

浙江大学

程 序 设 计 专 题

大 程 序 报 告



2021~2022 春夏学期    2022 年 5 月 23 日

## 目 录

<b>1</b>	<b>大程序简介 .....</b>	<b>3</b>
1.1	背景及意义 .....	3
1.2	目标要求 .....	3
1.3	术语说明 .....	3
<b>2</b>	<b>功能需求分析.....</b>	<b>6</b>
<b>3</b>	<b>程序开发设计.....</b>	<b>8</b>
3.1	总体架构设计 .....	7
3.2	功能模块设计 .....	7
3.3	数据结构设计 .....	11
3.4	源代码文件组织设计 .....	13
3.5	函数设计描述 .....	15
<b>4</b>	<b>部署运行和使用说明 .....</b>	<b>26</b>
4.1	编译安装 .....	26
4.2	运行测试 .....	32
4.3	使用操作 .....	47
<b>5</b>	<b>团队合作 .....</b>	<b>54</b>
5.1	团队分工 .....	54
5.2	开发计划 .....	55
5.3	编码规范 .....	56
5.4	合作总结（匿名） .....	58
5.5	收获感言（匿名） .....	61
<b>6</b>	<b>参考文献资料.....</b>	<b>63</b>

# 求是扫雷大程序设计

## 1 大程序简介

### 1.1 选题背景及意义

自新冠疫情以来，隔离已成常态，其中居家隔离和酒店隔离占大多数。个体在社交环境缺乏的大背景下，更偏向于自娱自乐的行为，加之社会兴起怀旧风大浪，经典游戏扫雷自然成为娱乐活动的第一梯队。

本项目旨在复现扫雷经典的前提下，丰富扫雷的功能、完善扫雷的不足，使之更贴近现在更习惯于手游、ACG 等形式游戏的玩家，毕竟娱乐性和创新性是一个游戏的最终目的。在游戏难度的设置上，本项目仅挑选了经典扫雷的中级难度，保证了游玩的“硬核度”。就学习内容而言，借助设计《求是扫雷》的大程序，有利于我们综合利用所学知识，补全知识空白，强化编程能力，为日后进一步的程序设计以及利用编程解决生活中的实际问题、创造生产力打下坚实的基础；在游戏开发的过程中体会到更多的游戏思想，也有助于未来可能的游戏开发实践。

### 1.2 目标要求

设计一个经典扫雷游戏，并进行适当的功能升级。程序功能包括：1.扫雷基础功能；2.按照种子打开扫雷地图，使相同的地图可重复游玩；3.角色扮演，配置相应的技能；4.允许开关的背景音乐；5.存储本地的分数排行榜；6.要求图形界面满足一定的美学。

### 1.3 术语说明

#### 1) 插入位图相关说明

osdc: graphics.c 里定义的, 当前窗口的设备上下文环境(DC)gdc 的兼容 DC, 通常情况下, 如果是同一类设备, 创建的 DC 的初始化环境是相同的, 问题在于

设备 DC 在变化,运行过程中一定会调整属性,如果再创建一个同类设备的 DC,初始属性肯定不一样,不兼容的可能性是很大的,所以以某个运行时刻的 DC 为基准创建一个兼容 DC,将复制当前时刻的 DC 属性,它的属性可以保证是相同的,这才是兼容 DC 的目的,而不是直接使用 CreateDC。

**pixelHeight、pixelWidth:** graphics 里定义的,窗口的像素高度和像素宽度。

**SelectObject:** 计算机编程语言函数,该函数选择一对象到指定的设备上下文中,该新对象替换先前的相同类型的对象。本项目中一般将位图指定到 osdc 的兼容 DC mdc 中,这样能够防止闪烁,是为双缓冲技术。

**BitBlt:** 函数对指定的源设备环境区域中的像素进行位块 (bit\_block) 转换,以传送到目标设备环境。关于贴图方式,本项目采取了直接贴图法和透明遮罩法。

**Delete 函数:** 及时删除 DC 和相关的 Object,释放资源。

## 2) 快捷键说明

重新开始游戏 —— 回车键

更换地图/打开种子/清空种子 —— Ctrl

使用技能 —— E

退出游戏 —— Esc

## 3) 播放音乐相关说明

**PlaySound 函数:** 在使用该函数时,共使用了三个关键 DWORD 类型参数。

**SND\_FILENAME** 指定了确认的 WAVE 文件名;

**SND\_ASYNC** 确保使用异步方式播放声音;

**SND\_LOOP** 可以保证背景音乐循环播放。

**mciSendString 函数:** 为实现多音轨播放,采用此函数。

三个关键指令: open 即打开音乐文件; play 即播放音乐文件; close 即关闭文件。

注意只有正确关闭音乐文件才能实现下一次对文件的打开和操作。

**SendString 函数:** 为实现多音轨播放,采用此函数。三个关键指令 open 即打开音乐文件;

play 即播放音乐文件; close 即关闭文件。注意只有正确关闭音乐文件才能实现下一次对文件的打开和操作。

## 2 功能需求分析

### 2.1 业务逻辑架构

#### 2.11 用户端表示层

扫雷游戏依赖图形界面直观呈现，通过图形界面进行交互操作。本程序需要一个完整的图形界面用于结果展示和交互功能。

#### 2.12 业务逻辑层

通过鼠标、键盘回调函数接收来自表示层的数据请求，逻辑判断后，向数据访问层提交请求，并传递数据访问结果。

#### 2.13 数据访问层

核心：采用结构体数组存储扫雷地图相关信息，进行程序使用过程中的数据比对、读取和存储，将每局游戏的分数以 txt 文件存储在用户磁盘，通过文件操作进行数据读取。

### 2.2 功能需求

基于候选题目中的基本要求和用户交互体验，梳理本程序功能需求如下。

#### 2.2.1 扫雷地图生成

要求初始化扫雷地图，并翻开第一个区域。生成地图时，既可以输入种子值，也可以直接生成（亦可以获取该种子值）。为了可重复游玩，要求针对某个种子值，第一个区域是固定的。为了可玩性，要求翻开的第一个区域大于等于两个格子。

#### 2.2.2 资源加载

要求载入背景音乐、角色位图、游戏背景、游戏图标等资源。

#### 2.2.3 扫雷游戏呈现

在游戏开始之前，要求在图形界面进行人物选择的交互，输入种子值以初始化地图等。

游戏时，要求在图形界面呈现扫雷游戏。要求能显示扫雷棋盘、剩余雷数、计时和角色。要求交互时，能翻开格子、判断炸弹、插旗、使用技能、退出游戏。

## 2.24 菜单栏与快捷键

要求菜单栏具有基础操作，包括打开种子、排行榜显示、开关背景音乐、打开开发者名单和退出游戏。要求实现若干个菜单中常见功能/命令的快捷键。

## 2.25 引导与提示

程序有一定的使用方法和局限。需要多维度的游戏引导。

## 2.3 数据结构需求

从内存利用和项目需求出发，由于此项目涉及数据存储并不多，占有内存不占大头，因此采取多个顺序表（具体来说，就是结构数组），存储扫雷地图的雷坐标、某格周围雷数、是否插旗和是否翻开。

另外，文件存储方面，由于数据读取采用了更合适的方法，这里选择了单向链表。单向链表也仅仅用了文件读取和写入的中间过程，保证效率即可。

## 2.4 性能需求

### 2.4.1 运行效率

本程序在展示之外需要和用户进行有效交互，因此交互功能响应应当迅速，做到低延迟、高刷新。在 SimpleGUI 高刷新频率支持下，也应当改良算法，尽量避免遍历操作。

### 2.4.2 适用平台

目前仅考虑 windows 平台的运行。

### 2.4.3 编译器要求

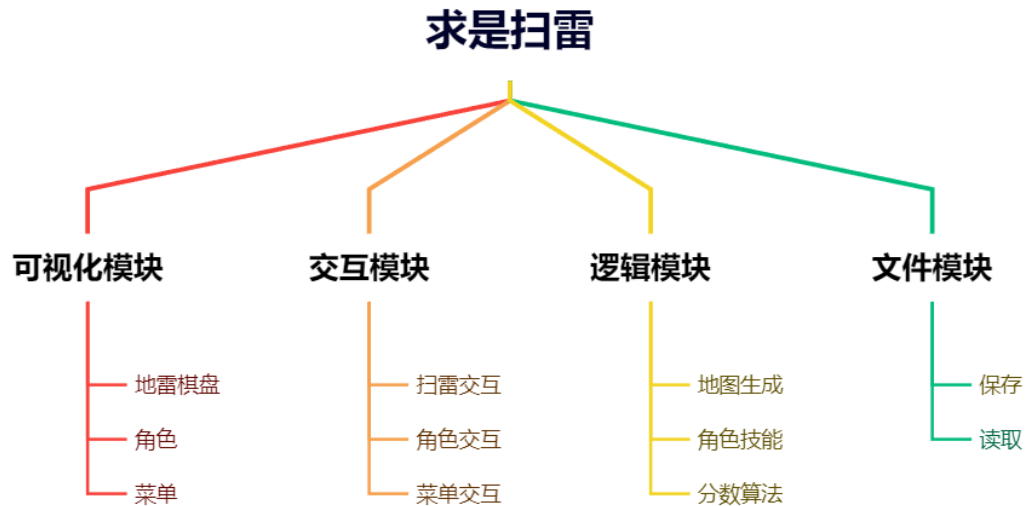
应在 Visual Studio 环境下编译，正常编译运行。

### 2.4.4 运行要求

应达到基本流畅，正常情况图形界面不闪烁、不闪退，能够正常保存和读取文件。

## 3 程序开发设计

### 3.1 总体架构设计



### 3.2 功能模块设计

#### 3.2.1 可视化模块

##### I. 地雷棋盘

地雷棋盘利用 `libgraphics` 库函数和本组成员自己编辑的逻辑函数实现。首先在 `DrawGrid` 函数中实现画单个地雷格子，再在 `DrawChessBoard` 函数中实现画出  $16 \times 16$  的地雷棋盘。旗子及地雷爆炸功能中，旗子利用 `lib - graphics` 库函数实现，并根据角色技能绘制了白色旗子和红色旗子。地雷爆炸图形通过 GeoGebra 几何图形计算器对每个点坐标进行计算，由计算数据绘制出地雷爆炸图像，实现其功能。

##### II. 菜单

菜单由库函数中提供的 `button` 和 `menulist` 两函数实现，并根据交互要求在“更多”按钮中设置了第二级菜单，供玩家选择。在菜单中，实现了音乐的播放与关闭和开发者名单、排行榜的弹窗显示。

##### III. 角色及背景

角色采用贴图的方法，导入了三个游戏角色图形及其装饰框，并加入了背景图案，背景图中我们贴心设计了艺术字体和校徽标志，彰显“求是”二字。

### 3.2.2 交互模块

#### I. 扫雷交互

- a. 鼠标交互：使用鼠标回调，实时返回鼠标位置与按键信息，实现在棋盘区左键掀开，右键插旗。
- b. 键盘交互：使用键盘回调，实时返回键盘按键信息，实现 Esc 键退出游戏，Ctrl+O 键打开种子，Tab 键确定种子值，Enter 重新开始游戏，E 释放技能。
- c. 时间交互：使用时间回调，通过不同计时器，做到游戏时间 1s 刷新一次，游戏画面 0.02s 刷新一次。

#### II. 角色交互

- a. 鼠标左键点击角色旁的“确定”按钮，通过存储改操作返回值确定角色。

#### III. 菜单交互

- a. 鼠标左键点击菜单区各个按钮，通过处理存储该操作返回值进行不同的操作。

### 3.2.3 逻辑模块

#### I. 地图生成

- a. 初始化地图
- b. 随机地雷：Default 方式、种子值方式
- c. 地图雷数分布图：遍历雷图
- d. 翻格子：递归算法
- e. 初始翻开域

\*扫雷地图的数据存储见 3.3

#### II. 角色技能

- a. 02 (ZeroTwo) 技能：结算分数\*2.5。



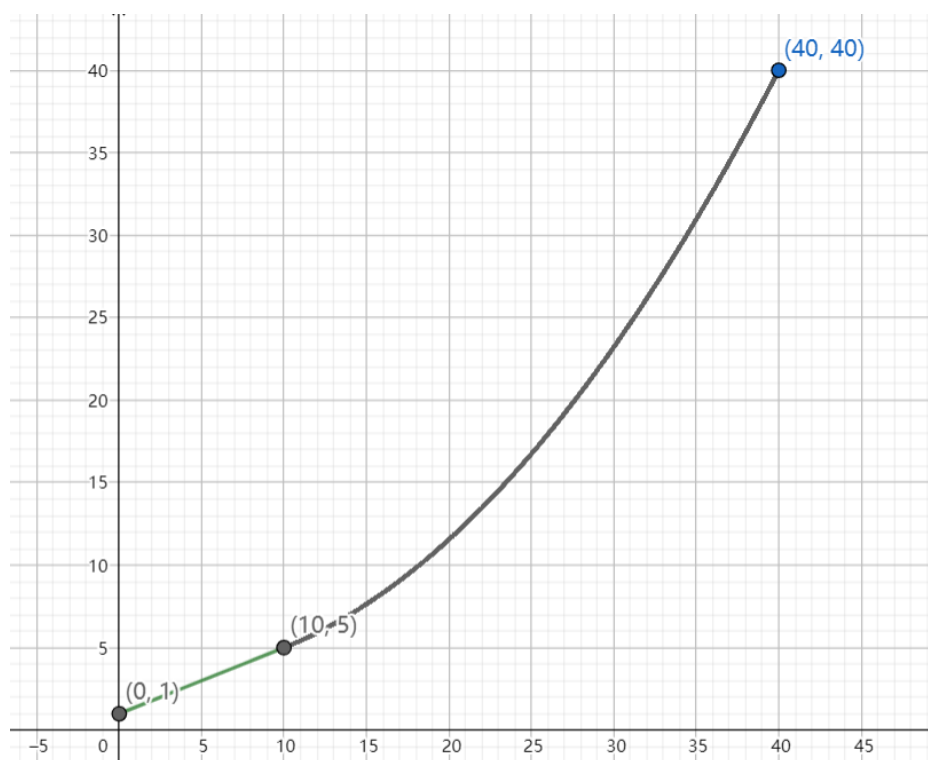
- b. 翠星石（Cui）技能：以鼠标所在方格为中心，翻开 5\*5 区域，自动插旗。
- c. 阿尔贝德（Albedo）技能：无条件复活一次，且在死亡地方插上白旗。

