计算机程序设计基础大作业：
扫雷文档

梅树尧
物理系 基科 92
2019010122

目录

Chapter 1

实验报告

1.1 选题：扫雷

本大作业的选题为扫雷。扫雷的规则是在最短的时间内，根据某一格子四周出现的数字找出所有非雷格子，同时避免踩雷，踩到雷即判定为输。清除扫雷区的速度越快，得分就越高。

1.1.1 选题背景

扫雷是我们童年的游戏，随着 Windows 的一次次更新，Win10 中已经没有系统自带的扫雷。在学习了程序设计后，编写一个扫雷也是对童年的回忆。

1.1.2 实现功能

扫雷需要实现以下若干功能：

1. 翻开方块。此方块若不是雷则显示周围雷数，若是雷则结束游戏，若周围一圈没有雷则递归地打开周围所有方块。

2. 标记方块。在未翻开的方块上插旗和取消插旗。

3. 连续开启方块。在有数字方块上操作，当某方块位置周围已标记雷数等于该位置数字时操作有效，相当于对该数字周围未打开的方块均进行一次挖方块操作。地雷未标记完全时使用此操作无效。若数字周围有标错的地雷，则游戏结束，标错的地雷上会进行显示。

4. 判断输赢。当翻开雷时记为输，当所有不是雷的方块被打开记为赢。

5. 自定义扫雷区域的大小和地雷数目。

6. 计时和排行榜。

7. 背景音乐。

1.2 设计说明

本程序将扫雷功能的数据和算法与界面函数分离，并分别打包成了结构体。通过一个图形库 easyx 和一个媒体库实现了图形界面和 bgm 播放。

值得一提的是，本程序在更改完 Minemap 后，根据返回值立马绘图，没有对图像的持续刷新，节省了计算资源。另外，作为一个多窗口应用程序，我们将 bgm 放在了 main 函数，这样就可以让其一直播放，在今后的改进中打算使用多线程来进一步优化。

除去算法方面，我在 UI 的设计上也花费了相当多的心思。我依据我多年玩扫雷的经验，去繁就简，使 UI 清晰醒目。也简化了操作，想减少不必要的信息和操作对追求速度的同学的干扰，力图对刷记录的同学友好。

以下将介绍主要设计和实现思路。

1.2.1 Minemap

将 Minemap 打包成类后，程序逻辑更清晰，同时保证了 map 成员的数据安全。enum STATE 定义了格子状态。

Minemap
-map: vector<STATE> -opened_box: int -w: Win_para -start_time: clock_t -mine_last: int
+Minemap(col: int, row: int, w: Win_para) +leftClicked(col: int, row: int, w: Win_para): bool +rightClicked(col: int, row: int, w: Win_para) +get_state(col: int, row: int): STATE +is_win(): bool +coo(col: int, row: int): int +get_around_box(m: int, n: int): vector<int> +get_mine_last(): int +get_time(): int +get_time_precise(): double

Minemap 说明	
数据成员	
vector<STATE> map	每个格子状态
int opened_box	已经翻开的格子数量
Win_para w	指向窗口参数结构体的指针
clock_t start_time	开始时间
int mine_last	剩余雷数量
函数成员	
Minemap(int col, int row, Win_para w)	构造函数
bool leftClicked(int col, int row, Win_para w)	点击左键
void rightClicked(int col, int row, Win_para w)	点击右键
STATE get_state(int col, int row)	查询某一块的状态
bool is_win()	查询是否胜利
int get_mine_last()	查询剩余雷数
int get_time()	查询游戏时间（精确到秒）
double get_time_precise()	查询游戏时间（精确到毫秒）
int coo(int row, int col)	辅助函数，将行列编号与数组编号对应
vector<int> get_around_box(int m, int n)	辅助函数，返回一周格子编号

enum STATE
CLOSE_NO_BOOM = 0
OPEN_NO
OPEN_ONE
OPEN_TWO
OPEN_THREE
OPEN_FOUR
OPEN_FIVE
OPEN_SIX
OPEN_SEVEN
OPEN_EIGHT
BOOM
FLAG_BOOM
FLAG_NO_BOOM
CLOSE_BOOM

1.2.2 Winpara、Rank

结构体 Winpara 包括了窗口参数，Rank 储存了排行榜信息。

Win_para
+i_n[15]: IMAGE +box_m: int +box_n: int +total_num: int +box_length: int +title_height: int +title_left: int +rank: Rank +r1: RECT +r2: RECT
+Win_para()

Win_para 说明	
IMAGE i_n[15]	资源图片
int box_n	列数
int box_m	行数
int total_num	雷数
int box_length	格子宽度（像素）
int title_height	标题高度
int title_left	标题左侧 x 坐标
Rank rank	排行榜
RECT r1	窗口中绘制时间位置
RECT r2	窗口中绘制雷数位置
Win_para()	构造函数

RANK
low_time[5]: double low_name[150]: char mid_time[5]: double mid_name[150]: char high_time[5]: double high_name[150]: char

