程序设计基础大作业

王者荣耀连连看

姓名 陶乐天

班级 汽 83

学号 2018010771

日期 2020 年 12 月 26 日

目录

1 程序功能介绍及实现思路

该游戏以绘图库 VS2019+EasyX_20200902 为基础，结合热门手游王者荣耀，制作而成的王者荣耀连连看。

编程以不同的界面为基础，分为 4 个模块，分别为菜单界面、游戏界面、游戏设置界面、排名界面。程序进入主函数后，通过全局参数的改变在函数 Menu()、GamePlay()、setting()、RankInit() 之间跳转，代码如下所示。

```
int main() {
    initgraph(900, 580);
    begin:
    dataInit();
    Menu();
    if (menu_choose == 0) {
        GameInit();
        GamePlay();
        goto begin;
    }
    else if (menu_choose == 1) {
        setting();
        goto begin;
    }
    else if (menu_choose == 2) {
        RankInit();
        goto begin;
    }
    else if (menu_choose == 3) {
    }
    closegraph();
    return 0;
}
```

接下来一一介绍各个功能模块的设计和实现思路。

1.1 菜单界面设计

1.1.1 功能介绍

该界面是用户开启软件的后直接出现的菜单界面，如图1所示。

菜单界面有 5 个可用按键，分别用于进入游戏界面、游戏设置界面、排名界面和游戏充值，其中游戏充值是本游戏的特色功能，充值后英雄会换上皮肤，游戏中点击提示会有特别效果，基础得分加 100。当鼠标放在对应可点击区域时，对应区域会出现高亮提示。这部分界面设计和代码实现参考 EasyX 参考项目数独[1]。

1.1.2 实现思路

高亮显示 在 Menu() 函数中运行一个死循环，每次循环通过 msg.GetMouseMsg().x 和 msg.GetMouseMsg().y 获取鼠标位置，msg 是一个 MOUSEMSG 类型变量，MOUSEMSG 用于给出鼠标信息，在 <graphics.h> 库中给出。获取鼠标位置信息后，每循环对相关显示参数 (如字体颜色、边框颜色) 进行刷新，从而鼠标在特定位置时，对应点击区域后出现高亮显示。

点击操作 当 msg.uMsg 为 WM_LBUTTONUP，即鼠标为点击操作时，判断鼠标位置是否在对应区域，是则设置全局变量，跳出循环，通过设置的全局变量进入下一个界面。

[1] https://codebus.cn/chenh/a/sudoku

图 1: 菜单界面

1.2 游戏界面设计

1.2.1 功能介绍

该界面是玩家玩游戏的界面，如图2所示。游戏界面分为游戏区域和设置区域。

游戏区域会显示 10*14 的头像，在英雄池中有 50 个英雄，每次游戏会选择 13 个英雄，在主界面点击充值后，所以英雄会换上皮肤。

设置区域显示得分、提示次数、洗牌次数、剩余游戏时间。点击提示会消去一对图片，充值后能连环消去多个图片，点击洗牌会随机洗牌，点击退出回到主界面，当对应提示、洗牌次数为 0 时，程序也进行了保护。每次正确消除会得到两分加成，当玩家连续两次消除小于两秒时，得分处会显示 DOUBLE，得分加倍，不同难度的关卡设置的加成分数也不一样。

图 2: 游戏界面

1.2.2 实现思路

加载图片 程序在 ImportImage() 用 loadimage(&img[0], L"./fig/0.jpg", box_width, box_width) 加载图片，输入参数为待加载的图片变量指针，图片路径，图片大小。对充值用户，在另一个文件夹中获取带有皮肤的英雄图片。每次加载图片 50 张，但每次只选择 13 张作为每次游戏的图片。这通过 1000 次随机交换对应图片实现。交换图片代码如下，这里多次交换代替随机打乱的思路在程序中多次用到，例如点击洗牌后的效果和下述图片生成的方法。

```
int i, j, T = 1000;
IMAGE tmp;
while (T--)
{
    i = rand() % 50+1;
    j = rand() % 50+1;

    tmp = img[i];
    img[i] = img[j];
    img[j] = tmp;
}
```