\*使用 `Passive_Character()`，Character 可以是 02、Cui 和 Albedo。

### III. 分数算法

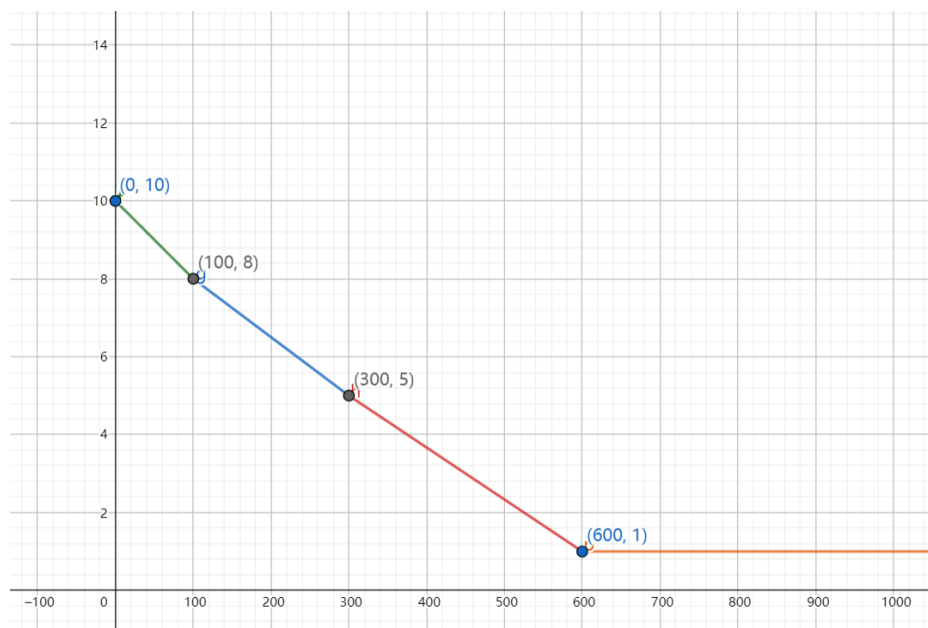
- a. 分数因子
- b. 数值拟合曲线：

正确插旗数-分数倍率曲线，正确插旗数在 0-10 区间时（图中绿色部分）为缓慢的线性增长，0-40 区间为爆炸式的二次增长，目的在于鼓励玩家插旗、保证游戏越到后期分数越高。



正确插旗数-分数倍率图

单局游玩时长-分数倍率曲线：采用分段线性降低曲线，分段逐渐降低绝对斜率，增强玩家体验，对新手较为友好。



单局游玩时长-分数倍率图

### 3.2.4 文件模块

#### I. 更新文件

在读取排行榜文本和写入的过程中，使用单向链表过渡。运用 `atoi()` 函数将 `list.txt` 内的成绩字符串转成整形数，再写入链表节点，读取结束 `fclose` 关于 `list.txt` 的文件流。遍历链表，在正确位置插入本局新的成绩节点。新建关于 `roam.txt` 的文件流，写入模式，将链表成绩用 `itoa()` 函数一一转化成字符串，`fput` 之。最后关闭 `roam.txt` 的文件流，用 `remove()` 函数删除原有的 `list.txt`，用 `rename()` 函数将 `roam.txt` 改名为 `list.txt`，是为覆盖过程。

#### II. 读取文件

- a. 能打开 `list.txt` 时，一行行打印成绩进 `String`，直到 `list.txt` 的尽头或读取了十个成绩。再将 `String` 用弹窗形式显示出来。实际操作比纸面描述复杂得多，涉及动态分配内存、格式化输入等等过程。
- b. 不能打开 `list.txt` 时，跳出错误弹窗，提醒“没有成绩，请至少完成一局”。

### 3.3 数据结构设计

本组选择的数据结构是顺序表和单向链表。

#### 3.3.1 顺序表简介

顺序表是在计算机内存中以数组的形式保存的线性表，线性表的顺序存储是指用一组地址连续的存储单元依次存储线性表中的各个元素、使得线性表中在逻辑结构上相邻的数据元素存储在相邻的物理存储单元中，即通过数据元素物理存储的相邻关系来反映数据元素之间逻辑上的相邻关系，采用顺序存储结构的线性表通常称为顺序表。顺序表是将表中的结点依次存放在计算机内存中一组地址连续的存储单元中。

#### 3.3.2 代码文件内顺序表结构设计

扫雷地图基本信息在代码文件内使用顺序表存储：

```
typedef struct {
    bool ifMine;    //地雷信息
    int  num;       //周围八格的雷数
    bool ifCovered; //格子是否被掀开
    bool flag;      //格子是否插旗
} MINE;

MINE Mine[MAXROW][MAXCOL];
```

每局游戏生成一个Mine[MAXROW][MAXCOL]，先进行初始化，再进行设雷、计算等操作，在游戏过程中，插旗、打开格子等交互也需要用到。

考虑游戏难度的设置，将MAXROW、MAXCOL设置成了16。

扫雷游戏相关信息在代码文件内使用顺序表存储：

```
typedef struct
{
    int gametime;    //记录游戏时间
    int minenum;     //记录已被标记的雷数，即已插旗数
    int mapnum;      //记录当前地图所在操作
    int seednum;     //记录地图种子数值
    bool ifSure;     //是否确定该种子
    bool ifSeed;     //是否开始输入种子
} GAMESTATE;

GAMESTATE Gamestate;    //记录游戏信息
```

每局游戏开始时初始化Gamestate结构体的各种信息，通过游戏中的各种交互，做到记录并显示时间、剩余雷数、当前地图种子数值，记录并根据mapnum

的值做到启动和关闭部分交互，控制种子输入与否以及种子的确定。

扫雷游戏过程中鼠标信息在代码文件中使用顺序表存储：

```
typedef struct {
    double mousex;           //记录鼠标当前所在位置的 x 坐标
    double mousey;           //记录鼠标当前所在位置的 y 坐标
    double CurrentX;
    double CurrentY;
} MOUSESTATE;
static MOUSESTATE Mousestate; //用于判断鼠标状态
```

游戏中利用鼠标回调函数，实时回调信息，在游戏过程中，插旗、打开格子，选择角色等交互需要用到。

### 3.3.3 链表简介

链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。相比于线性表顺序结构，操作复杂。由于不必须按顺序存储，链表在插入的时候可以达到 $O(1)$ 的复杂度，比另一种线性表顺序表快得多，但是查找一个节点或者访问特定编号的节点则需要 $O(n)$ 的时间，而线性表和顺序表相应的时间复杂度分别是 $O(\log n)$ 和 $O(1)$ 。

### 3.3.4 数据储存文件设计

求是扫雷本身的定位仍是单机游戏，没有多个用户，因此排行榜文件简化为仅由分数单列构成。每局游戏的分数独占一行。

### 3.3.5 代码文件内链表结构设计

代码文件内使用单向链表存储数据，用于排行榜分数更新的中间过程。其过程大致为打开->读取->存储到链表->创建新文件->写入->覆盖。

为了减少代码冗余，我们并没有设计链表头文件，而是借用了libgraphics里的linkedlist.h，将linkedlistCDT修改如下：

```
struct linkedlistCDT {
    int dataptr;
    struct linkedlistCDT *next;
};
```

其中dataptr指代分数。

### 3.4 源代码文件组织设计

<文件目录结构>

1) 文件函数结构

文件名	功能简介
logic.c	逻辑模块和文件模块的函数
logic.h	逻辑模块、文件模块函数声明
Init.c	资源载入、游戏初始化以及交互相关的函数
Init.h	游戏初始化以及交互相关函数声明
source	游戏需要的本地资源，包括 BGM、位图等，不一一举例
game.c	游戏运行时与 WinMain 接入之处，也是各个库整合运用之处

2) 多文件构成机制

方便起见，将下列文件记为**标准包含**：

```
#include "graphics.h"
#include "extgraph.h"
#include "genlib.h"
#include "simpio.h"
#include "imgui.h"
#include "conio.h"
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
```

```
#include <windows.h>
#include <olectl.h>
```

```
#include <mmsystem.h>
#include <wingdi.h>
#include <ole2.h>
#include <ocidl.h>
#include <winuser.h>
```

文件名	包含文件
logic.c	<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;math.h&gt;  #include "logic.h" #include "random.h" #include "boolean.h" #include "linkedlist.h" #include "time.h"</pre>
logic.h	<pre>#include "random.h" #include "boolean.h" #include "time.h"</pre>
Init.c	<pre>标准包含 #pragma comment(lib,"winmm.lib") #include "logic.h" #include "Init.h"</pre>
Init.h	<pre>标准包含 #pragma comment(lib,"winmm.lib") #include "logic.h"</pre>
source	<pre>02.bmp Albedo.bmp 翠星石.bmp txtbg02.bmp</pre>

	txtbgCui.bmp txtbgAl.bmp 边框.bmp Title.bmp minesweeper.ico bkmusic.wav boom.wav
game.c	<pre>#include&lt;stdio.h&gt; #include "logic.h" #include"Init.h"</pre>

### 3.5 函数设计描述

logic.c

序号	具体内容
1	<p>函数原型：void CreateMine();</p> <p>功能描述：初始化雷图。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：遍历结构数组 Mine，赋值使雷层全 FLASE（0）分布，周围雷数为 0，初始化所有格子被盖住（FALSE），所有格子未被插旗（FALSE）。</p>
2	<p>函数原型：void RandomMine();</p> <p>功能描述：纯随机出雷（与种子值随机相比），把这次的种子值赋给全局变量 Seed。</p> <p>参数描述： 无</p>

	<p>返回值描述：无</p> <p>重要局部变量定义：int index,cnt</p> <p>重要局部变量用途描述：index 为随机出的一维坐标；cnt 是成功设雷的计数</p> <p>函数算法描述：把 time() 值传入全局变量 Seed，再根据 Seed 用 srand() 产生伪随机序列。在 cnt&lt;=40 的情况下，用 rand() 函数得到 index 值，将 index 转化为二维坐标 (row,col)，Mine[row][col].ifMine（如果是 FALSE 的话）改为 TRUE，cnt++，反之，continue。</p>
3	<p>函数原型：void SeedRand(unsigned int seed);</p> <p>功能描述：按种子值随机出雷</p> <p>参数描述：seed，种子值</p> <p>返回值描述：无</p> <p>重要局部变量定义：同 2</p> <p>重要局部变量用途描述：同 2</p> <p>函数算法描述：把 seed 值传入 srand() 产生伪随机序列。其他同 2。</p>
4	<p>函数原型：int count(int row, int col);</p> <p>功能描述：计算某点周围八格的雷数</p> <p>参数描述：row, 该点行坐标；col, 该点列坐标</p> <p>返回值描述：该点周围八格的雷数</p> <p>重要局部变量定义：int num</p> <p>重要局部变量用途描述：该点周围八格的雷数</p> <p>函数算法描述：分九种情况，计算 (row,col) 周围格子的 ifMine==TRUE 的个数，用 num 代表，并返回。</p>
5	<p>函数原型：void Clear(int row, int col);</p> <p>功能描述：形成打开格子的后端结果</p> <p>参数描述：行列坐标</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p>



	<p>重要局部变量用途描述：无</p> <p>函数算法描述：递归算法。在该点不是雷且没被打开、没被插旗的前提下，如果 num 是 0，ifCovered 修改为 TRUE，分九种情况接着做上下左右四个方向的 Clear；如果 num 非 0，ifCovered 修改为 TRUE, 出口。</p>
6	<p>函数原型：void FirstSquire();</p> <p>功能描述：纯随机时，代替用户做第一次 Clear，保证游玩可重复性。</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：用对应全局变量 Seed，用 srand() 产生伪随机序列，在 while(1) 无限循环下直到该点不是雷、num 是 0，做该点的 Clear。</p>
7	<p>函数原型：void SeedFirst(unsigned int seed);</p> <p>功能描述：种子值随机时，代替用户做第一次 Clear，保证游玩可重复性。</p> <p>参数描述：seed，种子值</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：用传入的 seed，其他同 6。</p>
8	<p>函数原型：int FinalScore(int gametime);</p> <p>功能描述：得出最终游玩分数</p> <p>参数描述：gametime，单局时长</p> <p>返回值描述：最终游玩分数</p> <p>重要局部变量定义：int RightFlag</p> <p>重要局部变量用途描述：该局游戏正确插旗数</p> <p>函数算法描述：遍历结构数组 Mine，求出正确插旗数 RightFlag，</p>

	再根据传入的 gametime 和 RightFlag 按得分曲线算出最终得分。
9	<p>函数原型: void UpdateList(int score);</p> <p>功能描述: 一局游戏后, 更新排行榜</p> <p>参数描述: score, 最终游玩分数</p> <p>返回值描述: 无</p> <p>重要局部变量定义: FILE *fp, *fpnew</p> <p>重要局部变量用途描述: list.txt 的文件流, roam.txt 的文件流</p> <p>函数算法描述: 打开-&gt;读取-&gt;存储到链表-&gt;创建新文件-&gt;写入-&gt;覆盖 (删除+改名)。</p>
10	<p>函数原型: void Passive_Cui(int row, int col);</p> <p>功能描述: 以某点为中心, 在后端翻开 5*5 区域, 自动插旗。</p> <p>参数描述: 行列坐标</p> <p>重要局部变量定义: 无</p> <p>重要局部变量用途描述: 无</p> <p>函数算法描述: 算法与 Count 相近。</p>

## Init.c

序号	具体内容
1	<p>函数原型: void DrawBackground();</p> <p>功能描述: 插入游戏背景</p> <p>参数描述: 无</p> <p>返回值描述: 无</p> <p>重要局部变量定义: 无</p> <p>重要局部变量用途描述: 无</p> <p>函数算法描述: 算法与 Count 相近无特殊算法, 会者不难。</p>
2	<p>函数原型: void DrawFrame();</p> <p>功能描述: 插入角色边框</p> <p>参数描述: 无</p> <p>返回值描述: 无</p> <p>重要局部变量定义: 无</p>

	<p>重要局部变量用途描述：无</p> <p>函数算法描述：算法与 Count 相近同 1。</p>
3	<p>函数原型：void Draw_ZeroTwo(bool flag);</p> <p>功能描述：插入角色零二</p> <p>参数描述：鼠标是否在零二的区域</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：同 1。</p>
4	<p>函数原型：void Draw_Cui(bool flag);</p> <p>功能描述：插入角色翠星石</p> <p>参数描述：鼠标是否在翠星石的区域</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：同 1。</p>
5	<p>函数原型：void Draw_Albedo(bool flag);</p> <p>功能描述：插入角色雅儿贝德</p> <p>参数描述：鼠标是否在雅儿贝德的区域</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：同 1。</p>
6	<p>函数原型：void DwtxtBg02();</p> <p>功能描述：插入零二技能栏</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p>

	函数算法描述：同 1。
7	<p>函数原型：void DwtxtBgCui ();</p> <p>功能描述：插入翠星石技能栏</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：同 1。</p>
8	<p>函数原型：void DwtxtBgAl ();</p> <p>功能描述：插入雅儿贝德技能栏</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：同 1。</p>
9	<p>函数原型：void CrtItat ();</p> <p>功能描述：选择角色时做对应的交互</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：int current_character;</p> <p>重要局部变量用途描述：鼠标所在位置对应的角色</p> <p>函数算法描述：没有选择角色时，鼠标所在的角色位置放大；选择角色后，单一显示所选角色。</p>
9	<p>函数原型：GameOver ();</p> <p>功能描述：游戏失败时弹窗及爆炸音效。</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p>