1.2.3 算法设计

以下为示意代码，只保留了骨干，源代码详见文档最后。通过 while 循环实现了控制流翻转。

```
int main()
{
    // 定义全局变量
    Win_para* w;                    //窗口参数
    Minemap* game=nullptr;          //游戏数据
    MOUSEMSG m;                     //鼠标消息
    AudioClip bgm;                  //背景音乐

    //初始化
    initialize(w);                  //初始化，并读取settings写入w
    open_window(w);                 //以w为参数打开窗口

    while (true)
    {
        m = GetMouseMsg();          // 获取一条鼠标消息
        switch(m)
        {
            case LEFTBUTTONUP:      //左键
            {
                if(m_in_gamebox())                  //鼠标点击在游戏区
                {
                    if(game==nullptr)               //开始新游戏
                        new_game(game);
                    game->leftClicked(row,col);
                    if(game->lose())                //检查失败
                        lose();
                    if(game->win())                 //检查胜利
                        win();
                }
                else
                {
                    switch(m.where)
                    {
                        case SET:                   //设置
                            set();break;            //更改settings文件，
                                                        //并重新打开窗口
                        case RANK:                  //显示排行榜
                            show_rank();break;
                        case MUSIC:                 //播放、停止音乐
                            music();break;
                        default:
                            break;
                    }
```

```
44                  }
45                  break;
46              }
47
48              case RIGHTBUTTONUP:         // 右键
49              {
50                  game->rightClicked(row,col);
51                  break;
52              }
53              default:
54                  break;
55          }
56
57          print_time_minelast();          //打印时间和剩余雷数
58          play_music();                   //检查音乐状态
59      }
60
61      close_window();
62  }
```

1.2.4 其他全局函数

另外，我们还给每个主要功能定义了相应的函数。

全局函数说明	
void draw_box(int col, int row, STATE s, Win_para* w)	绘图函数
void check_time(double t, Win_para* w, bool& music_played)	检查记录
void show_record(Rank* r);	打印排行榜
void change_settings(Win_para* w, Minemap*& game, bool& music_played)	更改设置
void play_music(Win_para* w, AudioClip* m, bool& music_played, int& music_time)	播放、停止音乐
void initgame(Win_para* w, bool music_played)	初始化
void open_window(Win_para* w, bool& music_played)	打开窗口
void close_window(Win_para* w)	关闭窗口
其他辅助函数	

1.3 总结

通过本次大作业我收获颇丰，在实践的过程中学到了许多。这是我第一个图形界面程序，虽然使用的是 easyx 这样简单的库，但是这是我第一次通过 while 和 switch 搭建出控制流反转，这是一次难得的代码经历。在我今后使用更高级的图形库时，我相信这次经历会让我受益。另外，我也通过较为合理的头文件安排，使得程序结构清晰，代码易读。

另外一个方面，我自高中起就十分喜欢扫雷，也曾花了大把时间将扫雷高级练到 100 秒。但是现在 Win10 已经不自带扫雷，网站上下载的扫雷总是玩得不顺手，我想写一个扫雷已经很久。这学期在学会了这么多的编程知识后，也借此机会完成了一个一直以来的心愿。

这个项目也凝聚着我的心血，我从设计之初便从用户体验出发，力图对刷记录的同学友好。我依据我多年玩扫雷的经验，特地设计了 UI，使其清晰醒目。我也希望这个大作业能够给其他热爱玩扫雷的同学带来快乐。

Chapter 2

readme

readme

```
扫雷
@copyright 梅树尧

操作：
1.左键开始游戏。
2.左键翻开。
3.右键插旗、取消插旗。
4.左键点击数字，若周围标记雷的数量与数字相同，翻开所有未标记的块。
5.左上角数字为剩余雷数。
6.右上角数字为游戏时间。
7.点击第一个按钮进入设置，第二个按钮查看排行榜，第三个按钮播放、停止
    音乐。

规则：
你需要确定所有雷的位置以获得胜利。方块上的数字为周围8格中的雷数。
如果你能够打破纪录，还可以把你的名字保存到排行榜。赶快开始游戏吧！
```

Chapter 3

用户手册

3.1 程序简介

程序名：扫雷
版本号：v1.0
本程序为一个扫雷游戏，复制了 Windows XP 中扫雷的游戏内容。

3.1.1 游戏规则

扫雷的规则是在最短的时间内，根据某一格子四周出现的数字找出所有非雷格子，同时避免踩雷，踩到雷即判定为输。你需要用最快的速度清除雷区，当你打破了记录还可以将自己的名字计入排行榜。具体规则为：

- 挖开方块出现地雷，游戏即告结束。
- 挖开方块出现数字，则该数字表示其周围 3×3 区域中的地雷数（一般为 8 个格子，对于边块为 5 个格子，对于角块为 3 个格子。所以扫雷中最大的数字为 8）。
- 挖开空方块（相当于 0），则可以递归地打开与空相邻的方块。

3.1.2 作者及版权信息

作者：梅树尧
单位：清华大学物理系
邮箱：msy19@mails.tsinghua.edu.cn
设计时间：2020.6~2020.6

3.2 使用说明

3.2.1 运行环境

32 位 Windows 操作系统，64 位可以兼容运行。

3.2.2 安装及配置

不需要安装，点击“扫雷.exe”运行游戏。运行游戏需要资源文件夹“image”，否则无法打开游戏。需要“bgm.mp3”播放背景音乐，否则不能使用背景音乐功能。还有一个可选的“settings”文件保存设置及记录，若缺失应用程序会自动生成。

