彩球实验报告

同济大学

班级：计算机科学与技术2班

学号：2354218

姓名：肖佳彤

完成日期：2025年1月1日

1. 题目

1.1.彩球游戏规则概述：

1、游戏区域为 9*9，共有7种颜色的彩球随机出现，其中初始状态5个，以后每次出3个
2、按 F3 键可以预告下次出现的三个彩球的颜色，再按一次则关闭预告
3、按 F5 键可以在侧边显示当前状态统计，再按一次则关闭统计信息
4、用鼠标选中某个彩球，再选择一个空白区域作为目标位置，如果从源到目标位置有通路可达将彩球移动到目标位置:如果无通路可达，则不移动并给出相应提示
5、当同色彩球在横向、纵向、斜向达到5个及以上时，可以消除，同时得到相应的分数
6、如果本次消除则不产生新球，否则产生三个新球，颜色随机
在c++中实现以上功能

1.2.具体实现部分：

菜单项A：输入行列后，在规定范围内随机生成五个球的位置，然后打印整个内部数组为方便观察，打印时有球的位置用不同颜色输出。

菜单项B:输入行列后，在规定范围内随机生成 60%的球的位置，然后输入要移动球的起始坐标及目的坐标，找出将球移动过去的路径起始位置必须有球，目的位置必须为空生成过程中，如果该位置已经有球，要重新生成。

菜单项C:结合菜单项1和2，完成一个完整的实现过程(纯内部数组表现形式)球的位置用不同颜色标出。连续 5 个则消除，并可以得分，得分为(n-1)*(n-2)，双五连等情况，交叉点要重复计数。本次移动若得分，则不产生新球。

菜单项D:在cmd 伪图形界面上画出框架(无分隔线)及初始的五个球demo。

菜单项E:在cmd 伪图形界面上画出框架(有分隔线)及初始的五个球。。

菜单项F:在 cmd 伪图形界面上显示 60%的球，用鼠标选择要移动的球及目的位置，完成一个移动鼠标操作为:左键选择，右键退出本小题。鼠标移动过程中，要实时显示当前移动到n*n 矩阵的哪个位置(行:A-I，列:1-9)，放在边框线上不算。移动过程需要完整的移动轨迹显示，动画效果必须跨越分隔线。

菜单项 G:在 cmd 伪图形界面上实现完整版的游戏，右侧显示得分、预告彩球以及各色球的消除总数统计。

2. 整体设计思路

2.1.项目文件组成

2.1.1.cmd_console_tools.cpp，cmd console tools.h

cmd伪图形界面工具集以及头文件，包含图形显示所需的函数。

2.1.2. 90-b2-main.cpp

包含主函数和菜单函数，从主函数中调用各项子菜单。

2.1.3. 90-b2-base.cpp

存放内部数组方式实现的各个函数。包含菜单项A,B,C和打印内部数组函数。

2.1.4. 90-b2-console.cpp

存放cmd图形界面方式实现的各个函数。包含菜单项D,E,F,G和图形显示初始化函数，边栏显示和统计函数和图像移动相关函数，打印内部数组。

2.1.5. 90-b2-tools.cpp

存放公用的函数，供其他文件调用。包含判断结束函数，输入输出函数，内部数组初始化函数，随机生成小球函数，寻路函数和积分及清除函数。

2.1.6. 90-b2.h

放置源文件程序的公用声明和全局常量。

2.2.设计思路概述

在主程序中，首先通过调用 main 函数启动整个程序。在 main 函数中会调用一个专门的菜单函数，用于展示用户可以选择的操作。而菜单函数则进一步调用存放在其他文件中的各个菜单项函数，以实现功能模块化和代码分离。整个程序的核心逻辑是通过一个 9x9 的二维数组来模拟彩球游戏的棋盘，其中每个元素代表一个彩球或空位。棋盘的有效范围由用户输入的内部数组大小决定，从而动态调整二维数组的可操作范围。

彩球以编号 1-7 进行区分，不同的编号对应不同的颜色。程序使用伪随机数生成初始棋盘，并在游戏运行过程中生成新的彩球。通过输入函数，决定棋盘的大小并控制有效操作范围。