	函数算法描述：使用 mciSendString 函数实现播放地雷爆炸音效，并调用分数、排行榜函数实现弹窗中分数的计算和显示。
10	<p>函数原型：Win ()；</p> <p>功能描述：游戏胜利时实现弹窗。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 游戏胜利时调用分数、排行榜函数实现弹窗中游分数的计算和显示。</p>
11	<p>函数原型：DrawWhiteFlag ()；</p> <p>功能描述：在指定区域画白色旗子。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 无特殊算法</p>
12	<p>函数原型：void AddBGM()；</p> <p>功能描述：播放背景音乐。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 使用 PlaySound 函数实现播放背景音乐</p>
13	<p>函数原型：GetMueseButton (int x, int y, int button, int event)；</p> <p>功能描述：获得鼠标当前状态并判断其状态。</p> <p>参数描述： x, y 为鼠标位置，button 为鼠标状态，event 为鼠标事件。</p>

	<p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量定义：int row, col</p> <p>重要局部变量用途描述：将鼠标坐标转化为棋盘上的行列坐标</p> <p>函数算法描述：利用 switch 语句和 if 语句对鼠标在不同情况下的不同操作进行相应的响应。</p>
14	<p>函数原型：GetCurrentPlace();</p> <p>功能描述：得到鼠标当前位置，协助画地雷格子。</p> <p>参数描述：无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：无特殊算法</p>
15	<p>函数原型：DrawMine (double x, double y);</p> <p>功能描述：在地雷格子中画出地雷爆炸图像。</p> <p>参数描述：x, y 为画出地雷爆炸图像的起始位置</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：无特殊算法</p>
16	<p>函数原型：DrawFlag (double x, double y);</p> <p>功能描述：在指定区域画红色旗子。</p> <p>参数描述：x, y 为画旗子的起始位置</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：无特殊算法</p>
17	<p>函数原型：TurnLight (double x, double y);</p> <p>功能描述：使指定区域变亮（地雷格子内）。</p>

	<p>参数描述: x, y 为变亮区域</p> <p>返回值描述: 无</p> <p>重要局部变量定义: 无</p> <p>重要局部变量用途描述: 无</p> <p>函数算法描述: 无特殊算法</p>
18	<p>函数原型: DrawGrid ();</p> <p>功能描述: 画地雷格子。</p> <p>参数描述: 无</p> <p>返回值描述: 无</p> <p>重要局部变量定义: 无</p> <p>重要局部变量用途描述: 无</p> <p>函数算法描述: 无特殊算法</p>
19	<p>函数原型: DrawChessBoard ();</p> <p>功能描述: 画地雷棋盘。</p> <p>参数描述: 无</p> <p>返回值描述: 无</p> <p>重要局部变量定义: 无</p> <p>重要局部变量用途描述: 无</p> <p>函数算法描述: 使用双重循环并使用 DrawGrid 函数画出 16*16 棋盘, 并根据鼠标状态判断是否在格子上插旗, 判断格子是否被翻开以及翻开后有无雷并对之进行相应的处理。</p>
20	<p>函数原型: DrawMenu ();</p> <p>功能描述: 画出菜单并实现菜单相关功能。</p> <p>参数描述: 无</p> <p>返回值描述: 无</p> <p>重要局部变量定义: 无</p> <p>重要局部变量用途描述: 无</p> <p>函数算法描述: 三个 button 函数实现前打开种子、开始游戏、退出游戏功能。最后一个 menulist 函数实现二级菜单。</p>

21	<p>函数原型：Rank ();</p> <p>功能描述：实现排行榜功能。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：char* String;</p> <p>重要局部变量用途描述：存储 list.txt 的所有成绩</p> <p>函数算法描述：当能打开 list.txt 时，每读一行对 String 用 realloc() 重新分配内存，用 sprintf 格式化输入该行的序号 “[序号]”，再用 strcat 把成绩字符串拼接进 String，在弹窗中打印 String；如果不能打开 list.txt，将跳出错误弹窗，显示“没有成绩，请至少完成一局游戏!”。</p>
22	<p>函数原型：TimerEventProcess (int timerID);</p> <p>功能描述：判断游戏开始。</p> <p>参数描述： timerID 用于注册</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：利用 switch 函数对不同计时器进行不同的操作</p>
23	<p>函数原型：Renew ();</p> <p>功能描述：刷新界面重新开始游戏。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 无特殊算法</p>
24	<p>函数原型：KeyboardEventProcess(int key, int event);</p> <p>功能描述：对在不同情况下键盘的不同操作做出相应的反应，如退出游戏、重新开始、打开/清空种子、输入种子。</p> <p>参数描述： 无</p>



	<p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：通过 switch 函数和结构体中的变量进行判断和响应。</p>
25	<p>函数原型：MouseEventProcess(int x, int y, int button, int event);</p> <p>功能描述：回调鼠标函数。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 无特殊算法</p>
25	<p>函数原型：CreateString();</p> <p>功能描述：输出地雷下的数字。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述：利用两重循环实现对每个格子的遍历并通过 itoa 函数转化。</p>
26	<p>函数原型：Display();</p> <p>功能描述：显示游戏主界面。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：</p> <p>char num[600],minecurrent[600],yourseed[600];</p> <p>重要局部变量用途描述：记录转化而来的游戏时间、剩余雷数、地图种子值。</p> <p>函数算法描述： 无特殊算法</p>

27	<p>函数原型：CreateMap();</p> <p>功能描述：链接地雷棋盘和地雷，随机在格子下生成地雷并在雷旁产生数字。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 无特殊算法</p>
28	<p>函数原型：UseSkill();</p> <p>功能描述：实现翠星石的技能。</p> <p>参数描述： 无</p> <p>返回值描述：无</p> <p>重要局部变量定义：无</p> <p>重要局部变量用途描述：无</p> <p>函数算法描述： 无特殊算法</p>

## 4 部署运行和使用说明

请使用 Visual Studio 进行部署运行。以 VS2022 为例。

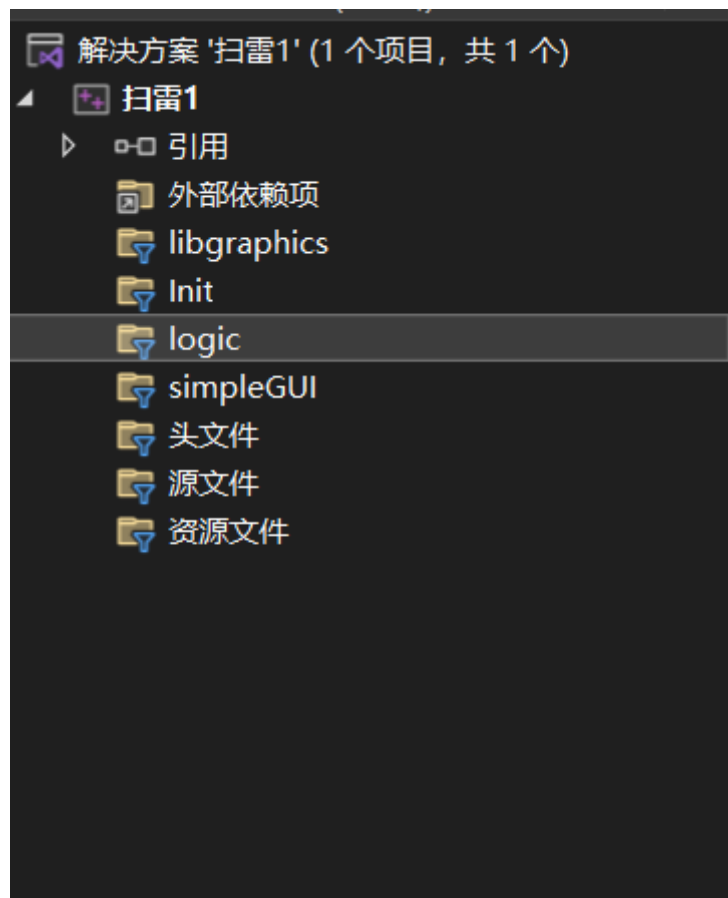
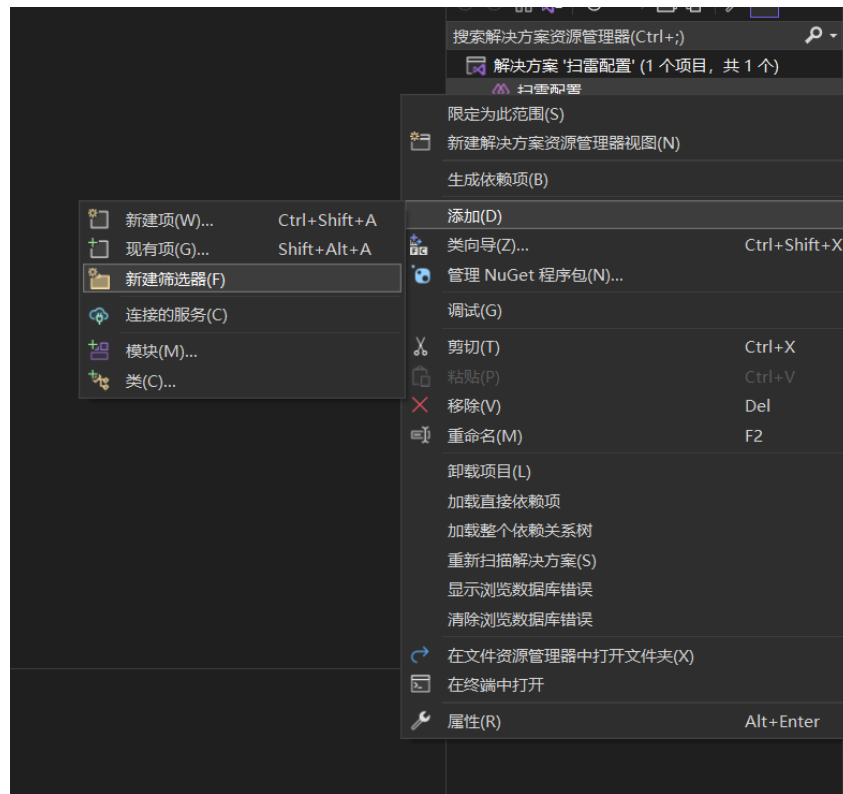
### 4.1 编译安装

- 1) 解压扫雷.zip 得到扫雷文件夹
- 2) 打开 VS2022，在开始使用栏点击“创建新项目”->空项目->改上合适的名字（教程取名为扫雷 1）和存储地址（建议直接建在刚解压的文件夹里）->创建，打开扫雷 1 文件夹，将解压得到的文件夹都拖入其中的扫雷 1 文件夹。



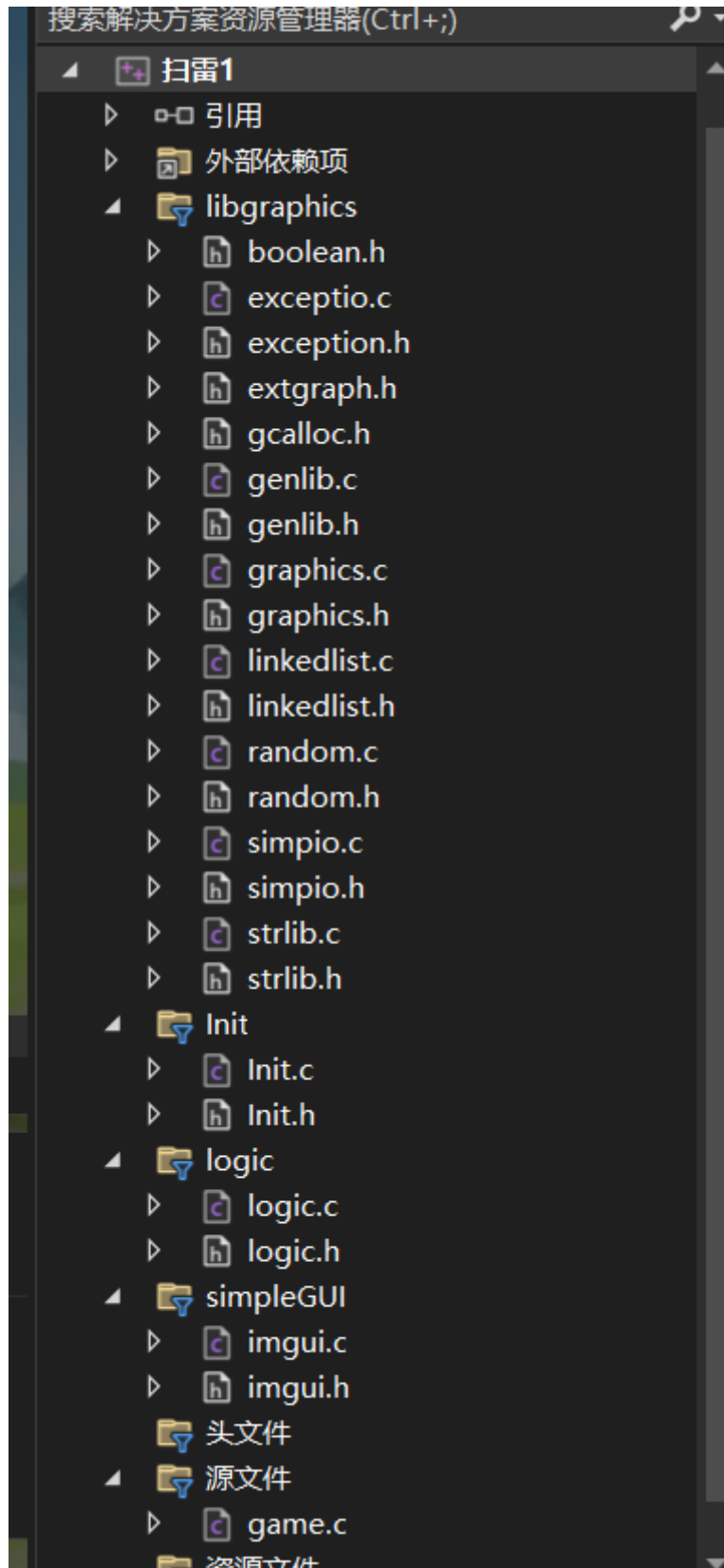
<div> <div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div> </div> <div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div>				
<div> <div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div> </div> <div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div>				
名称	修改日期	类型	大小	
Init	2022/6/1 19:11	文件夹		
libgraphics	2022/6/1 19:11	文件夹		
logic	2022/6/1 19:11	文件夹		
simpleGUI	2022/6/1 19:11	文件夹		
source	2022/6/1 19:11	文件夹		
game.c	2022/5/31 22:22	C Source	1 KB	
README.txt	2022/6/1 1:07	文本文档	1 KB	
扫雷1.vcxproj	2022/6/1 21:38	VC++ Project	7 KB	
扫雷1.vcxproj.filters	2022/6/1 21:38	VC++ Project Fil...	1 KB	
扫雷1.vcxproj.user	2022/6/1 21:38	Per-User Project...	1 KB	