图 3.1: 所有文件

3.2.3 开始游戏

点击任意方块开始游戏，并视为第一次翻开操作，此时开始计时。另外，在第一次点击的方块附近一定没有雷。

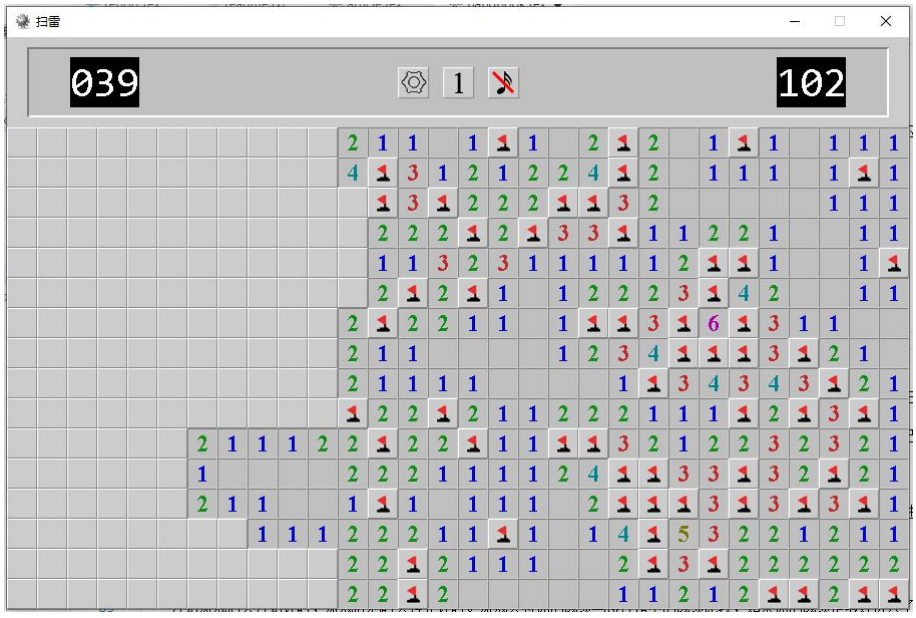

图 3.2: 开始游戏、翻开、插旗、连续开启

3.2.4 翻开

点击左键翻开方块。若点击到雷，游戏结束。若未点击到雷，在此方块显示周围雷的数量，若此方块周围没有雷，则递归地打开周围所有方块。

3.2.5 插旗

点击右键插旗，即标记雷的位置。不论一个格子中是否真的有雷，你都可以将其标记为雷，但是错误的标记可能将你自己引向错误的判断而导致游戏失败。

另外，你可以再次用右键点击已经插旗的方块将旗拔掉，即取消标记。

左上角的数字为没有标记的雷的数量。

图 3.3: 剩余雷的数量

3.2.6 连续打开方块

在有数字方块上操作，当某方块位置周围已标记雷数等于该位置数字时操作有效，相当于对该数字周围未打开的方块均进行一次挖方块操作。地雷未标记完全时使用此操作无效。若数字周围有标错的地雷，则游戏结束，标错的地雷上会进行显示。

3.2.7 判定胜负

当点击到雷时，游戏失败。当翻开所有不是雷的方块时，游戏胜利，注意插旗不是胜利的必要条件。

图 3.4: 胜利

图 3.5: 失败

3.2.8 计时及排行榜

右上角数字为游戏时间，开始游戏后会开始计时，游戏胜利时会停止计时。游戏会将你的成绩与排行榜中的成绩比较，如果你的成绩足够好进入了排行榜，你可以输入自己的名字并将其保存。