图片生成 在 GameInit() 中生成图片，为了保障可解性，我在图像的上半区的每个位置随机选择一个英雄放入 1 维数组 MAP。MAP 用于储存各个位置的图片标签，可以通过 MAP[x , max_col * y] 访问到各个位置的图片。下半区直接复制上半区，并并进行 100 次的随机交换作为随机打乱。有趣的是，当随机交换次数为 1000 次时，上下部分出现的图片大部分相同，没有达到打乱效果，交换 100 次打乱效果较好。因此，这里通过随机一般图片-直接复制-随机交换实现了可解性。打乱后对相邻图片进行判断，相邻相同图片小于 10 个则会重新生成。

可连性判断 消除思路参考知乎:C++ 连连看教程[2]。总体思路是将三段连接判断 is_connect_three() 分解为其中一点位置十字区域内是否存在点和另一点二段连接，二段连接判断 is_connect_two() 分解为是否能通过矩形对角点和另一点一段连接，而图像的一段连接 is_connect_one() 分解为横向连接和纵向连接，即两点之间是否全部为空白进行判断。在判断两点连接性时，通过 is_connect 一次调用三种可连性判断函数进行判决。

消去动画 当成功连接两点时，对应路径上会出现一条系统判断的消去路径。此路径的绘制和消去耗费了我很多精力。简单的思路就是在显示路径后，将需要消去的点存入全局变量 clear_list，等待 200ms 后消去，同时这 200ms 也是加倍得分 DOUBLE 显示的时间。由于在可连续判断时，多层函数调用，因此很难理清这些线是否需要连接，点是否真需要删除，因此在 is_connect() 传入两个点坐标的同时，也传输参数 is_draw 判断是否需要绘制点。这里的消去动画设置逻辑复杂，且没有移植性，因此不做详细介绍。从代码优化的角度看，我觉得这里的代码虽然实现了消去效果，但是不够优雅，值得进一步优化。

提示功能 点击提示时，程序会从左上角开始，每个点一次判断和其后的点的可连接性。一开始我找到第一个点没有及时 break 导致较多点对被消除，因此我把这个 BUG 修改成充值后的提示效果。这里的二重遍历也在每一次消除后判断图中是否还有可连接的点，不能继续连接就会进行自动洗牌。但程序从未跑到出现不可连接的情况，因此这部分程序一直没有得到测试。

[2]https://zhuanlan.zhihu.com/p/59069810

1.3 游戏设置界面设计

1.3.1 功能介绍

通过改变英雄数量提高游戏难度，基础数量为 13，可以选择 13、15、17 个英雄，三者在快速消去时分别可获得 4、5、6 分的加成。

1.3.2 实现思路

通过改变全局变量 num_hero 改变英雄数量。

1.4 排名界面设计

1.4.1 功能介绍

读取记录分数的 txt 文件，显示排行榜。

1.4.2 实现思路

这里采用 fscanf_s 进行文件读取，再通过冒泡排序一次排行，最终只显示前 8 名。文件读取和排序代码如下:

```
err = fopen_s(&stream, "score.txt", "a+");
char ch = 'c';
int term = 0;
while (ch != EOF) {
    fscanf_s(stream, "%d", &time_rank[term]);
    ch = fgetc(stream);
    term++;
}
int temp;
for (int i = 0; i < term-2 ; i++) {
    int isSorted = 1;
    for (int j = 0; j < term-2 -i; j++) {
        if (time_rank[j] < time_rank[j + 1]) {
            temp = time_rank[j];
            time_rank[j] = time_rank[j + 1];
            time_rank[j + 1] = temp;
            isSorted = 0;
            }
        }
        if (isSorted) break;
}
```

类似地，每次胜利后将分数写入文件的代码如下:

```
FILE* stream;
errno_t err;
err = fopen_s(&stream, "score.txt", "a+");
fprintf(stream, "%d\n", score);
_fcloseall();
```

2 程序整体架构和调用关系

2.1 程序整体架构

程序写在一个 main.cpp 文件中，main 调用 Menu、GamePlay、setting、RankInit 四个函数产生四个图形界面。在每个界面函数中先进行界面初始化，显示初始界面；再进行死循环，在循环中获取鼠标位置信息和状态信息，当鼠标操作满足特定条件时，有两种处理方式:

1. 通过循环内部的函数改变图形显示状态，例如鼠标在特定位置的高亮显示、连对满足可连性消除;
2. 通过全局变量改变，跳出循环，进入下一个函数中，例如在菜单界面点击开始游戏，进入游戏界面。