3) 打开解决方案资源管理器，右键，添加->新建筛选器->建立四个筛选器。



4) 右键扫雷 1->添加->现有项->game.c, 对四个筛选器也做一样的操作, 导入相

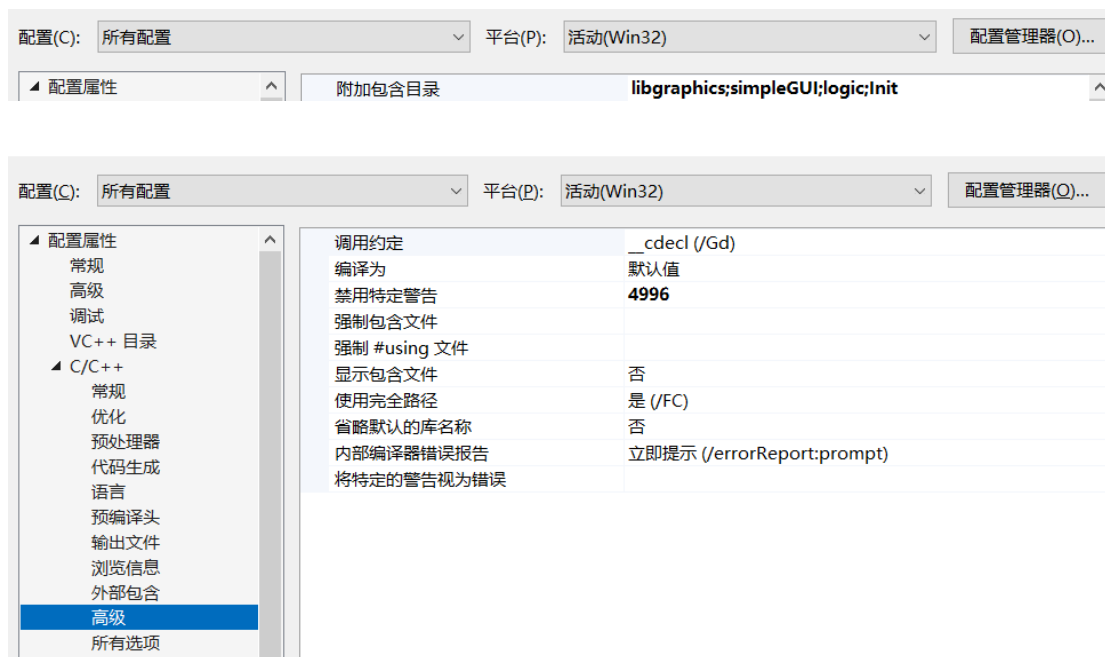
同文件名内的所有文件



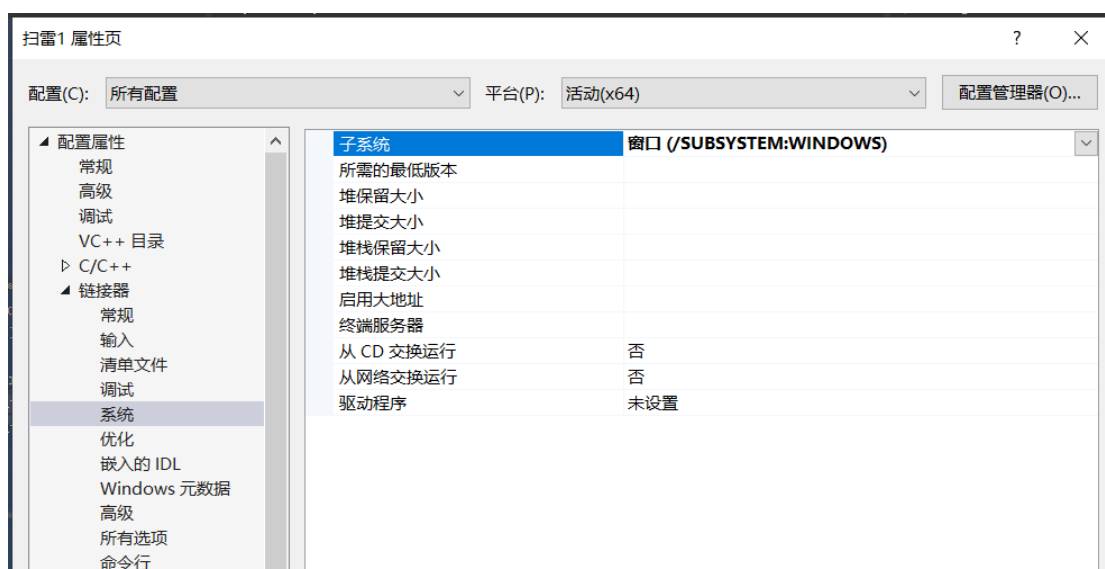
5) 在下图位置改成 x86



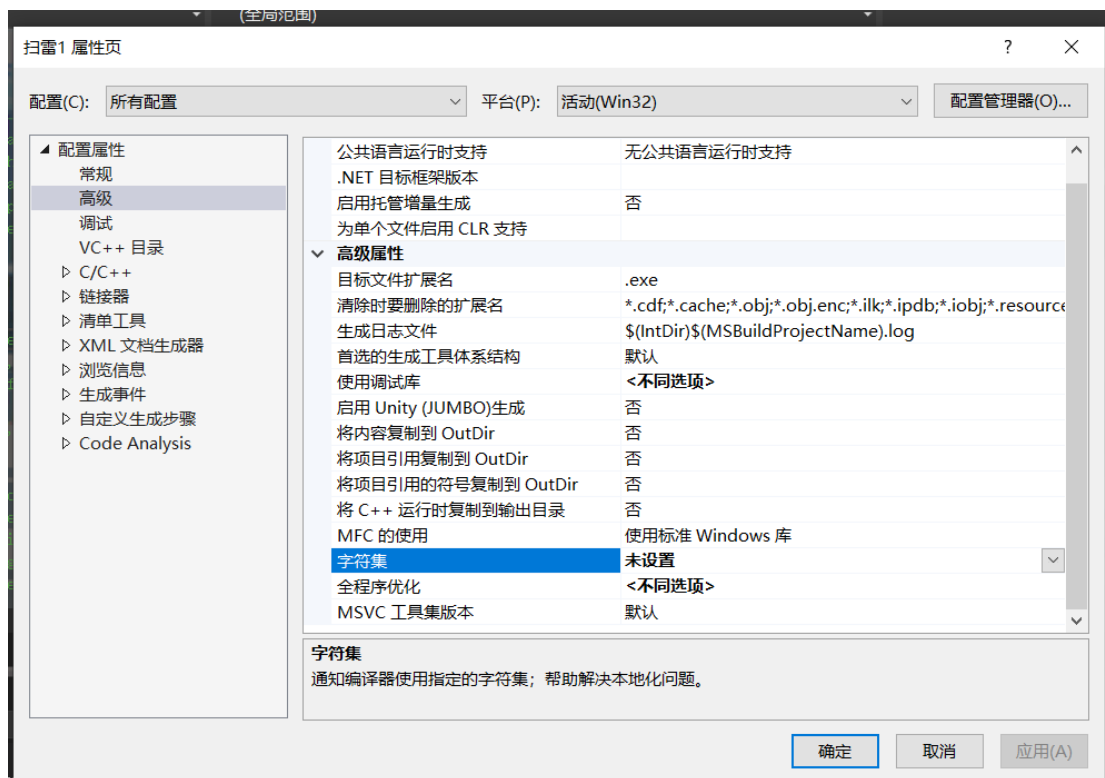
- 5) 扫雷 1 右键->属性->C/C++->附加包含目录->配置改为“所有配置”->输入“libgraphics;Init;logic;simpleGUI”->高级->禁用特定警告->输入“4996”



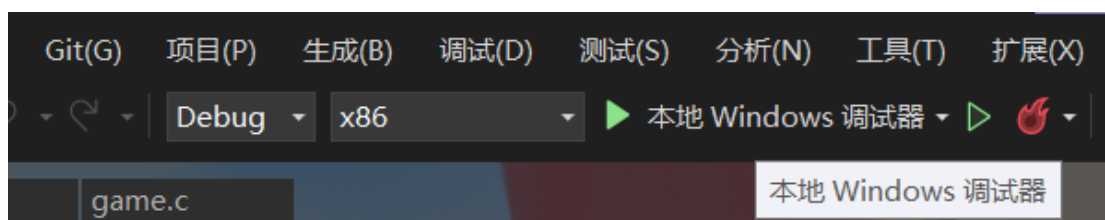
- 6) 扫雷 1 右键->属性->链接器->子系统，改为窗口。



- 7) 配置属性->高级->字符集->未设置，确定。



8) 点击本地 Windows 调试器，编译运行。



9) 生成可执行文件：生成->生成扫雷 1，复制 source 文件夹进 Debug 文件夹，点击扫雷 1.exe，可以开始游玩。



## 4.2 运行测试

### 测试案例一

文件: logic.c

测试内容: 测试地图后端信息生成是否有误。

测试结果: 无误。

```
using the same seed = 1652666202
in a:
000001000000000000
000001100000010000
000010001000010000
000100000011001000
000100000010000000
000000000000000100
000010001000001001
000010000000000000
000001000000010000
000001010000000000
01000001100010001000
000010000000010000
011000000100000010
010000000100010000
010000000000000000
000100000000000001

in b:
000001000000000000
000001100000010000
000010001000010000
000100000011001000
000100000010000000
000000000000000100
000010001000001001
000010000000010000
000001000000010000
000001010000000000
01000001100010001000
000010000000010000
011000000100000010
010000000100010000
010000000000000000
000100000000000001
```

按照种子值生成地雷图，完全一致。

```
wh 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

后端生成是否翻开，1 代表未翻开，0 代表初始翻开域，成功。

测试代码:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```



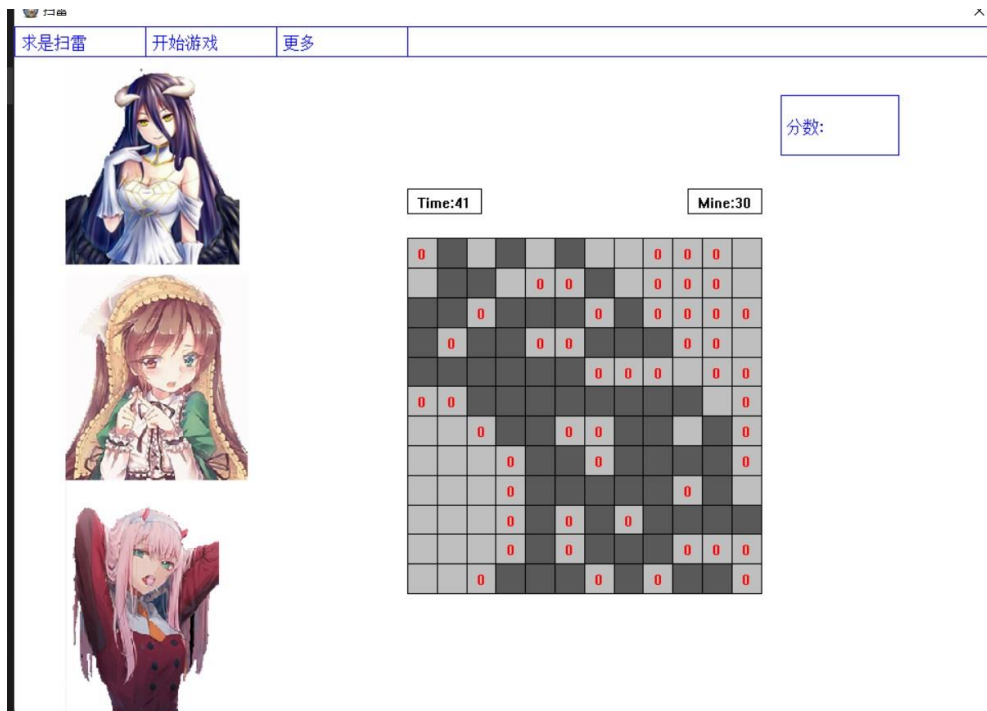
```
#include <boolean.h>
#include "logic.h"

int main() {
    CreateMine();
    RandomMine();
    int i, j;
    for (i = 0; i < MAXROW; i++) {
        for (j = 0; j < MAXCOL; j++) {
            Mine[i][j].num = Count(i, j);
        }
    }
    for (i = 0; i < MAXROW; i++) {
        for (j = 0; j < MAXCOL; j++) {
            printf("%d ", Mine[i][j].ifMine);
        }
        printf("\n");
    }
    printf("\n\n");
    FisrtSquare();
    for (i = 0; i < MAXROW; i++) {
        for (j = 0; j < MAXCOL; j++) {
            printf("%d ", !Mine[i][j].ifCovered);
        }
        printf("\n");
    }
}
```

## 测试案例二

文件: Init.c

测试错误: 数字层前端显示错误



**原因总结：**因为生成雷层和数字层在刚写完函数时就已经测试通过，所以推测是输出出了问题，经过测试发现，问题在于数字层的三种表现形式上。由于对接成员想法各异加之必要的代码实现，原代码的数字层有 `int`、`char` 和 `char*` 三种形式，`int` 是后端计算出来的，`char` 用于判断是否是 ‘0’，`char*` 作为字符串用于打印。赋值时三种形式的运用出现了混乱，将 `char*` 直接赋值导致赋值失败，`char*` 仍然是 “0”，也就造成了显示的数字都是 “0”。

#### 修改方案：

##### 1. 聚焦输出原代码

```
drawBox(x + j * MINESIZE, y - k * MINESIZE - MINESIZE, MINESIZE, MINESIZE, 0, NumToString[k][j], '1', "Red");
```

（`Minenum` 是由 `Mine[k][j].num` 通过计算转化而来的二维数组）

因为要输出的是字符串，所以最终做出两套方案（已测试均可运行）如下。

方案一（采纳方案）：将 `Mine[k][j].num` 转化为三维数组 `NumToString[k][j][20]`。

方案二：将 `Mine[k][j].num` 通过 `itoa` 函数转化为二维数组 `Minenum[i+16*j][20]`。

##### 2. 删去 `char` 的表现形式，判断是否为 0 直接用 `int` 形式。

#### 完整代码：