点击“1”按钮（左起第二个）查看排行榜。

图 3.7: 游戏时间

图 3.8: 排行榜

图 3.9: 打破纪录

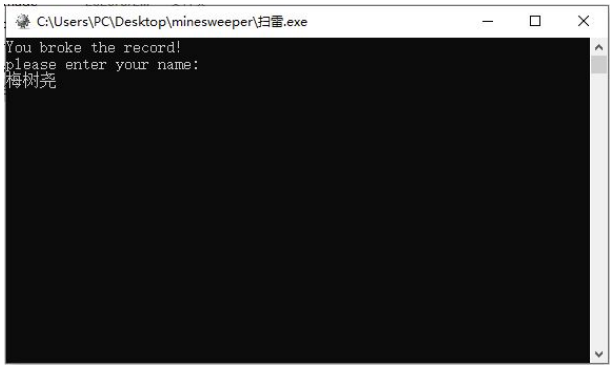

图 3.10: 保存记录

3.2.9 游戏难度

点击齿轮按钮（左起第一个）设置游戏难度。雷区的大小和雷的数量决定了游戏难度，游戏有三种默认难度：

- 初级：81 个方块 (9*9)、10 个雷
- 中级：256 个方块 (16*16)、40 个雷
- 高级：480 个方块 (16*30)、99 个雷

还可以自定义区域和雷数，但只有标准难度才能将成绩计入排行榜。

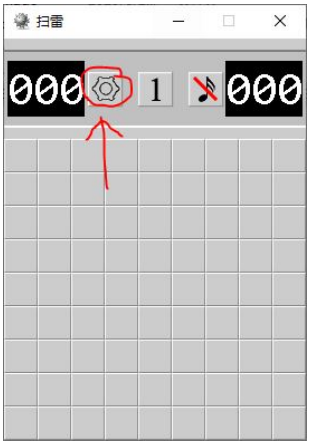

图 3.10: 齿轮按钮

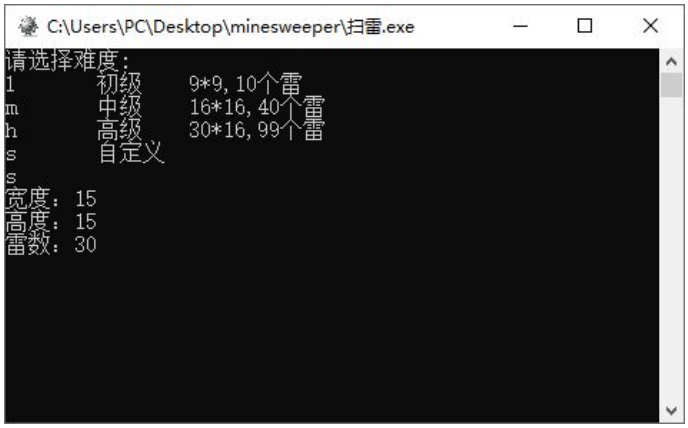

图 3.11: 选择游戏难度

3.2.10 背景音乐

点击音乐按钮（左起第三个）打开、关闭背景音乐，若根目录下没有“bgm.mp3”此功能不可用。玩家可以手动更改 bgm，但 mp3 文件必须没有封面、作者等信息。

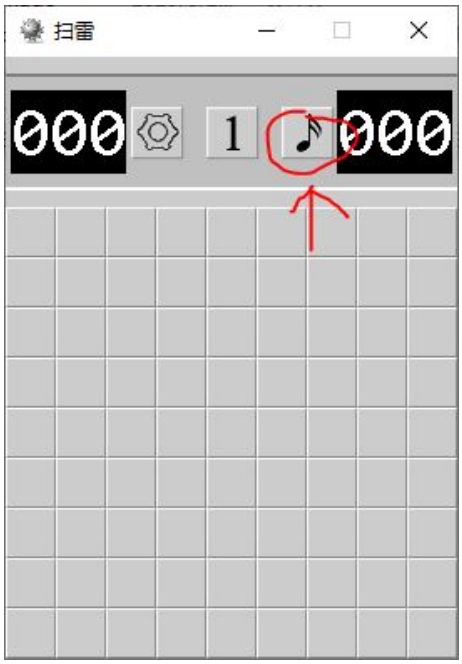

图 3.12: 背景音乐

Chapter 4

源代码

mine.cpp/main.cpp

```
#include "minemap.h"

void check_time(double t, Win_para* w, bool& music_played);        //检查排行榜
void show_record(Rank* r);                        //打印排行榜
void change_settings(Win_para* w, Minemap*& game, bool& music_played);                        //更改设置
void play_music(Win_para* w, AudioClip* m, bool& music_played, int& music_time);                        //音乐
void open_window(Win_para* w, bool& music_played);
void close_window(Win_para* w);
void initgame(Win_para* w, bool music_played);            //初始化画图


int main()
{
    AudioClip ac;                        //加载bgm
    bool music_played=false;
    int music_time = 0;
    //ac.load("D:/bgm.mp3");
    ac.load("./bgm.mp3");

    Win_para* w=new Win_para;
    //加载资源
    for (int i = 0; i < 15; i++)
        if ((w->i_n + i)->getheight() == 0)
        {
            std::cout << "Cannot read images!" << std::endl;    //检查资源是否加载
            char c;
            std::cin.get(c);
            return 1;
        }

    // 初始化图形窗口
    open_window(w,music_played);

    MOUSEMSG m;        // 定义鼠标消息
    Minemap* game=nullptr;

    while (true)
    {
        // 获取一条鼠标消息
        m = GetMouseMsg();

        switch (m.uMsg)
        {
        case WM_LBUTTONUP:
            //左键
        {
            if (m.y < w->title_height)
            {
                if (m.y > w->title_height / 3 && m.y < w->title_height * 2 / 3)
                {
```

```
51                      if (m.x > w->title_height * 29 / 6 + w->title_left && m.x < w->title_height * 31 / 6 + w->
      title_left)
52                          show_record(&(w->rank));
53                      else if (m.x > w->title_height * 13 / 3 + w->title_left && m.x < w->title_height * 14 / 3 + w->
      title_left)
54                          change_settings(w,game, music_played);
55                      else if (m.x > w->title_height * 16 / 3 + w->title_left && m.x < w->title_height * 17 / 3 + w->
      title_left)
56                          play_music(w,&ac,music_played,music_time);
57                  }
58                  break;
59              }
60              int col = m.x / w->box_length;
61              int row = (m.y- w->title_height) / w->box_length;
62              if (game == nullptr)
63              {
64                  initgame(w,music_played);
65                  game = new Minemap(col,row,w);
66              }
67              if (game->leftClicked(col, row, w))      //lose
68              {
69                  delete game;
70                  game = nullptr;
71                  MessageBox(NULL, _T("You lose!"), _T(""), MB_OK|MB_SYSTEMMODAL);
72              }
73              if (game!=nullptr&&game->is_win())
74              {
75                  double total_time = game->get_time_precise();
76                  MessageBox(NULL, _T("You win!"), _T(""), MB_OK | MB_SYSTEMMODAL);
77                  check_time(total_time,w,music_played);
78                  delete game;
79                  game = nullptr;
80              }
81              break;
82          }
83
84          case WM_RBUTTONUP:
85              // 右键
86              if (game != nullptr)
87                  game->rightClicked(m.x / w->box_length,( m.y- w->title_height) / w->box_length, w);
88              break;
89
90          }
91          //时间和雷数
92          if (game != nullptr)
93          {
94              TCHAR t[5];
95              _stprintf_s(t, _T("%03d"), game->get_mine_last());
96              drawtext(t, &(w->r1), DT_CENTER | DT_VCENTER | DT_SINGLELINE);
