMenu(); // 显示菜单

while (inMenu|| inRanking) {

if (inMenu) {

drawMenu(); // 显示菜单

while(inMenu){

ExMessage msg;

if (peekmessage(&msg, EX\_MOUSE)) {

if (msg.message == WM\_LBUTTONDOWN) {

int action = handleMenuClick(msg.x, msg.y);

if (action == 1) { // 开始游戏

init(map);

show(map);//显示雷区状况

startTime = 0; // 重置计时器

gameOver = false; // 重置游戏结束标志

inMenu = false; // 进入游戏模式

}

else if (action == 2) { // 显示排行榜

inMenu = false;

inRanking = true;

}

else if (action == 0) { // 退出游戏

closegraph();

exit(0);

}

}

}

}

}

else if (inRanking) {

drawRanking(); // 显示排行榜

while(inRanking){

ExMessage msg;

if (peekmessage(&msg, EX\_MOUSE)) {

if (msg.message == WM\_LBUTTONDOWN) {

int action = handleRankingClick(msg.x, msg.y);

if (action == 1) { // 返回菜单

inRanking = false;

inMenu = true;

}

}

}

}

}

else {

draw(map); // 绘制游戏界面

}

}

//游戏主循环

while (true) {

// 消息相应

ExMessage msg;

while (peekmessage(&msg, EX\_MOUSE)) {

switch (msg.message) {

case WM\_RBUTTONDOWN:// 鼠标右键点击

mouseMsg(&msg, map);

system("cls");

draw(map);

break;

case WM\_LBUTTONDOWN: // 鼠标左键点击

mouseMsg(&msg, map);

int ret = judge(map, msg.y / ImgSize, msg.x / ImgSize); // 点击之后判断

if (!gameOver && (ret == -1 || ret == 1)) {

stopTimer(); // 游戏结束时停止计时

draw(map);

gameOver = true; // 设置游戏结束标志

char resultMessage[200];

char title[50];

if (ret == -1) {

sprintf(resultMessage, "你也不行啊小杂鱼～说白了你有啥实力啊?敢再来一把嘛你？\n%s", displayTime());

strcpy(title, "失败");

}

else if (ret == 1) {

double gameDuration = (double)(endTime - startTime) / CLOCKS\_PER\_SEC;

updateRank(gameDuration); // 更新排行榜

sprintf(resultMessage, "烙铁你太有实力了！再来一把不？\n%s", displayTime());

strcpy(title, "胜利");

}

char fullMessage[700];

sprintf(fullMessage, "%s\n%s", resultMessage, getRankString());

int select = MessageBox(GetHWnd(), fullMessage, title, MB\_OKCANCEL);

if (select == IDOK) { // 再来一把

init(map);

startTime = 0; // 重置计时器

gameOver = false; // 重置游戏结束标志

}

else if (select == IDCANCEL) { // 退出游戏

closegraph();

exit(0);

}

}

system("cls");

printf("judge=%d\n", ret);

show(map);

break;

}

}

draw(map);

}

getchar();

return 0;

}