```
char NumToString[16][16][20];
```

```
void CreateString(void) {
```

```

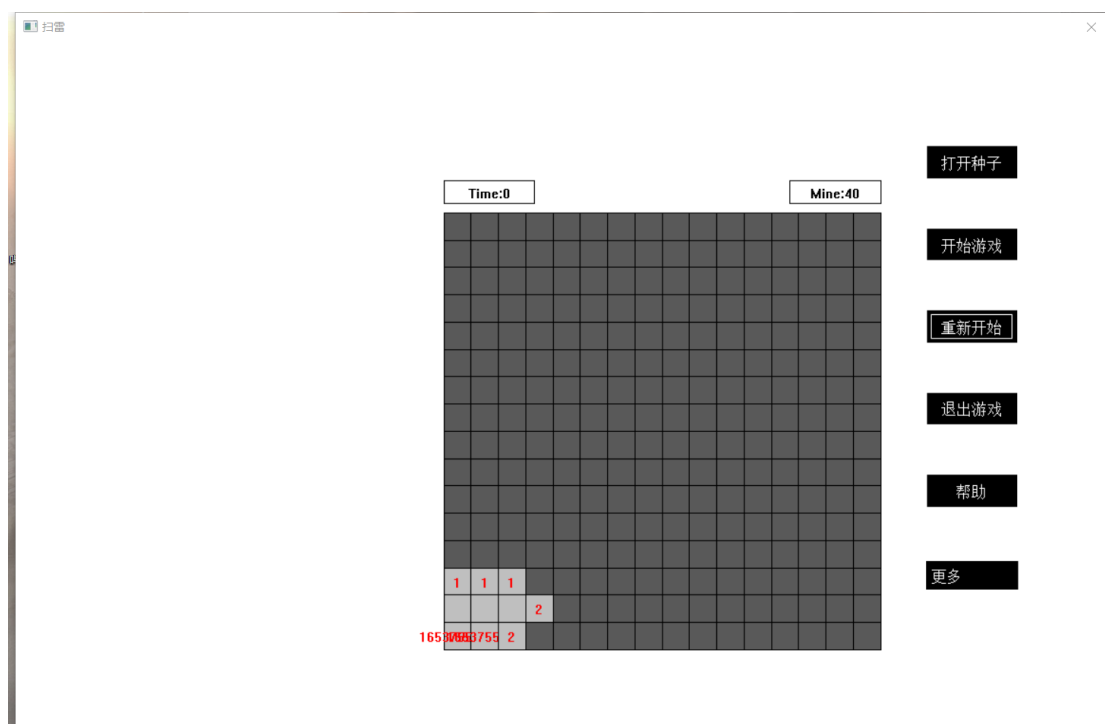
for (int i = 0; i < 16; i++) {
    for (int j = 0; j < 16; j++) {
        itoa(Mine[i][j].num, NumToString[i][j], 10);
    }
}

```

### 测试案例三

文件：Init.c、logic.c

测试错误：数字层异常数字，见左下角。



**原因总结：**通过将与数字层相关的控制显示的函数全部调整，发现不是显示的问题。然后取紊乱处的数字层数据 `Mine[k][j].num`，发现左下角和右下角数字层生成出错。猜测可能是异常的位置未被赋值导致产生了异常数字，查看代码

```

int Count(int row, int col) {
    int num = 0;

    int i, j;

    for (i = -1; i <= 1; i++) {
        if (row + i > MAXROW) break;
        for (j = -1; j <= 1; j++) {
            if (col + j > MAXCOL) break;

```

```

        if (row + i < 0 || col + j < 0) continue;
        else num += Mine[row + i][col + j].ifMine;
    }

    return num;
}

```

num 是经过初始化的，因此如果 for 循环没有赋值成功，也应该返回 0，这个诡异的结果并没有找到很好的解释，可能是程序连接过程中出了事故。

**解决方案：**本来是为了代码的简洁才修改成上述的样子，现今只能改回分类遍历的老手段。

**完整代码：**

```

int Count(int i, int j) {
    int num;
    if (i == 0) {
        if (j == 0)
            num = Mine[i + 1][j].ifMine + Mine[i + 1][j + 1].ifMine + Mine[i][j + 1].ifMine;
        else if (j == MAXCOL - 1)
            num = Mine[i + 1][j].ifMine + Mine[i + 1][j - 1].ifMine + Mine[i][j - 1].ifMine;
        else
            num = Mine[i + 1][j].ifMine + Mine[i + 1][j - 1].ifMine + Mine[i][j - 1].ifMine + Mine[i][j + 1].ifMine + Mine[i + 1][j + 1].ifMine;
    }
    else if (i == MAXROW - 1) {
        if (j == 0) num = Mine[i - 1][j].ifMine + Mine[i - 1][j + 1].ifMine + Mine[i][j + 1].ifMine;
        else if (j == MAXCOL - 1)
            num = Mine[i - 1][j].ifMine + Mine[i - 1][j - 1].ifMine + Mine[i][j - 1].ifMine;
        else
            num = Mine[i - 1][j].ifMine + Mine[i - 1][j - 1].ifMine + Mine[i][j - 1].ifMine + Mine[i][j + 1].ifMine + Mine[i - 1][j + 1].ifMine;
    }
    else if (j == 0) {

```

```

        num = Mine[i - 1][j].ifMine + Mine[i - 1][j + 1].ifMine + Mine[i][j + 1].ifMine + Mine[i + 1][j].ifMine + Mine[i + 1][j + 1].ifMine;
    }
    else if (j == MAXCOL - 1) {

        num = Mine[i - 1][j].ifMine + Mine[i - 1][j - 1].ifMine + Mine[i][j - 1].ifMine + Mine[i + 1][j].ifMine + Mine[i + 1][j - 1].ifMine;
    }
    else {

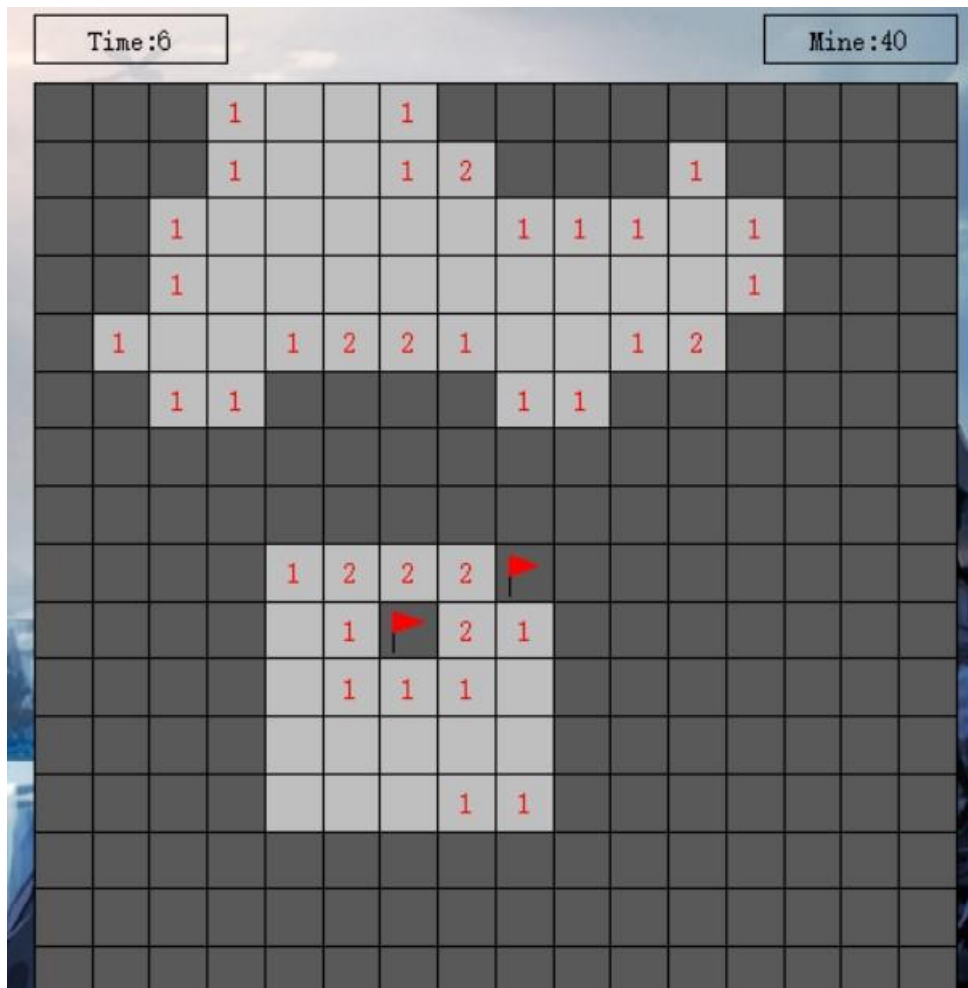
        num = Mine[i - 1][j].ifMine + Mine[i - 1][j - 1].ifMine + Mine[i][j - 1].ifMine + Mine[i + 1][j].ifMine + Mine[i + 1][j - 1].ifMine + Mine[i - 1][j + 1].ifMine + Mine[i][j + 1].ifMine + Mine[i + 1][j + 1].ifMine;
    }
    return num;
}

```

## 测试案例四

文件：Init.c

测试错误：翠星石使用技能时地雷数没有减少



**原因总结：**通过查看与显示和计算剩余地雷数的函数，发现在计算剩余雷数时原方案是右键未掀开的格子雷数-1，右键插旗的格子雷数+1，并没有包含翠星石的技能。

**解决方案：**将计算插旗数转移到 DrawChessBoard 函数中判断画旗子的循环中。

## 测试案例五

文件:graphics.c、Init.c

测试内容：游戏贴图

测试过程：

1.把贴图代码段放在 graphics.c 的 DoUpdate()函数里，代码段如下

```
static void DoUpdate(void)
{
    HDC dc;
```

```

h02 = (HBITMAP)LoadImage(NULL, "../source/02.bmp", IMAGE_BITMAP, 280, 224, LR_LOADFROMFILE);

dc = BeginPaint(graphicsWindow, &ps);

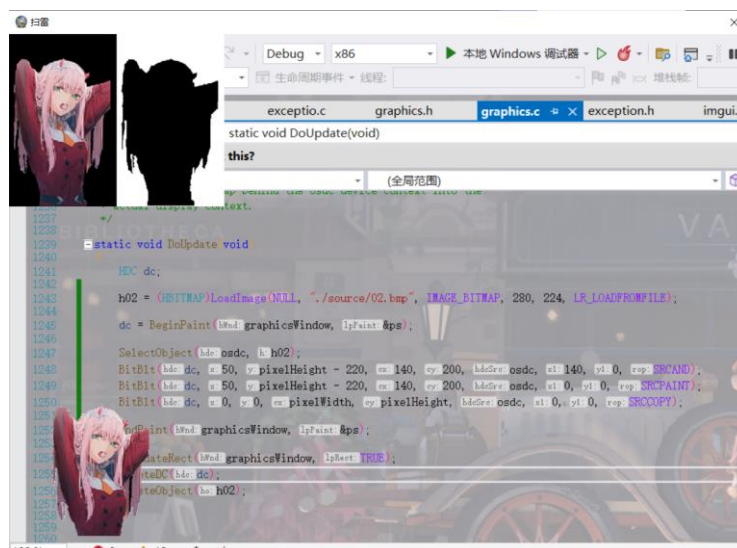
SelectObject(osdc, h02);
BitBlt(dc, 50, pixelHeight - 220, 140, 200, osdc, 140, 0, SRCAND);
BitBlt(dc, 50, pixelHeight - 220, 140, 200, osdc, 0, 0, SRCPAINT);
BitBlt(dc, 0, 0, pixelWidth, pixelHeight, osdc, 0, 0, SRCCOPY);

EndPaint(graphicsWindow, &ps);

ValidateRect(graphicsWindow, TRUE);
DeleteDC(dc);
DeleteObject(h02);
}

```

测试结果如下，原本画的背景变成透明的，左上角出现贴图素材，左下角是正确的贴图结果，推测原因在于屏幕既包含了 gdc 的内容也包含了 osdc 的内容，并且 osdc 的贴图晚于 gdc 的贴图。



2.在 1 的结果上，不在 osdc 里直接选中 h02，而是另起兼容 DC，采用双缓冲技术。代码如下

```

static void DoUpdate(void)
{
    HDC dc;

    h02 = (HBITMAP)LoadImage(NULL, "../source/02.bmp", IMAGE_BITMAP, 280, 224, LR_LOADFROMFILE);
}

```

```

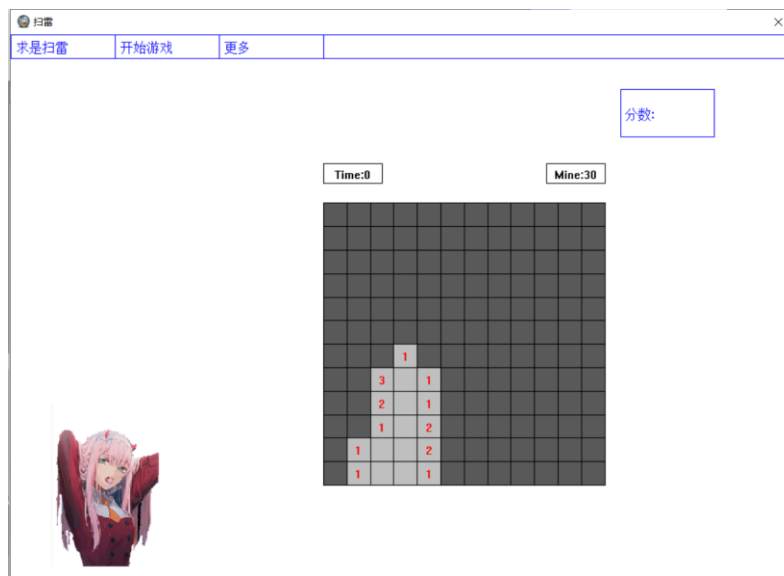
dc = BeginPaint(graphicsWindow, &ps);
mdc = CreateCompatibleDC(dc);

SelectObject(mdc, h02);
BitBlt(osdc, 50, pixelHeight - 220, 140, 200, mdc, 140, 0, SRCAND);
BitBlt(osdc, 50, pixelHeight - 220, 140, 200, mdc, 0, 0, SRCPAINT);
BitBlt(dc, 0, 0, pixelWidth, pixelHeight, osdc, 0, 0, SRCCOPY);

EndPoint(graphicsWindow, &ps);