97              _stprintf_s(t, _T("%03d"), game->get_time());
98              drawtext(t, &(w->r2), DT_CENTER | DT_VCENTER | DT_SINGLELINE);
99          }
100
101         if (music_played == true && clock() - music_time > ac.milliseconds() + 1000)         //音乐循环
102         {
103             ac.stop();
104             ac.play();
105             music_time = clock();
106         }
107
108     }
109
110     // 关闭图形窗口
111     close_window(w);
112 }
113
114
115 void read_name(double t, double time[5], char name[30], Win_para* w, bool& music_played)         //破纪录后读姓名，保存
      记录
116 {
117     for (int i = 0; i < 5; i++)
118         if (t < time[i])
119         {
120             if (MessageBox(NULL, _T("You broke the record! Do you want to save it?"), _T("Congratulations!"), MB_YESNO
```

```
        | MB_SYSTEMMODAL) == IDNO)
121         {
122             return;
123         }
124         closegraph();
125         for (int j = 4; j > i; j--)
126         {
127             (time)[j] = (time)[j - 1];
128             strcpy_s((name + j * 30), 28, (name + (j - 1) * 30));
129         }
130         (time)[i] = t;
131         std::cout << "You broke the record!\nplease enter your name:\n";
132         std::string s;
133         std::cin >> s;
134         while (s.length() > 28)
135         {
136             std::cout << "too long name, less than 28 characters\nyour name:";
137             std::cin >> s;
138         }
139         strcpy_s((name + i * 30), 28, s.c_str());
140         std::ofstream os("./settings", std::ios_base::out | std::ios_base::binary);
141         os.clear();
142         os.write(reinterpret_cast<char*>(&(w->box_n)), sizeof(int));
143         os.write(reinterpret_cast<char*>(&(w->box_m)), sizeof(int));
144         os.write(reinterpret_cast<char*>(&(w->total_num)), sizeof(int));
145         os.write(reinterpret_cast<char*>(&(w->rank)), sizeof(Rank));
146         os.close();
147         open_window(w, music_played);
148         show_record(&(w->rank));
149         break;
150     }
151 }
152
153 void check_time(double t, Win_para* w, bool& music_played)                    //检查是否破纪录
154 {
155     if (w->box_n == 30 && w->box_m == 16 && w->total_num == 99)
156     {
157         read_name(t, w->rank.high_time, w->rank.high_name, w, music_played);
158     }
159     else if (w->box_n == 16 && w->box_m == 16 && w->total_num == 40)
160     {
161         read_name(t, w->rank.mid_time, w->rank.mid_name, w, music_played);
162     }
163     else if (w->box_n == 9 && w->box_m == 9 && w->total_num == 10)
164     {
165         read_name(t, w->rank.low_time, w->rank.low_name, w, music_played);
166     }
167 }
168
169 void show_record(Rank* r)                                   //打印记录
170 {
171     setlocale(LC_CTYPE, "chs");
172     WCHAR t[400];
173     swprintf_s(t, 400, L"初级:\t1\t%.3lf\t%S\n\t2\t%.3lf\t%S\n\t3\t%.3lf\t%S\n\t4\t%.3lf\t%S\n\t5\t%.3lf\t%S\n中级:\t1
       \t%.3lf\t%S\n\t2\t%.3lf\t%S\n\t3\t%.3lf\t%S\n\t4\t%.3lf\t%S\n\t5\t%.3lf\t%S\n高级:\t1\t%.3lf\t%S\n\t2\t%.3lf\t%S
       \n\t3\t%.3lf\t%S\n\t4\t%.3lf\t%S\n\t5\t%.3lf\t%S\n"
174         , r->low_time[0], r->low_name, r->low_time[1], r->low_name + 30, r->low_time[2], r->low_name + 60, r->low_time
       [3], r->low_name + 90, r->low_time[4], r->low_name + 120
175         , r->mid_time[0], r->mid_name, r->mid_time[1], r->mid_name + 30, r->mid_time[2], r->mid_name + 60, r->mid_time
       [3], r->mid_name + 90, r->mid_time[4], r->mid_name + 120
176         , r->high_time[0], r->high_name, r->high_time[1], r->high_name + 30, r->high_time[2], r->high_name + 60, r->
       high_time[3], r->high_name + 90, r->high_time[4], r->high_name + 120);
177     MessageBox(NULL, t, L"ranking list", MB_OK | MB_SYSTEMMODAL);
178 }
179
180 int read_int(int min, int max)                              //设置，读数字
181 {
182     int a;
183     char c;
184     std::cin.get(c);
185     if (!isdigit(c))
186     {
187         std::cin.ignore(100, '\n');
188         std::cout << "请输入范围正确的数字，最小" << min << "，最大" << max << "\n";
```

```