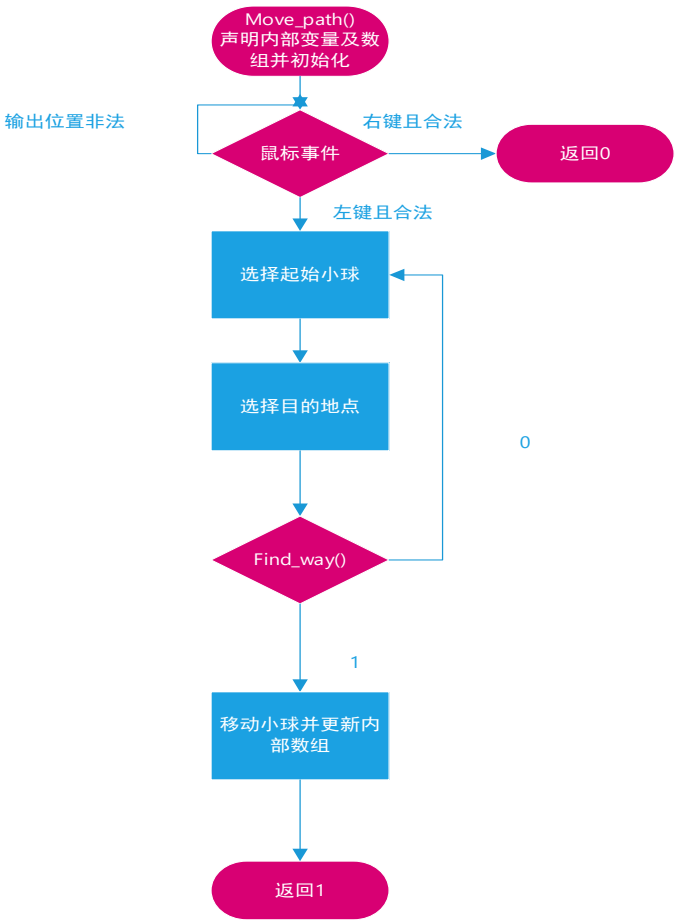
游戏的核心功能是实现小球的移动和消除。首先，程序进行图形界面显示初始化和内部数组的初始

化，随后进入循环。循环首先得到下一步生成的3个小球的颜色并显示，然后调用路径移动函数move_path()，使用鼠标事件选中起始小球和目的位置。在路径移动函数中，使用findway()函数查找移动路径。程序使用深度优先搜索（DFS）算法来寻找从起点到终点的有效路径。DFS 算法会将搜索到的路径保存在两个专用路径数组中以支持后续的动画显示。如果路径搜索函数返回值无效，则提示用户重新输入路径；否则，程序会调用move_a_path()函数结合生成的路径数组将小球移动到指定位置，并更新内部数组。退出路径移动函数后，判断移动后的棋盘是否形成五个或更多同色小球的连线。如果形成连线，则触发消除逻辑，清除这些小球并计算得到的分数以及将消除的小球记录在消除记录数组中；如果没有形成连线，则随机生成三个新的彩球并打印，并再次判断是否形成连线。随后进入下一个循环，更新侧栏的显示内容。如果在调用生成三个小球的函数时发现内部数组满了，则返回一个值，让程序退出循环并结束游戏。

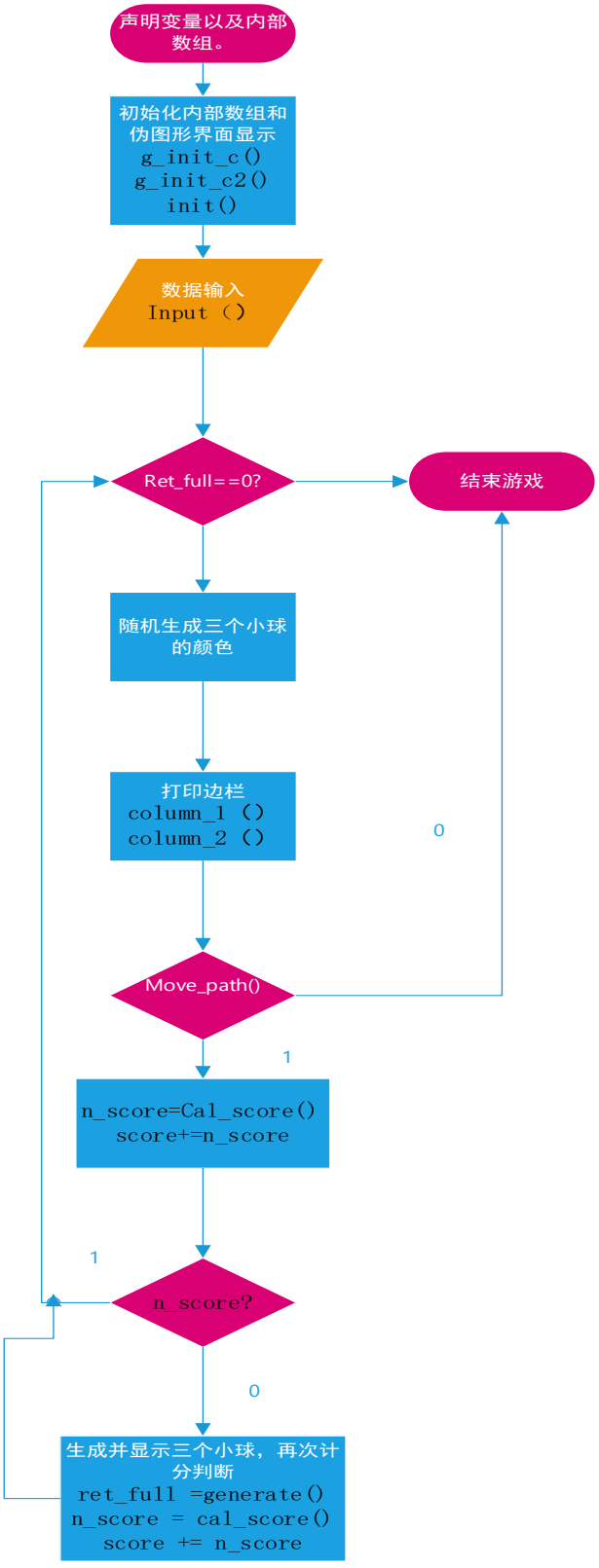
在伪图形显示部分，为了避免每次移动后整体屏闪，以及为了提高程序的灵活性，部分函数通过参数控制输出方式，可以选择以内部数组的形式输出或以伪图形形式输出。在内部数组生成时选择是否打印。