ValidateRect(graphicsWindow, TRUE);
DeleteDC(dc);
DeleteDC(mdc);
DeleteObject(h02);
}

```



贴图成功。但是仍有图片覆盖高于绘图的问题，见下图。





3.本并不打算解决贴图高于绘图问题，所以在 2 基础上完善贴图。

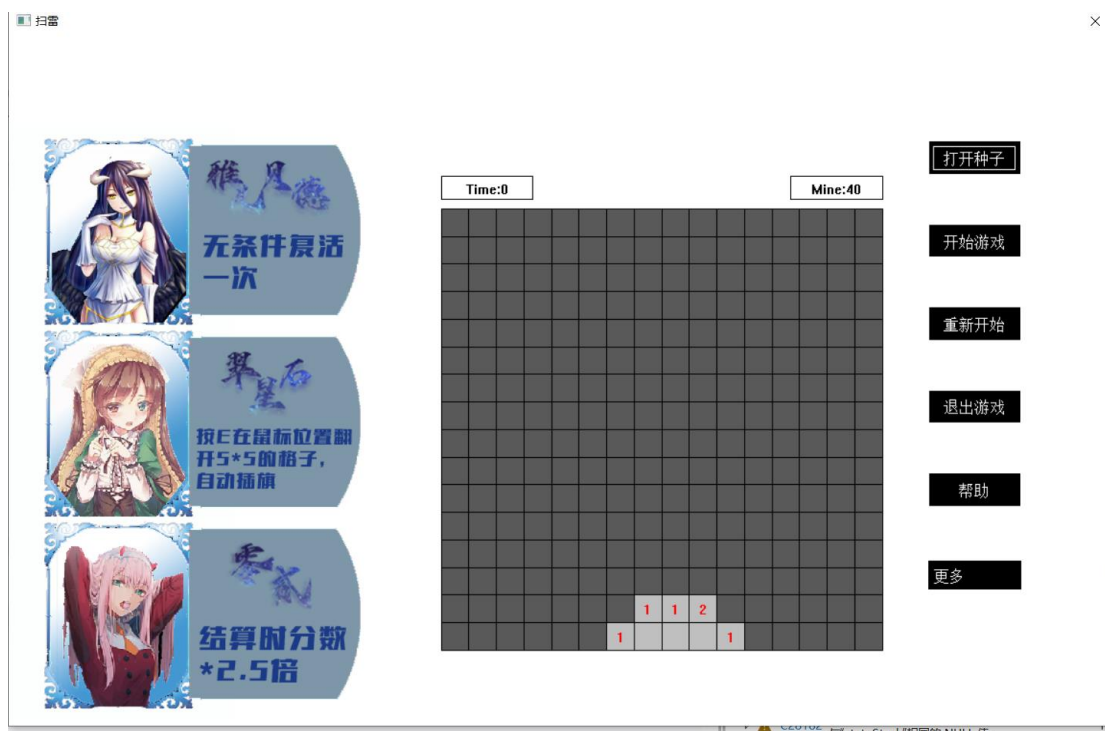
```
static void DoUpdate(void)
{
    HDC dc;
    h02 = (HBITMAP)LoadImage(NULL, "./source/02.bmp", IMAGE_BITMAP, 280, 224, LR_LOADFROMFILE);
    hCui = (HBITMAP)LoadImage(NULL, "./source/翠星石.bmp", IMAGE_BITMAP, 280, 188.5, LR_LOADFROMFILE);
    hAlbedo = (HBITMAP)LoadImage(NULL, "./source/Albedo.bmp", IMAGE_BITMAP, 280, 188.5, LR_LOADFROMFILE);
    hFrame = (HBITMAP)LoadImage(NULL, "./source/边框.bmp", IMAGE_BITMAP, 480, 220, LR_LOADFROMFILE);
    htxtBg_02 = (HBITMAP)LoadImage(NULL, "./source/textbg02.bmp", IMAGE_BITMAP, 510, 260, LR_LOADFROMFILE);
    htxtBg_Cui = (HBITMAP)LoadImage(NULL, "./source/textbgCui.bmp", IMAGE_BITMAP, 510, 260, LR_LOADFROMFILE);
    htxtBg_Al = (HBITMAP)LoadImage(NULL, "./source/textbgAl.bmp", IMAGE_BITMAP, 510, 260, LR_LOADFROMFILE);
    dc = BeginPaint(graphicsWindow, &ps);
    mdc = CreateCompatibleDC(dc);
    mmdc = CreateCompatibleDC(dc);
    bmp = CreateCompatibleBitmap(dc, pixelWidth, pixelHeight);
    SelectObject(mdc, bmp);
    SelectObject(mmdc, hFrame);
    BitBlt(osdc, -5, pixelHeight - 225, 240, 220, mmdc, 240, 0, SRCAND);
    BitBlt(osdc, -5, pixelHeight - 225, 240, 220, mmdc, 0, 0, SRCPAINT);
    BitBlt(osdc, -5, pixelHeight - 425, 240, 220, mmdc, 240, 0, SRCAND);
}
```

```

BitBlt(osdc, -5, pixelHeight - 425, 240, 220, mmdc, 0, 0, SRCPAINT);
BitBlt(osdc, -5, pixelHeight - 625, 240, 220, mmdc, 240, 0, SRCAND);
BitBlt(osdc, -5, pixelHeight - 625, 240, 220, mmdc, 0, 0, SRCPAINT);
SelectObject(mmdc, hCui);
BitBlt(osdc, 44, pixelHeight - 398, 140, 180, mmdc, 140, 0, SRCAND);
BitBlt(osdc, 44, pixelHeight - 398, 140, 180, mmdc, 0, 0, SRCPAINT);
SelectObject(mmdc, hAlbedo);
BitBlt(osdc, 45, pixelHeight - 600, 140, 180, mmdc, 140, 0, SRCAND);
BitBlt(osdc, 45, pixelHeight - 600, 140, 180, mmdc, 0, 0, SRCPAINT);
SelectObject(mmdc, h02);
BitBlt(osdc, 50, pixelHeight - 220, 140, 200, mmdc, 140, 0, SRCAND);
BitBlt(osdc, 50, pixelHeight - 220, 140, 200, mmdc, 0, 0, SRCPAINT);
SelectObject(mmdc, htxtBg_Cui);
BitBlt(osdc, 162, pixelHeight - 440, 255, 260, mmdc, 255, 0, SRCAND);
BitBlt(osdc, 162, pixelHeight - 440, 255, 260, mmdc, 0, 0, SRCPAINT);
SelectObject(mmdc, htxtBg_02);
BitBlt(osdc, 162, pixelHeight - 240, 255, 260, mmdc, 255, 0, SRCAND);
BitBlt(osdc, 162, pixelHeight - 240, 255, 260, mmdc, 0, 0, SRCPAINT);
SelectObject(mmdc, htxtBg_Al);
BitBlt(osdc, 162, pixelHeight - 640, 255, 260, mmdc, 255, 0, SRCAND);
BitBlt(osdc, 162, pixelHeight - 640, 255, 260, mmdc, 0, 0, SRCPAINT);
BitBlt(dc, 0, 0, pixelWidth, pixelHeight, osdc, 0, 0, SRCCOPY);
EndPaint(graphicsWindow, &ps);
ValidateRect(graphicsWindow, TRUE);
DeleteDC(mdc);
DeleteDC(mmdc);
DeleteDC(dc);
DeleteObject(h02);
DeleteObject(hAlbedo);
DeleteObject(hCui);
DeleteObject(hFrame);
DeleteObject(htxtBg_02);
DeleteObject(htxtBg_Al);
DeleteObject(htxtBg_Cui);
}

```

测试结果如下



4.2、3 中贴图优先于绘图的原因在于绘图所在的 DC 和贴图所在的 DC 都是 osdc, DoUpdate()里 select 任意 object, 都会在绘图之后, 所以应该另找放置贴图代码 (最好正好在执行绘图代码之前)。graphics.c 里的绘图代码庞杂无章, 因此耗费了大量时间找绘图代码和执行绘图的接口, 寻寻觅觅, 最终发现 display()的位置正好满足所有要求! 一时间茅塞顿开, (1) 贴图代码可以移出 graphics.c 了 (2) 贴图能在绘图之后了 (3) 游戏可以插入背景, 极大拉高了美术上限 (4) 代码不用扎堆在一起, 可以分写几个函数了。代码如下 (角色部分只展示 02, 其他角色篇幅限制不作展示)

```
void Display() {
    DisplayClear();
    DrawBackground();
    CrtItat();
    DrawChessBoard();
    DrawMenu();
    /*交互代码, 省略*/
}

void DrawBackground() {
    extern int pixelWidth, pixelHeight;
    HBITMAP hbg = (HBITMAP)LoadImage(NULL, "./source/bg.bmp", IMAGE_BITMAP, pixelWidth, pixelHeight, LR_LOADFROMFILE);
    HDC mdc;
```

```

extern HDC osdc;

mdc = CreateCompatibleDC(osdc);
SelectObject(mdc, hbg);
BitBlt(osdc, 0, 0, pixelWidth, pixelHeight, mdc, 0, 0, SRCCOPY);
DeleteObject(hbg);
DeleteDC(mdc);
}

void CrtItat() {
    if (Mypassive.ifChoose == FALSE) {

        extern int pixelHeight;
        bool flag = FALSE;
        int current_character = 0;
        if (Mousestate.mousex >= 0.5 && Mousestate.mousex <= 2) {

            if (Mousestate.mousey >= 5.2 && Mousestate.mousey <= 8) current_character = 1;

            if (Mousestate.mousey >= 2.8 && Mousestate.mousey <= 5) current_character = 2;

            if (Mousestate.mousey >= 0 && Mousestate.mousey <= 2.1) current_character = 3;
        }
        Dwtxtbg02();
        DwtxtbgAlbedo();
        DwtxtbgCui();
        DrawFrame(current_character);
        Draw_ZeroTwo(current_character == 3);
        Draw_Cui(current_character == 2);
        Draw_Albedo(current_character == 1);
    }
    else {
        switch (Mypassive.passive)
        {
            case 1: DwPlaying_02(); break;
            case 2: DwPlaying_Cui(); break;
            case 3: DwPlaying_Al(); break;
        }
    }
}

void Draw_ZeroTwo(bool flag) {
    HDC mdc;
    extern HDC osdc;

```

```

extern int pixelHeight;
mdc = CreateCompatibleDC(osdc);
if (flag == FALSE) {

    HBITMAP h02 = (HBITMAP)LoadImage(NULL, "./source/02.bmp", IMAGE_BITMAP, 2
80, 224, LR_LOADFROMFILE);
    SelectObject(mdc, h02);
    BitBlt(osdc, 50, pixelHeight - 220, 140, 200, mdc, 140, 0, SRCAND);
    BitBlt(osdc, 50, pixelHeight - 220, 140, 200, mdc, 0, 0, SRCPAINT);
    DeleteObject(h02);
}
else {

    HBITMAP h02 = (HBITMAP)LoadImage(NULL, "./source/02.bmp", IMAGE_BITMAP, 3
92, 313, LR_LOADFROMFILE);
    SelectObject(mdc, h02);
    BitBlt(osdc, 22, pixelHeight - 276, 196, 313, mdc, 196, 0, SRCAND);
    BitBlt(osdc, 22, pixelHeight - 276, 196, 313, mdc, 0, 0, SRCPAINT);
    DeleteObject(h02);
}

DeleteDC(mdc);
}

void Dwtxtbg02() {
    HBITMAP htxtBg_02 = (HBITMAP)LoadImage(NULL, "./source/textbg02.bmp", IMAGE
_BITMAP, 510, 260, LR_LOADFROMFILE);
    HDC mdc;
    extern HDC osdc;
    extern int pixelHeight;
    mdc = CreateCompatibleDC(osdc);
    SelectObject(mdc, htxtBg_02);
    BitBlt(osdc, 162, pixelHeight - 240, 255, 260, mdc, 255, 0, SRCAND);
    BitBlt(osdc, 162, pixelHeight - 240, 255, 260, mdc, 0, 0, SRCPAINT);
    DeleteObject(htxtBg_02);
    DeleteDC(mdc);
}

void DrawFrame(int current_character) {

    HDC mdc;
    extern HDC osdc;
    extern int pixelHeight;

```

```

mdc = CreateCompatibleDC(osdc);

if (!current_character) {
    HBITMAP hFrame = (HBITMAP)LoadImage(NULL, "./source/边框.bmp", IMAGE_BITMAP, 480, 220, LR_LOADFROMFILE);
    SelectObject(mdc, hFrame);
    BitBlt(osdc, -5, pixelHeight - 225, 240, 220, mdc, 240, 0, SRCAND);
    BitBlt(osdc, -5, pixelHeight - 225, 240, 220, mdc, 0, 0, SRCPAINT);
    BitBlt(osdc, -5, pixelHeight - 465, 240, 220, mdc, 240, 0, SRCAND);
    BitBlt(osdc, -5, pixelHeight - 465, 240, 220, mdc, 0, 0, SRCPAINT);
    BitBlt(osdc, -5, pixelHeight - 705, 240, 220, mdc, 240, 0, SRCAND);
    BitBlt(osdc, -5, pixelHeight - 705, 240, 220, mdc, 0, 0, SRCPAINT);
    DeleteObject(hFrame);
}
else if(current_character == 3){
    HBITMAP hFrame = (HBITMAP)LoadImage(NULL, "./source/边框.bmp", IMAGE_BITMAP, 480, 220, LR_LOADFROMFILE);
    SelectObject(mdc, hFrame);
    BitBlt(osdc, -5, pixelHeight - 465, 240, 220, mdc, 240, 0, SRCAND);
    BitBlt(osdc, -5, pixelHeight - 465, 240, 220, mdc, 0, 0, SRCPAINT);
    BitBlt(osdc, -5, pixelHeight - 705, 240, 220, mdc, 240, 0, SRCAND);
    BitBlt(osdc, -5, pixelHeight - 705, 240, 220, mdc, 0, 0, SRCPAINT);
    DeleteObject(hFrame);
    hFrame = (HBITMAP)LoadImage(NULL, "./source/边框.bmp", IMAGE_BITMAP, 672, 330, LR_LOADFROMFILE);
    SelectObject(mdc, hFrame);
    BitBlt(osdc, -53, pixelHeight - 280, 336, 330, mdc, 336, 0, SRCAND);
    BitBlt(osdc, -53, pixelHeight - 280, 336, 330, mdc, 0, 0, SRCPAINT);

    DeleteObject(hFrame);
}
else if (current_character == 2) {
    (省略)
}
else
{
    (省略)
}

DeleteDC(mdc);
}

void DwPlaying_02() {

```

```

HBITMAP htxtPl_02 = (HBITMAP)LoadImage(NULL, "./source/02_playing.bmp", IMAGE_BITMAP, 960, 580, LR_LOADFROMFILE);
HDC mdc;
extern HDC osdc;
extern int pixelHeight;
mdc = CreateCompatibleDC(osdc);
SelectObject(mdc, htxtPl_02);
BitBlt(osdc, 15, 200, 480, 580, mdc, 480, 0, SRCAND);
BitBlt(osdc, 15, 200, 480, 580, mdc, 0, 0, SRCPAINT);
DeleteObject(htxtPl_02);
DeleteDC(mdc);
}

```

最终呈现如下，为定稿。



## 测试案例六

文件：Init.c

测试内容：检查音乐播放。

测试过程：

由于 graphics.c 库函数中并没有给出导入音频的函数，所以在准备导入背景音乐时首先选择了 mciSendString。

使用 mciSendString 函数进行首次测试时，函数如下。

```

void AddBGM(){
    mciSendString("open ./source/boom.wav", NULL, 0, NULL);
    mciSendString("play ./source/boom.wav", NULL, 0, NULL);
}

```

```
}
```

播放音乐成功，随后实现背景音乐的循环、暂停功能。

实现循环：`mciSendString("open ./source/boom.wav repeat", NULL, 0, NULL);`

实现暂停：`mciSendString("pause ./source/boom.wav ", NULL, 0, NULL);`

在开始测试后发现音乐无法正常播放，在搜集资料后得知，repeat 现已不支持在文件名后实现功能。

暂停功能应由关键字 stop 实现。

由于无法实现循环，遂改用 playsound 函数实现。

```
void AddBGM() {
    PlaySound("./source/bkmusic.wav", NULL, SND_FILENAME |
SND_ASYNC | SND_LOOP);
}
```

音乐正常循环播放。实现其停止播放变得非常简单：

```
PlaySound( NULL, 0, NULL);
```

但是在播放爆炸声音时出现了问题，由于 PlaySound 函数无法实现多音轨同时播放音乐，导致爆炸声的播放会暂停背景音乐，所以在播放爆炸声时采用了 mciSendString 函数实现：

```
mciSendString("open ./source/boom.wav", NULL, 0, NULL);
mciSendString("play ./source/boom.wav", NULL, 0, NULL);
.....
mciSendString("close ./source/boom.wav", NULL, 0, NULL);
```

实现爆炸声和背景音乐的正常播放。

## 测试案例七

文件：Init.c

测试内容：检查菜单栏和弹窗的实现。

测试过程：

菜单栏建立过程中使用了 graphics.c 库函数提供的 button 和 menulist 函数，对其坐标的确认特别是对其进行对其、居中的操作进行了测试。