189         return read_int(min, max);
190     }
191     std::cin.putback(c);
192     std::cin >> a;
193     std::cin.ignore(100, '\n');
194     if (a<min || a>max)
195     {
196         std::cout << "请输入范围正确的数字，最小" << min << "，最大" << max << "\n";
197         return read_int(min, max);
198     }
199     return a;
200 }
201
202 void change_settings(Win_para* w, Minemap*& game, bool& music_played)              //设置
203 {
204     if (game != nullptr)
205         if (MessageBox(NULL, _T("你想进入设置吗？这将终止正在进行的游戏。"), _T("warning"), MB_YESNO | MB_SYSTEMMODAL)
        == IDNO)
206         {
207             return;
208         }
209         else
210         {
211             delete game;
212             game = nullptr;
213         }
214     closegraph();
215     std::cout << "请选择难度:\nl\t初级\t9*9,10个雷\nm\t中级\t16*16,40个雷\nh\t高级\t30*16,99个雷\ns\t自定义\n";
216     char c;
217     std::cin.get(c);
218     std::cin.ignore(100, '\n');
219     while (c != 'l' && c != 'm' && c != 'h' && c != 's')
220     {
221         std::cout << "请输入正确的字符\n";
222         std::cin.get(c);
223         std::cin.ignore(100, '\n');
224     }
225     switch (c)
226     {
227     case 'l':
228         w->box_m = 9;
229         w->box_n = 9;
230         w->total_num = 10;
231         break;
232     case 'm':
233         w->box_m = 16;
234         w->box_n = 16;
235         w->total_num = 40;
236         break;
237     case 'h':
238         w->box_n = 30;
239         w->box_m = 16;
240         w->total_num = 99;
241         break;
242     case 's':
243     {
244         std::cout << "宽度:";
245         w->box_n = read_int(9, 100);
246         std::cout << "高度:";
247         w->box_m = read_int(9, 50);
248         std::cout << "雷数:";
249         w->total_num = read_int(1, w->box_m * w->box_n - 9);
250         break;
251     }
252     }
253     std::ofstream os("./settings", std::ios_base::out | std::ios_base::binary);
254     os.clear();
255     os.write(reinterpret_cast<char*>(&(w->box_n)), sizeof(int));
256     os.write(reinterpret_cast<char*>(&(w->box_m)), sizeof(int));
257     os.write(reinterpret_cast<char*>(&(w->total_num)), sizeof(int));
258     os.write(reinterpret_cast<char*>(&(w->rank)), sizeof(Rank));
259     os.close();
260     delete w;
261     w = new Win_para();
```

```
262     open_window(w, music_played);
263 }
264
265 void play_music(Win_para* w, AudioClip* m, bool& music_played, int& music_time)      //音乐
266 {
267     if (m->milliseconds() == 0)
268     {
269         MessageBox(NULL, _T("Can't load background music."), _T("error"), MB_OK | MB_SYSTEMMODAL);
270         return;
271     }
272
273     if (music_played == false)
274     {
275         m->play();
276         music_time = clock();
277         putimage(w->title_height * 16 / 3 + w->title_left, w->title_height / 3, w->i_n + 13);
278         music_played = true;
279     }
280     else
281     {
282         m->stop();
283         putimage(w->title_height * 16 / 3 + w->title_left, w->title_height / 3, w->i_n + 14);
284         music_played = false;
285     }
286 }
287
288 void open_window(Win_para* w, bool& music_played)
289 {
290     initgraph(w->box_n * w->box_length, w->box_m * w->box_length + w->title_height);
291     initgame(w, music_played);
292 }
293 void close_window(Win_para* w)
294 {
295     closegraph();
296     delete w;
297 }
298
299 void initgame(Win_para* w, bool music_played)
300 {
301     cleardevice();
302     putimage(w->title_left, 0, w->i_n + 12);
303     if (music_played == false)
304         putimage(w->title_height * 16 / 3 + w->title_left, w->title_height / 3, w->i_n + 14);
305     for (int i = 0; i < (w->box_m); i++)
306         for (int j = 0; j < (w->box_n); j++)
307         {
308             putimage(j * (w->box_length), i * (w->box_length) + w->title_height, (w->i_n + 9));
309         }
310     settextcolor(WHITE);
311     settextstyle(50, 0, _T("Consolas"));
312     drawtext(_T("000"), &(w->r1), DT_CENTER | DT_VCENTER | DT_SINGLELINE);
313     drawtext(_T("000"), &(w->r2), DT_CENTER | DT_VCENTER | DT_SINGLELINE);
314 }
```