除了图形显示外，其他通用的数据处理操作我也封装成了函数。例如判断两点的可连性，除了传入判断点的坐标外，还使用了全局变量 MAP 表示当前状态。

综上所述，程序中函数分为两类: 界面显示函数和数据处理函数，主函数外还封装了 23 个函数，命名力求通俗，排布顺序即为函数调用顺序，复用性和可读性强。

2.2 函数调用关系

各个模块的函数调用关系通过 Visual Studio Enterprise 2019 的 code map 功能自动生成，如图3所示，由于调用关系不易用语言说清，在此不做详细分析。

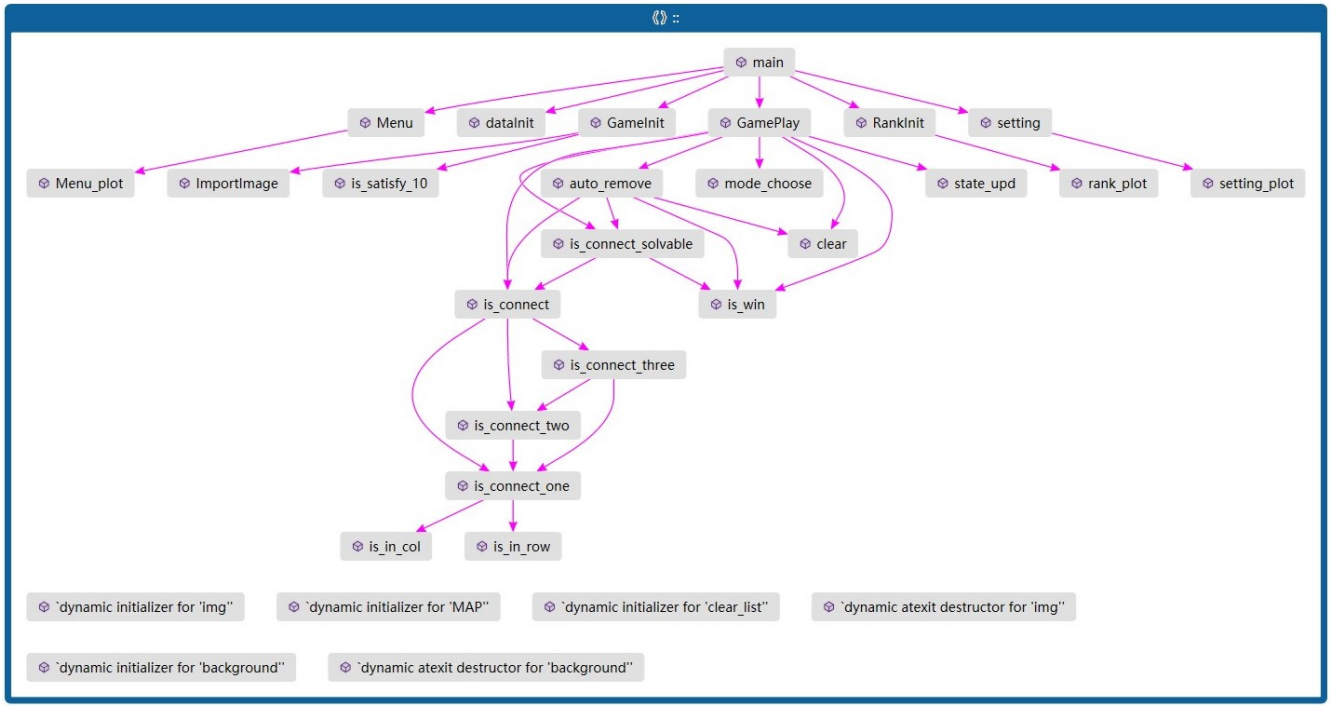

图 3: 函数调用关系

3 程序亮点与不足

3.1 亮点

3.1.1 函数封装

除主函数外，通过以下 23 个函数实现全部功能，每个函数目标明确，易于阅读和修改。同时在函数调用关系图3中可以看到，一个函数被多个函数调用，说明代码复用性好。

```
void dataInit();
void Menu_plot();
void Menu();
void GameInit();
void ImportImage();
void GamePlay();
int is_connect(int x1, int y1, int x2, int y2,int is_draw);
void state_upd();
int is_in_col(int x1, int y1, int x2, int y2);
int is_in_row(int x1, int y1, int x2, int y2);
int is_connect_one(int x1, int y1, int x2, int y2);
int is_connect_two(int x1, int y1, int x2, int y2);
int is_connect_three(int x1, int y1, int x2, int y2, int is_draw);
void clear();
int is_satisfy_10();
void RankInit();
int is_win();
void mode_choose();
void setting();
void setting_plot();
void rank_plot();
int is_connect_solvable();
void auto_remove();
```