```
setMenuColors("BLACK", "WHITE", "Gray", "WHITE", 1);
setButtonColors("BLACK", "WHITE", "Gray", "WHITE", 1);
SetFont("微软雅黑");
SetPointSize(14);
static char* MENUMORE[] = { "更多",
```



```

    "开发者名单",
    "关闭背景音乐",
    "打开背景音乐",
    "排行榜"
};
double FH = GetFontHeight();
double x = 10;
double y = cy;
double h = FH * 1.8;
double w = TextStringWidth(MENUMORE[0]) * 3;
double wlist = TextStringWidth(MENUMORE[1]) * 1.3;

```

这些函数规定了菜单中的字体风格、字体大小以及按钮的大小。w和wlist两变量根据已有菜单大小规格规定菜单位置，x，y两变量根据窗口大小确定位置。

在一次测试后全部实现。

实现窗口功能：

```

HWND zzz = GetStdHandle(STD_OUTPUT_HANDLE);
SetWindowText(zzz, ".....");
int isok1 = MessageBox(NULL, ".....", ".....", 0);

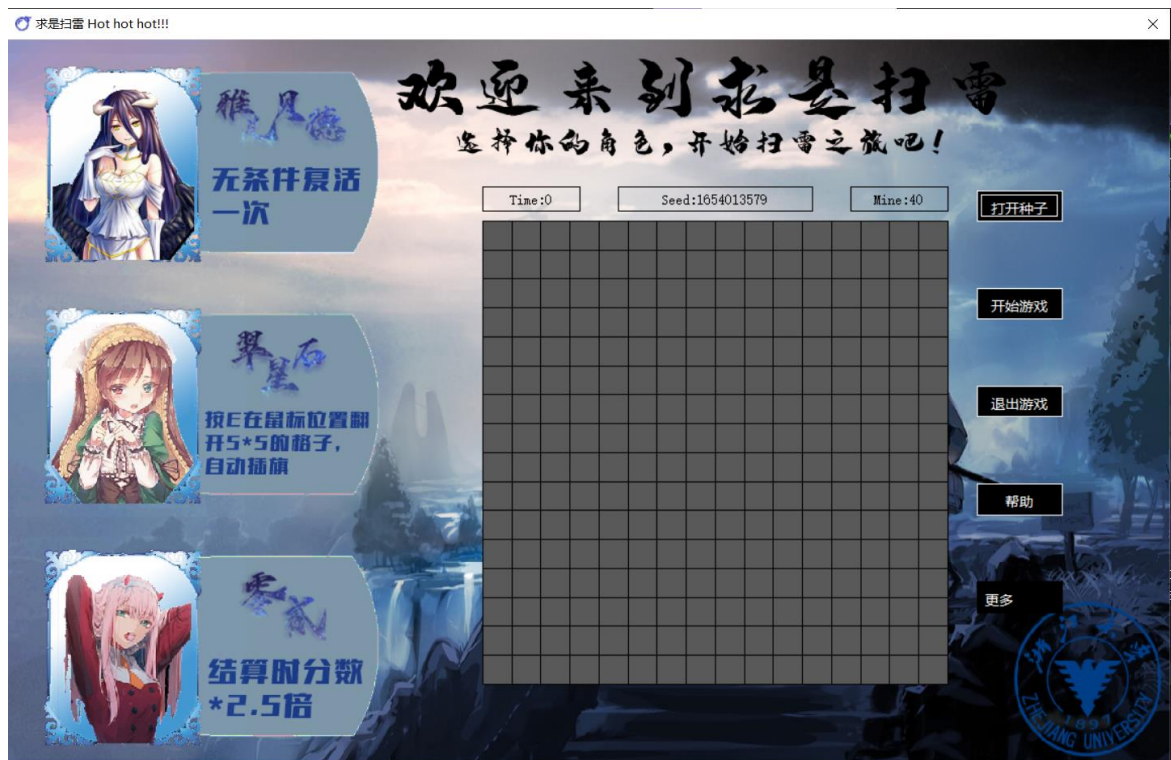
```

使用GetStdHandle（STD\_OUTPUT\_HANDLE）获取标准输入句柄，即当先窗口句柄，再使用MessageBox函数输出窗口。一次测试后全部实现。

## 使用操作

### 1) 基本分区介绍

编译运行后初始界面如下。最上方的白框可以看到，扫雷游戏的图标和游戏的名称。往下进入图形界面。左侧为角色界面，从左到右是角色图和技能介绍，从上到下分别是雅儿贝德、翠星石和零二。雅儿贝德的右边是本游戏的标题和问候语。位列中央的是扫雷棋盘，游戏主体，可以看到棋盘左上角是个计时器、右上角则显示剩余雷数。棋盘右侧，是种种游戏选项，从上到下分别是打开种子、开始游戏、退出游戏、帮助和更多。



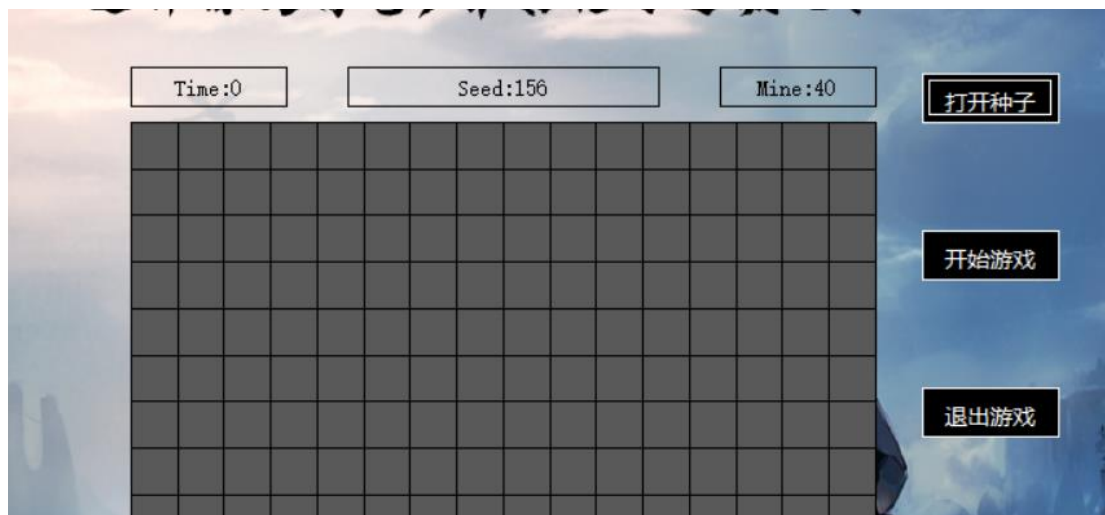
## 2) 角色交互（以零二为例）

当鼠标箭头放在零二图的位置时，零二图会放大，左键点击后，剩下角色消失，零二（更换插图）单独显示在图形界面左侧。



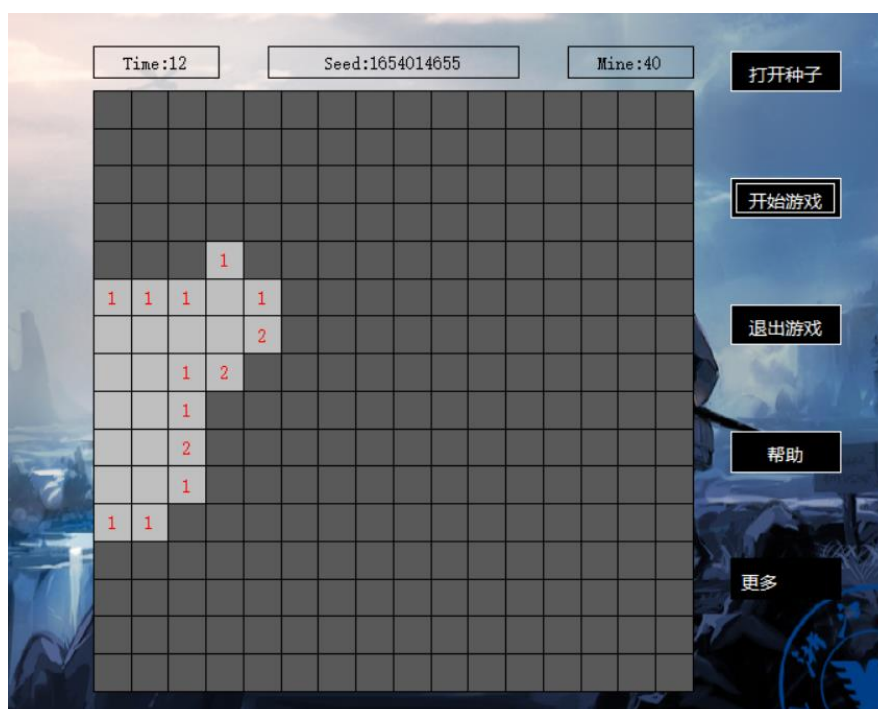
### 3) 打开种子

左键点击“打开种子”按钮（或者按“ctrl”）后，种子栏将会清空。要想使用种子值游戏，输入数字即可，可以按 Backspace 键撤回输入的数字。输入完点击“开始游戏”按钮，即可按照输入的种子值游玩游戏。



### 4) 开始游戏

当没有选择“打开种子”按钮，直接左键点击“开始游戏”，棋盘生成地图，当前种子值显示在 Seed 框中，后续可以输入相同的种子值重复游玩。



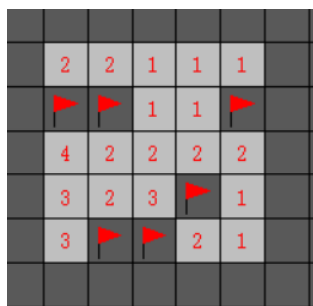
当选择“打开种子”并输入种子值后点击“开始游戏”，将按照输入的种子值生成扫雷地图。

## 5) 游戏过程中

对于未翻开的格子，鼠标停留会有高亮。左键点击该格子，如果是空白，会接着翻周围的区域；如果是数字，仅翻开这个格子；如果是雷，没有选择雅儿贝德则游戏失败，选择雅儿贝德则还有一次机会，点中的雷显示为白旗作为提示，如下图。右键点击没有翻开的格子，将会插红旗，剩余雷数减一，再次右键取消插旗。



另外，翠星石的技能在游戏过程中使用，鼠标停留在某个格子上，按“E”即可。效果如下。



## 6) 游戏失败

踩中雷，视为游戏失败。此时会有弹窗提示，会显示分数。



## 7) 游戏胜利

当除雷以外的格子全部打开时，游戏胜利，跳出弹窗提示，显示得分。



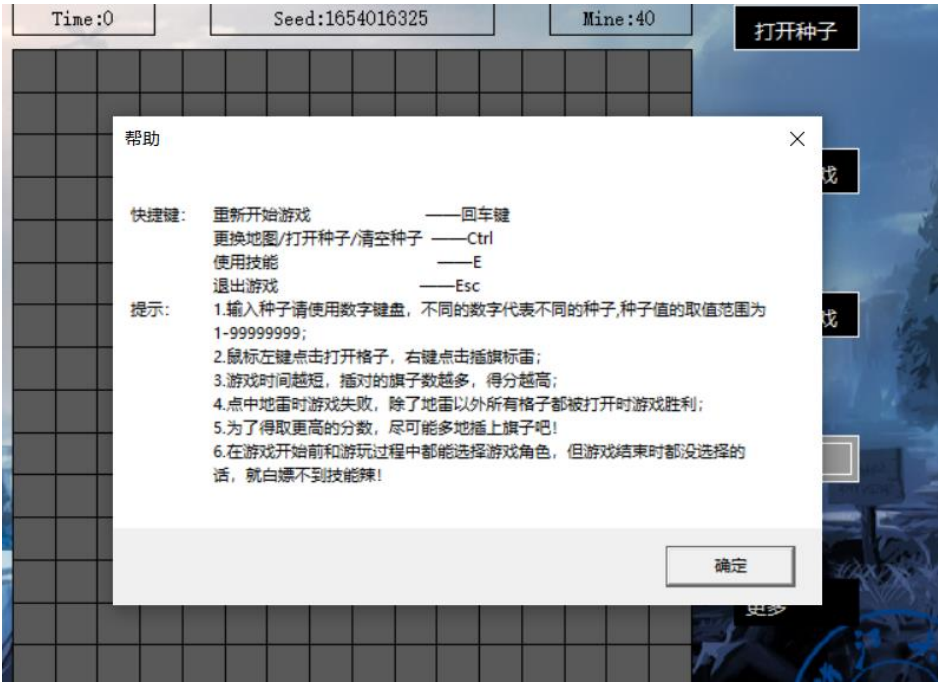
## 8) 退出游戏

左键点击“退出游戏”按钮，关闭应用。

## 9) 帮助

左键点击“帮助”按钮，跳出帮助弹窗。帮助弹窗里写明了游戏的快捷键、一些温馨提醒和游玩提示。





10) 更多

左键单击“更多按钮”，会弹出四个选项。



5 团队合作

5.1 任务分工

模块	人员	分工
可视化	B	界面设计，音乐，游戏背景设计
	A	位图贴图、美工，界面布局设计
	C	按钮、计时器、游戏主要交互部分

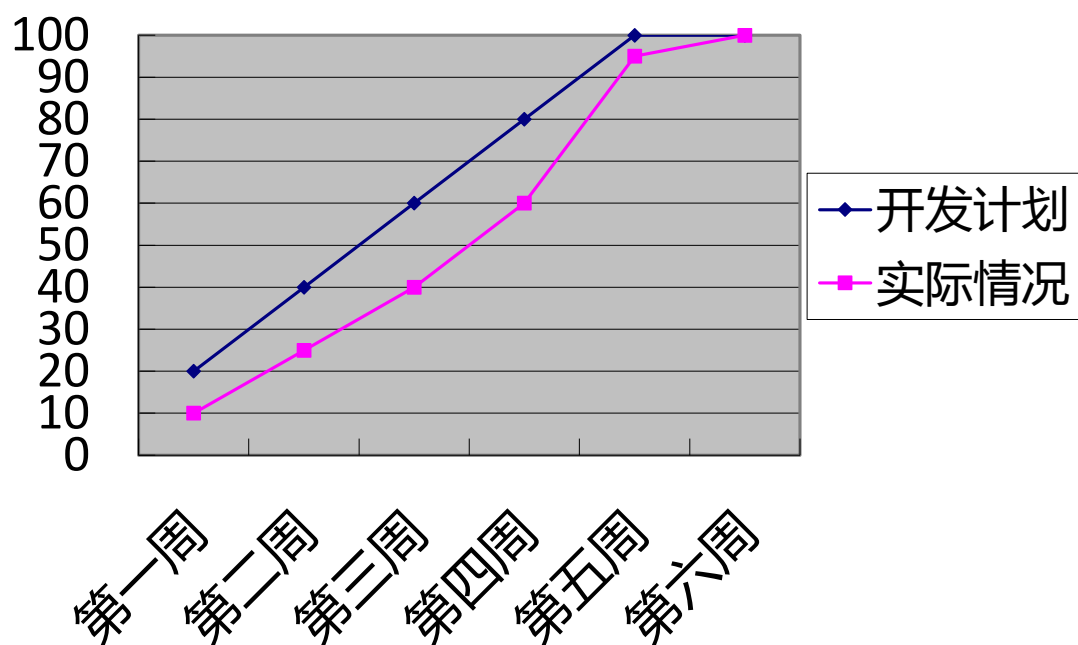
		位置确定
交互	C	键盘、鼠标等交互的实现，整合交互、可视化和逻辑模块
	B	修改整合
	A	角色交互
逻辑	A	地图后端信息生成、翠星石技能、后端打开格子、最终分数算法
	C	最终分数算法、游戏过程逻辑整合
文件	A	写入、读取排行榜
	B	读取排行榜

## 5.2 开发计划

时间	人员	内容
第一周（4.18~）	A、B、C	浏览 libgraphics 库和 SimpleGUI 库；初步定下游戏内容：1.经典游戏实现；2.存档；3.暂停；4.加入游戏道具；5.双人对战
第二周（4.25~）	A	初步分工负责逻辑部分。设计扫雷信息后端存储方式，完成初始化地图、随机出雷等后端扫雷函数
	B	初步分工负责图形界面实现。进行了一定的界面设计，尝试使用外接的库进行插入图片
	C	初步分工负责交互。构建了基本交互结构
第三周（5.9~）	A	完成更新排行榜函数，02 的被动技能函数；开会后写完种子值相关函数，修改原先的随机出雷函数

	B	加入音效，绘出图形界面雏形
	C	拼接完善三人现有成果
第四周（5.16~）	A	完成初始翻开域的函数，完善了翻格子的递归算法，WinAPI 学习，在周末实现了图片导入
	B	尝试导入图片，绘制旗子，图形界面菜单栏修改
	C	将图片、图形与已有函数对接，实现快捷键和基本游戏功能
第五周（5.23~）	A	修改代码使规范，完成角色的所有内容
	B	对接、修改图形界面设计
	C	修改测试中出现的游戏 bug
第六周（5.30~）	A、B、C	完成项目、撰写报告

开发进度图：



## 5.3 编码规范

### 1. 变量命名



变量命名采用驼峰命名法。如