minemap.h

```
1  #ifndef MINEMAP_H
2  #define MINEMAP_H
3
4  #include<stdlib.h>
5  #include<time.h>
6  #include<vector>
7  #include <graphics.h>
8  #include <conio.h>
9  #include<iostream>
10 #include<fstream>
11 #include<string>
12 #include<sstream>
13 #include<iomanip>
14 #include"AudioClip.h"    //音乐库
15
16 enum STATE              //格子状态
17 {
```

```
    CLOSE_NO_BOOM= 0,OPEN_NO , OPEN_ONE, OPEN_TWO, OPEN_THREE, OPEN_FOUR, OPEN_FIVE, OPEN_SIX, OPEN_SEVEN, OPEN_EIGHT,
    BOOM, FLAG_BOOM, FLAG_NO_BOOM, CLOSE_BOOM
};

struct Rank              //排行榜
{
    double low_time[5];
    char low_name[150];
    double mid_time[5];
    char mid_name[150];
    double high_time[5];
    char high_name[150];
};

struct Win_para              //窗口参数
{
    Win_para();

    IMAGE i_n[15];
    int box_n;
    int box_m;
    int total_num;
    int box_length;
    int title_height;
    int title_left;
    Rank rank;
    RECT r1, r2;    //时间雷数位置
};

struct Minemap        //行列编号从0开始
{
    Minemap(int col,int row,Win_para* w);
    bool leftClicked(int col, int row, Win_para* w);
    void rightClicked(int col, int row, Win_para* w);
    STATE get_state(int col,int row);
    bool is_win();
    int coo(int row, int col);       //返回编号
    std::vector<int> get_around_box(int m, int n);//返回周围的格子编号
    int get_mine_last();
    int get_time();
    double get_time_precise();

    //private
    std::vector<STATE> map;
    int opened_box;
    Win_para* w;
    clock_t start_time;
    int mine_last;
};

bool is_in_vector(const std::vector<int>& a,int n);      //辅助函数，查找
void draw_box(int col, int row, STATE s, Win_para* w);  //画一个格子

#endif // MINEMAP_H
```