3. 主要功能的实现

3.1.Move_path()函数内部实现



3.2. 总体流程图


声明变量以及内部数组。
初始化内部数组和伪图形界面显示
g_init_c()
g_init_c2()
init()
数据输入
Input ()
Ret_full==0?
结束游戏
随机生成三个小球的颜色
打印边栏
column_1 ()
column_2 ()
0
Move_path()
1
n_score=Cal_score()
score+=n_score
1
n_score?
0
生成并显示三个小球，再次计分判断
ret_full =generate()
n_score = cal_score()
score += n_score

4. 调试过程碰到的问题

4.1.问题一

菜单项G中选中加粗后，前一个被移动的小球仍然是特殊显示模式。解决方法：将目的位置设置为静态局部变量，下一次循环，鼠标事件选中移动小球后，更新上一次被移动的小球。这样带来另一个问题，即如果是第一次显示和如果上一次移动的小球被消除，会产生显示错误的问题，解决办法：设置一个标识flag，当没有发生消除活动时flag=1,清除上一次的小球特殊标记，否则flag=0。

4.1.问题二

在编写路径寻找算法时，最初采用的是递归方法。递归虽然直观且实现简单，但在路径较为复杂或棋盘较大时，容易出现栈溢出的问题，导致算法运行失败。为了解决这一问题，对算法进行了改进，使用非递归的方法替代递归，改用深度优先搜索（DFS）实现。非递归的 DFS 方法不仅解决了递归深度过大可能引发的栈溢出问题，还能够更高效地处理较复杂的路径寻找任务。此外，DFS 也能兼顾路径的搜索深度，适用于宽广的搜索空间，为路径寻找提供了可靠的解决方案。

4.1.问题三

使用cin.getline()时经常会出现还未输入就已经显示输入错误，请重新输入的情况，或是在while循环中出现只执行一次的情况。经过检查，发现第一个问题发现是上一次输如残留了换行符，解决办法：在输入行列数后用cin,ignore()清除缓冲区。第二个问题是因为，输入错误后cin状态位为fail,解决办法：在输入错误后用cin.clear()清除错误状态位。

5. 心得体会

5.1.心得体会与经验教训

这次的彩球作业作业量不小，整个程序加起来将近有一千多行。思考量很大，断断续续花了整整两周时间完成了彩球游戏的程序和报告。通过这次作业，我收获了很多，不仅在编程技巧上有了进步，也对开发过程中的一些细节和心态有了更深的理解。接下来我就简单总结一下我的收获和体会。

首先，在实现彩球游戏时，我学会了如何划分程序的各个部分。刚开始我也有点迷茫，不知道从哪里入手。最后，我决定按照游戏的不同模块来分块处理，主要分为两大部分：一个是核心逻辑部分，另一个是界面和动画部分。为了实现代码的清晰和可维护性，我还根据模块需求将不同的功能分散到了多个 .cpp 文件中。这样做不仅让代码更清晰，而且能避免不同功能的耦合，方便后期维护。