```
char NumToString[16][16][20];
char* String = (char*)malloc(sizeof(char) * LINELENGTH);
```

## 2.h 文件格式

条件编译示例：

```
#pragma once
#ifndef _logic_h
#define _logic_h
    (省略)
#endif // !_logic_h
```

## 3.预处理指令

```
#include<stdio.h>
#include "logic.h"
```

## 4.函数声明/函数定义

左花括号与函数名同行。

```
void CreateMine() {
    int i = 0;
    for (; i < MAXROW; i++) {
        for (int j = 0; j < MAXCOL; j++) {
            Mine[i][j].ifMine = FALSE;
            Mine[i][j].ifCovered = FALSE;
            Mine[i][j].num = 0;
            Mine[i][j].flag = 0;
        }
    }
}
```

## 5.空格的使用

关键字之后要留空格，否则无法辨析关键字。像“if”、“for”、“while”、“catch”

等关键字之后应留一个空格再跟左括号“(”，以突出关键字。

逗号“,”之后要留空格，如

```
setButtonColors("Dark Gray", "Black", "Gray", "Black", 1);
```

如果“;”不是一行的结束符号，其后要留空格，如

```
for (int j = 0; j < MAXCOL; j++)。
```

函数名后不要留空格，紧跟左括号“(”，以与关键字区别。

一元操作符如“!”、“~”、“++”、“--”、“&”（地址运算符）等与操作数之间

不留空格；像“[]”、句点“.”或箭头“->”这类操作符前后不加空格。

赋值操作符、比较操作符、算术操作符、逻辑操作符、位操作符，如“=”、“+=”、“>=”、“<=”、“+”、“\*”、“%”、“&&”、“|”、“<<”、“^”等二元操作

符的前后应当加空格。

如对于表达式比较长的 for、do、while、switch 语句和 if 语句，为了紧凑起见可以适当地去掉一些空格。

## 5.4 合作总结

2022 年 4 月 29 日：



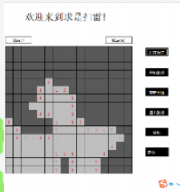
会议形式	线下会议	参会人员	A、B、C
会议内容			
<p>1. 初步确定扫雷游戏的功能和实现：</p> <p>经典游戏实现：存档；暂停；加入游戏道具；双人对战</p> <p>2. 开发亮点：</p> <p>存档、暂停可以看作游戏的功能完善；加入游戏道具和双人对战的模式，可以让扫雷的可玩性更进一步。</p> <p>3. 挑战点：</p> <p>传统功能方面：对绘图函数不熟悉，对交互的实现不熟悉，对扫雷游戏本身理解不够透彻；拓展功能方面，存档的数据有哪些、需要如何储存，游戏道</p>			

具的设计和双人对战如何实现。

2022 年 5 月 8 日：

会议形式	线下会议	参会人员	A、B、C
会议内容			
<p>1. 删改扫雷游戏的功能和实现：</p> <p>经典游戏实现：存档改为分数排行榜；加入角色扮演；分数机制；音效</p> <p>*暂停功能的删除：扫雷是按时间计分的游戏，如果能暂停，就能先把雷计算好再继续，有损公平，因此删去暂停功能。</p> <p>*双人对战的删除：首先，一台电脑一般只配置一个鼠标，无法同时进行两句扫雷游戏；其次，完全可以一个人先游玩一局，再另一个人游玩，再进行对比。由此，我们也诞生了一个新 idea——固定种子值随机。</p> <p>2. 开发亮点（修改后）：</p> <p>分数排行榜可以看作游戏的功能完善；加入角色扮演的形式，是对 ARPG 游戏的学习借鉴，既美化了界面，亦更迎合广大青年向往新潮时髦的心理；宏大的背景音乐和动人的音效进一步加强了游玩体验；不低的难度使游玩充满惊喜；</p> <p>3. 挑战点（修改后）：</p> <p>传统功能方面：对绘图函数不熟悉，对交互的实现不熟悉，对扫雷游戏本身理解不够透彻；拓展功能方面，排行榜设计文件操作、动态分配内存等不熟练的知识，角色贴图的插入以及伪随机过程的学习掌握。</p>			

2022 年 5 月 28 日：

会议形式	线上会议	参会人员	A、B、C
会议内容			
1. 互相补足信息差，便于整合：			
<div><div><div>怎么关闭啊</div><div>哦是哦</div><div>我这因为没有数据 按了确定就会闪退</div><div></div><div>雷我可以画成上面红色这部分的样子</div><div>就是不画雷 画雷爆炸</div><div>星期天 12:51</div><div>是按esc退出 他这设了isrank的判定</div><div>噢噢噢</div></div><div><div>我这里改的应该能循环了</div><div>排行榜</div><div>我想的是 把文本信息读出来赋给一个指针</div><div>然后弹窗可以输入指针</div><div>你看你能把排行榜的函数返回成一个字符指针不</div><div>我不知道那个黄金矿工那个怎么实现的</div><div>我试试 改这个真的折寿</div><div>那个bgm循环</div><div>就是在bgm那个函数后面加一个repeat</div><div>星期天 15:57</div><div>那个没有成绩 至少完成一局那个我可以外部判断</div><div>可以试试</div></div></div> <div><div><div>比等线好看一点</div><div>把LINELEGNTH改成13</div><div>把字号改大一点</div><div>星期天 16:30</div><div>那个没有成绩 至少完成一局那个我可以外部判断</div><div>这个排行榜只需要返回txt里面的字符内容</div><div>既然我这写了 你就不用再写了</div><div>全放一个字符串 得动态分配</div><div>一直报错</div><div>我懂了</div><div>My Soul July</div><div></div></div><div><div>欢迎来到我的迷宫!</div><div></div><div>这样子的</div><div>字号大一点好的</div><div>选项的字号对吧</div><div>不是</div></div></div>			
2.开发亮点（最终版）：			

分数排行榜可以看作游戏的功能完善；加入角色扮演的形式，是对 ARPG 游戏的学习借鉴，既美化了界面，亦更迎合广大青年向往新潮时髦的心理；不同的角色有着不同的技能，使游戏里里外外充满了博弈；宏大的背景音乐和动人的音效进一步加强了游玩体验；精心设计的游戏背景；可支持按种子值游玩同一个地图，充分满足玩家不断拔高分数的心理，是对经典扫雷最大的提升；不低的难度使游玩过程充满惊喜。

### 3.挑战点（已克服）：

传统功能方面：对绘图函数不熟悉，对交互的实现不熟悉，对扫雷游戏本身理解不够透彻；拓展功能方面，排行榜设计文件操作、动态分配内存等不熟练的知识，角色贴图的插入以及伪随机过程的学习掌握。

## 5.5 收获感言

A:

本项目是我学习编程以来负责的第一个项目，仅仅学习半年的我就要一改此前零散、碎片的编程实践，去写一个功能丰富的项目。起初我非常质疑我的能力，图形界面、交互根本就没学过，但一想到，我的同龄人行，那我也行，遂放下惶恐。

这个项目，总体上我是满意的，大概 17 个版本才最终定型，每一次版本的更迭，都是用户体验的提升。而现在，看着屏幕里最终稿的样子，只剩下热泪盈眶，一个月前，我们根本没想过要创造如此精彩的作品。但终究时间受限，欠缺打磨，分数曲线设计得比较粗糙，美工我还觉得有些不足等等。

我很庆幸，第一个项目会是最热爱的游戏项目。我想，如果不是做一款游戏，那我绝不会有 coding 到深夜一两点的热情，绝不会有反复扣一张图的执着，也绝不会有源源不断的 idea。

这个项目承载了太多的第一次，也承载了太多的疯狂。第一次做大项目、第一次担任美工的角色、第一次拍板决定、第一次为了一件事情疯狂查资料看教程、

第一次在这么短的时间完成一次蜕变。我从未想过，原来短短三天时间就足以改变一个人——三天之前，我还在纠结如何导入图片；三天之后，我学会了 PS 的基本操作、能扣出我满意的图片，用 WinAPI 初步创建窗口、导入我的图片，在长达千行的 `graphics.c` 里上下求索、努力看懂原作者的代码、反复修改我添加代码段的位置……最令我满意的一次瓶颈突破，无疑是我在某一天对 `osdc` 的理解突然升华了，突然就学会了如何插入背景（此前背景会把绘图遮住），那个瞬间我意识到了这个游戏的美术上限将被拔高多少，激动到舌头打结。

我还要感谢我的另外两位搭档，我的好 `bro`。他们的热情和投入与我相差无几，他们的 `idea` 和野心也都令我怦然心动。我很佩服他们在 `coding` 过程中一些精彩的创造，有一部分可能我至今也写不出来。我们没有一个人可以替换，但凡缺少一人，都不会有如此棒的作品。

最后，我还得感谢出题者，是你让我认识，我对游戏的热爱，是真的。你为我开辟的游戏创造之路，我会走下去。

PS：学校提供的库功能残疾得可怜，更新一下吧，不要再祸害学弟学妹了。

## B:

学习 C 语言是我上大学以来遇到的最困难的学科，除了缺乏基础知识以外，也并没有很大的兴趣。然而在大项目开始之初，想到在团队中合作一起设计一个扫雷游戏，就令我热血沸腾。兴趣是最好的老师，当有了兴趣之后开始愿意花费时间去学习相关知识。一开始我想主攻 `easy.x` 库函数辅助进行界面设计，在仔细研究了好多个小时的 b 站课程后，初步实现了界面，然而，由于和 `graphics` 库函数不兼容，之前的学习打了水漂。我开始重新研究 `graphics` 的每一个函数，从画线到按钮。因为一些功能的需要，去学习 Windows 窗口程序的各类函数，从理解句柄到实现弹窗。兴趣驱动自然是不够的，还有责任驱动。我的两位队友也是我的室友，我不能给团队拖后腿，所以我更有动力去学习。

虽然如此，在工作中仍然会有懈怠，会有问题，感谢我的两位室友的包容和理解，一起的合作与努力不仅使扫雷游戏如期实现，更增进了我们之间的了解，体会了合作的乐趣，增进了我们的友谊，这是比项目实施还要重要的。

## C:

这个项目是我在上大学以来做过最有成就感的事情，在刚看到这个任务的时候我感觉十分困难，但是当大家一起讨论，自主学习，然后一点一点的逐步实现甚至超越了原有扫雷的功能时，一种叫做骄傲与自豪的情绪油然而生。

因为上学期 C 语言的基础本来就不是很好，这个学期在图形与交互这一块又一直没怎么弄懂，所以在刚接手交互这一块任务的时候我是很没有把握的。刚开始的时候，我把智云课堂看了好多遍，又去 bilibili 上找了一些制作扫雷、推箱子的游戏的视频去学习，但是大家使用的库不一样，导致视频中的交互部分和我们需要的截然不同。而且，因为刚开始没有调试好 visual studio，我仅仅将没用报错作为依据来写函数，最终导致的结果就是在调试好后我的所有函数都是不可用的。随后，我突然想起老师发过回调函数使用的案例，我就在那些.c 文件里去一个一个的看过来，再结合自己的想法改进，最终做到了想要的效果。

之后的环节中，我基本都是在根据讨论的结果将大家的函数缝合在一起，在后续的美工环境都没有出过什么力，这让我有些许的惭愧，但好在后来讨论中又加入了新的内容，让我有了一定的存在感。

虽然过程中做了很多的无用功，但是这段寝室三人一起从天将暗一起讨论、修改产品到凌晨的日子不仅让我体会到了充实成就的感觉，更加深了我们彼此的了解，增进了我们的友谊，我为学校即将不再开设这门课感到遗憾，希望将来有一天，这个作业能重现江湖。

## 6 参考文献资料

[1] 毛星云. 逐梦旅程: Windows 游戏编程之从零开始

[2] C++中获取静态数组和动态数组的长度

<https://blog.csdn.net/kksc1099054857/article/details/78265970>

[3] C 语言 rename() 函数和 remove() 函数——重命名或移动、删除文件或目录

[https://blog.csdn.net/Brouce\\_Lee/article/details/81194040](https://blog.csdn.net/Brouce_Lee/article/details/81194040)

[4] C 语言游戏 双缓存解决闪屏问题 详细总结

[https://blog.csdn.net/Young\\_IT/article/details/103103867](https://blog.csdn.net/Young_IT/article/details/103103867)

[5] VK-虚拟键盘 (C++)

<https://abraverman.gitee.io/2021/01/25/VK-biao/>

[7] 学在浙大 图形编程相关资料 (2) .rar

<https://courses.zju.edu.cn/course/39131/learning-activity/full-screen#/495540>

[8] C/C++初级编程实践——推箱子小游戏~大学作业不用愁啦~

[https://www.bilibili.com/video/BV11i4y177aD?spm\\_id\\_from=333.337.search-card.all.click](https://www.bilibili.com/video/BV11i4y177aD?spm_id_from=333.337.search-card.all.click)