3.1.2 程序保护

当提示次数和洗牌次数为 0 时，玩家再次点击相应按钮会提示次数用完，而不会出现负值情况，对程序进行保护，代码如下。

```
//鼠标点击提示区域
else if  (x > 760 && x < 850 && y >245 && y < 295){
    if (shuffle == 0) {
        MessageBox(GetHWnd(), L"你的提示次数已经用完!", L"提示", 0);
     }
    else {
        //进入正常的提示消除部分
        //省略
    }
}
```

3.1.3 图像刷新

很多样例程序采用在每次 while 循环中对全部图片进行刷新，导致出现闪屏的情况。这里为了避免这个情况，在 GameInit 中一次性显示图片，之后每次消除后只需要将对应清空的格点置白。但这样也给其他功能的

实现带来麻烦，例如消去连接提示线 (每次成功消去短暂停留的连接线) 时，就需要额外的 clear_list 记录需要消去的格子。

3.1.4 充值功能

充值提示功能原本是我找到提示对后没有 break 的 BUG，但这个 BUG 被我改成了一个充值亮点。同时这种小游戏的充值也讽刺了当今的游戏环境。

3.2 不足

3.2.1 边界处理

玩家无法对边界上的相同图片通过边界连线进行消除。这里的实现思路只需要将原 MAP 周围加上一圈空白格即可。但由于一开始编程时我没注意这点，且程序中对 MAP 采用了两种如下的索引方式，我尝试修改后没能正确实现。最后向谌老师征求意见，得到答复是这属于功能实现范畴，不属于编程问题，于是我就没对边界格点消除进行实现。

```
1  //1维索引
2  for (int i = 0; i < max_col * max_row  ; i++)
3      MAP[i] = rand() % (num_hero) + 1;
4  //2维索引
5  for (int i = 0; i < max_col; i++)
6      for (int j = 0; j < max_row; j++)
7          putimage(start_x + i * box_width, start_y + j * box_width, &img[MAP[i + j * max_col]]);
```

3.2.2 提示线绘制

提示线消除的代码集中在判断可连性处，但由于我只返回是否可连的 01 变量，因此具体实现复杂，无法复用，例如练练规则改成允许四段连接，程序就需要大改。具体来说一段连接和二段的提示线在 is_connect 中绘制，三段连接提示线在 is_connect_three 中绘制，虽然实现了功能，但代码逻辑不清晰。

4 用户使用手册

4.1 基础功能

4.1.1 开启程序

点击王者荣耀连连看文件中的 code.exe 开启游戏，进入菜单界面如图1。

4.1.2 开始游戏

在菜单界面中点击开始游戏，进入游戏界面如图2所示，左边游戏区域为练练看游戏，右边设置区域可以查看游戏分数和剩余游戏时间。具体说明如下：

1. 一次消除获得 2 分，相邻两次消除时间小于 2s 会有加倍得分；
2. 普通游戏玩家点击提示会消除一对图片，VIP 玩家会消除多对；
3. 点击洗牌会对图片随机排序；
4. 点击退出返回到主界面。

4.1.3 游戏设置

在菜单界面点击游戏设计，进入游戏设置界面如图4所示，可选择游戏难度，即游戏人物数量，默认为 13 个英雄。对应地，游戏难度越高，连续消除的额外得分也越高。

图 4: 游戏设置界面

4.1.4 游戏排名

在菜单界面点击游戏设计，进入游戏设置界面如图5所示。

4.1.5 游戏充值

在菜单界面点击充值，游戏中基础得分 100 分，英雄获得全皮肤，点击提示有多对消除。再次点击退款可能回到普通模式。

图 5: 游戏排名界面

4.2 扩展功能

用户可替换./fig 文件夹中的图片，定制属于自己的练练看。注意图片总数和命名不能改变。

5 功能性要求得分清单

1. 基本功能: 如前文所示，实现基本功能；
2. 扩展功能: 背景音乐和音效，进入游戏后开始播放王者荣耀歌曲；每次完成游戏后英雄人数会增加 2，直到 17 个英雄；
3. 自定义功能: 附加充值功能。