minemap.cpp

```
#include "minemap.h"

Minemap::Minemap(int col,int row, Win_para* ww):opened_box(0),w(ww),mine_last(w->total_num)//输入初始点击坐标
{
    int total_box = w->box_m * w->box_n;
    map.resize(total_box);
    srand((int)time(0));
    std::vector<int> around=get_around_box(row,col);
    around.push_back(coo(row, col));
    int i=0;
    while (i< w->total_num) {
        int n=(rand()%(total_box));
        if(is_in_vector(around,n))
            continue;
        if(map[n]!=CLOSE_BOOM)
        {
```

```
17              map[n]=CLOSE_BOOM;
18              i++;
19          }
20      }
21      start_time= clock();
22  }
23
24  int Minemap::get_mine_last()
25  {
26      return mine_last;
27  }
28
29  int Minemap::get_time()
30  {
31      clock_t end = clock();
32      return (end - start_time ) / CLOCKS_PER_SEC;
33  }
34  double Minemap::get_time_precise()
35  {
36      clock_t end = clock();
37      return (double)(end - start_time) / CLOCKS_PER_SEC;
38  }
39
40  bool Minemap::leftClicked (int col,int row, Win_para* w)
41  {
42      int num=coo(row,col);
43      switch (map[num])
44      {
45          case CLOSE_BOOM:
46              map[num]=BOOM;
47              putimage(col * w->box_length, row * w->box_length + w->title_height, w->i_n + 11);  //draw boom
48              return true;
49          case CLOSE_NO_BOOM:
50          {
51              opened_box++;
52               std::vector<int> around=get_around_box(row,col);
53               int around_num=0;
54               for(int i:around)
55                   if(map[i]==CLOSE_BOOM||map[i]==FLAG_BOOM)
56                       around_num++;
57               map[num]=(STATE)(around_num+1);      //STATE中雷数与编号相差1
58               draw_box(col, row, map[num], w);
59               if(around_num==0)
60                   for(int i:around)
61                       leftClicked(i%w->box_n,i/ w->box_n, w);
62               return false;
63          }
64          case OPEN_ONE: case OPEN_TWO: case OPEN_THREE: case OPEN_FOUR:
65          case OPEN_FIVE: case OPEN_SIX: case OPEN_SEVEN: case OPEN_EIGHT:
66          {
67              std::vector<int> around = get_around_box(row, col);
68              int around_num = 0;
69              for (int i : around)
70                  if (map[i] == FLAG_NO_BOOM || map[i] == FLAG_BOOM)
71                      around_num++;
72              bool b = 0;
73              if (around_num+1 == map[num])          //STATE中雷数与编号相差1
74              {
75                  for (int i : around)
76                      if (map[i] == CLOSE_BOOM || map[i] == CLOSE_NO_BOOM)
77                          b|=leftClicked(i % w->box_n, i / w->box_n, w);
78              }
79              return b;
80          }
81          default:
82              return false;
83      }
84  }
85
86  void Minemap::rightClicked(int col,int row, Win_para* w)
87  {
88      int num=coo(row,col);
89      switch (map[num])
90      {
```

```
91      case FLAG_BOOM:
92          map[num]=CLOSE_BOOM;
93          draw_box(col, row, CLOSE_BOOM, w);
94          mine_last++;
95          return;
96      case FLAG_NO_BOOM:
97          map[num]=CLOSE_NO_BOOM;
98          draw_box(col, row, CLOSE_NO_BOOM, w);
99          mine_last++;
100         return;
101     case CLOSE_BOOM:
102         map[num]=FLAG_BOOM;
103         draw_box(col, row, FLAG_BOOM, w);
104         mine_last--;
105         return;
106     case CLOSE_NO_BOOM:
107         map[num]=FLAG_NO_BOOM;
108         draw_box(col, row, FLAG_NO_BOOM, w);
109         mine_last--;
110         return;
111     default:
112         return;
113     }
114 }
115
116 STATE Minemap::get_state(int col,int row)
117 {
118     return map[coo(row,col)];
119 }
120
121 bool Minemap::is_win()
122 {
123     if(opened_box==w->box_m* w->box_n- w->total_num)
124         return true;
125     else
126         return false;
127 }
128
129 int Minemap::coo(int row,int col)//返回编号
130 {
131     return row* w->box_n+col;
132 }
133
134 std::vector<int> Minemap::get_around_box(int m,int n)//返回周围一圈编号,row,col
135 {
136     std::vector<int> around;
137     if(n>0)
138         around.push_back(coo(m,n-1));
139     if(n<w->box_n-1)
140         around.push_back(coo(m,n+1));
141     if(m>0)
142     {
143         around.push_back(coo(m-1,n));
144         if(n>0)
145             around.push_back(coo(m-1,n-1));
146         if(n<w->box_n-1)
147             around.push_back(coo(m-1,n+1));
148     }
149     if(m<w->box_m-1)
150     {
151         around.push_back(coo(m+1,n));
152         if(n>0)
153             around.push_back(coo(m+1,n-1));
154         if(n<w->box_n-1)
155             around.push_back(coo(m+1,n+1));
156     }
157     return around;
158 }
159
160 Win_para::Win_para ():box_n(30), box_m(16), total_num(99) , box_length(30),title_height(90),title_left(0)
161 {
162     std::ifstream settings("./settings", std::ios_base::binary);
163     if (settings) {
164         settings.read(reinterpret_cast<char*>(&box_n), sizeof(int));
```

```
165         settings.read(reinterpret_cast<char*>(&box_m), sizeof(int));
166         settings.read(reinterpret_cast<char*>(&total_num), sizeof(int));
167         settings.read(reinterpret_cast<char*>(&rank), sizeof(Rank));
168     }
169
170     loadimage(i_n+9, _T("./image/u.bmp"));            //release or debug
171     loadimage(i_n+10, _T("./image/f.bmp"));
172     loadimage(i_n, _T("./image/b.bmp"));
173     loadimage(i_n + 1, _T("./image/1.bmp"));
174     loadimage(i_n + 2, _T("./image/2.bmp"));
175     loadimage(i_n + 3, _T("./image/3.bmp"));
176     loadimage(i_n + 4, _T("./image/4.bmp"));
177     loadimage(i_n + 5, _T("./image/5.bmp"));
178     loadimage(i_n + 6, _T("./image/6.bmp"));
179     loadimage(i_n + 7, _T("./image/7.bmp"));
180     loadimage(i_n + 8, _T("./image/8.bmp"));
181     loadimage(i_n + 11, _T("./image/boom.bmp"));
182     if (box_n > 30)
183     {
184         title_height = (box_n * box_length) / 10;
185         loadimage(i_n + 12, _T("./image/title.bmp"), box_n * box_length, title_height);
186         loadimage(i_n + 14, _T("./image/no_music.bmp"), title_height / 3, title_height / 3);
187         loadimage(i_n + 13, _T("./image/music.bmp"), title_height / 3, title_height / 3);
188     }
189     else
190     {
191         title_left = (box_n * box_length - 900) / 2;
192         loadimage(i_n + 12, _T("./image/title.bmp"));
193         loadimage(i_n + 14, _T("./image/no_music.bmp"));
194         loadimage(i_n + 13, _T("./image/music.bmp"));
195     }
196
197     //时间雷数位置r1,r2
198     if (title_left < -30)
199     {
200         if (title_left <= -270)
201         {
202             r1 = { 0, 20,75 ,70 };
203             r2 = { box_n * box_length - r1.right, r1.top ,box_n * box_length - r1.left ,r1.bottom };
204         }
205         else
206         {
207             r1 = { 30, 20,105 ,70 };
208             r2 = { box_n * box_length - r1.right, r1.top ,box_n * box_length - r1.left ,r1.bottom };
209         }
210     }
211     else
212     {
213         r1 = { title_height * 2 / 3, title_height * 2 / 9,title_height * 3 / 2,title_height * 7 / 9 };
214         r2 = { box_n * box_length - r1.right, r1.top ,box_n * box_length - r1.left ,r1.bottom };
215     }
216 }
217
218
219 bool is_in_vector(const std::vector<int>& a, int n)
220 {
221     for (int i : a)
222         if (i == n)
223             return true;
224     return false;
225 }
226
227 void draw_box(int col, int row, STATE s, Win_para* w)
228 {
229     int x = col * w->box_length;
230     int y = row * w->box_length + w->title_height;
231     switch (s)
232     {
233     case OPEN_NO: {
234         putimage(x, y, w->i_n);
235         break;
236     }
237     case OPEN_ONE: {
238         putimage(x, y, w->i_n);
```

```
239             putimage(x + 5, y + 5, w->i_n + 1);
240             break;
241         }
242         case OPEN_TWO: {
243             putimage(x, y, w->i_n);
244             putimage(x + 5, y + 5, w->i_n + 2);
245             break;
246         }
247         case OPEN_THREE: {
248             putimage(x, y, w->i_n);
249             putimage(x + 5, y + 5, w->i_n + 3);
250             break;
251         }
252         case OPEN_FOUR: {
253             putimage(x, y, w->i_n);
254             putimage(x + 5, y + 5, w->i_n + 4);
255             break;
256         }
257         case OPEN_FIVE: {
258             putimage(x, y, w->i_n);
259             putimage(x + 5, y + 5, w->i_n + 5);
260             break;
261         }
262         case OPEN_SIX: {
263             putimage(x, y, w->i_n);
264             putimage(x + 5, y + 5, w->i_n + 6);
265             break;
266         }
267         case OPEN_SEVEN: {
268             putimage(x, y, w->i_n);
269             putimage(x + 5, y + 5, w->i_n + 7);
270             break;
271         }
272         case OPEN_EIGHT: {
273             putimage(x, y, w->i_n);
274             putimage(x + 5, y + 5, w->i_n + 8);
275             break;
276         }
277         case FLAG_BOOM: case FLAG_NO_BOOM: {
278             putimage(x + 5, y + 5, w->i_n + 10);
279             break;
280         }
281         case CLOSE_BOOM: case CLOSE_NO_BOOM:
282             putimage(x, y, w->i_n + 9);
283         }
284 }
```