然后，在这次作业中，我对一些工具函数的使用也变得更加熟练了，尤其是涉及到输入输出的部分。在实现游戏时，需要处理很多用户输入，比如选择棋盘的大小、控制小球的移动以及定位等，这些操作都需要细致的输入处理。特别是在实现动画和游戏界面时，涉及到控制台字符颜色的变化、界面的

动态刷新等，这些都离不开一些特殊的工具函数。通过这次作业，我对这些函数的应用更加熟悉了，也更加明白了如何根据需求调整控制台显示的效果。

再者，鼠标操作的部分是我这次作业的一大收获。在彩球游戏中，玩家需要用鼠标来点击不同的位置来操作小球，这就要求我能够灵活地处理鼠标输入。鼠标点击、右键菜单等功能的实现一开始让我感觉有点复杂，但经过一段时间的摸索，终于掌握了相关的技巧。通过这一部分的实践，我对鼠标事件的处理有了更直观的理解，也能更加高效地处理类似的输入方式了。

另外，这次作业让我深刻体会到，编写复杂程序时，合理分工和模块化是非常重要的。模块化的思路让我能够在主函数中调用各个功能函数，清晰明了，也便于调试和扩展。通过函数的拆分，代码变得更加整洁，而且每个模块的功能更加清晰，也减少了出错的几率。

最后，最重要的一点就是，这次作业让我学会了如何调整自己的心态，面对代码时更加细心。编程过程中，调试是不可避免的，尤其是当程序出现问题时，很容易陷入迷茫的状态。但通过这次作业，我学会了耐心地分析问题，逐步定位错误，逐个排除可能的原因，最终找到问题所在。设置断点、查看变量值，逐步调试，都是我在这次作业中积累的宝贵经验。

总的来说，这次的彩球作业让我在编程技能、解决问题的思路以及调试技巧方面都有了很大的提升。虽然过程很辛苦，但也充满了挑战和乐趣。

5.2.函数分解与使用

在完成彩球作业和汉诺塔程序时，我注重了代码的整体框架设计以及函数的复用。在一开始，我先对彩球作业和汉诺塔问题的逻辑结构进行了全面的分析，明确了任务中核心的功能需求以及各部分之间的联系。同时在各个函数中增添形参能更好复用代码。例如在彩球程序中，最开始generate()函数只进行更新内部数组的操作，后来在图形显示方式中，需要在伪图形页面中显示随机生成的三个小球，于是我在之前的函数中添加一个参数来控制是否显示图形。

同样在汉诺塔问题和彩球问题中，都需要先进行内部数组的编程，再进行图形化显示。所以在内部数组方式编程过程中我都注意到在图形化方式显示编程过程中代码的复用问题。例如在汉诺塔问题中，只使用了一个递归算法。在彩球问题中图形化显示使用了相同的内部数组操作。

对于如何更好地重用代码，首先，还是需要对程序的整体框架有大概的想法，理解问题的操作逻辑，找到不同功能之间的相似点。然后，在分析问题时，也要注意分层分块的思想，先在大体上对程序的模块有把握再去细化各个模块，当然如果没有把握也可以边写边想，在后续需要用到相似的功能时。再添加参数，使得函数能够重用。

代码重用是开发过程中特别重要的一件事，不仅能节省时间，还能减少出错的概率，同时让代码更容易维护。实验中发现，如果把代码拆成一个个独立的小模块，每个模块只做一件事情，那么后续再用到类似功能的时候，直接调用这些模块就行了，不用重复写。这种模块化的设计，大大提高了代码的灵活性和效率。

附件：源程序

```
void g_init(int x,int y,int shuzu[9][9],int
line,int col)
{
    cct_showstr(x, y, "╔", COLOR_HWHITE, 0);
    cct_showstr(x + 2, y, "═", COLOR_HWHITE, 0,
col);
    cct_showstr(x + 2+col*2, y, "╗",
COLOR_HWHITE, 0 );

    for (int i = 0; i < line; i++) {
        cct_showstr(x, y+i+1, "║",
COLOR_HWHITE, 0);
        cct_showstr(x + 2 + col * 2, y + i + 1,
"║", COLOR_HWHITE, 0);
    }
    cct_showstr(x, y+line+1, "╚", COLOR_HWHITE,
0);
    cct_showstr(x + 2, y+line+1, "═",
COLOR_HWHITE, 0, col);
    cct_showstr(x + 2 + col * 2, y+line+1, "╝",
COLOR_HWHITE, 0);
  //框架
    for (int i = 0; i < line; i++) {
        for (int j = 0; j < col; j++) {
            if (shuzu[i][j] == 0)
                cct_showstr(x + 2 + 2*j ,y + 1
+ i, "○", COLOR_HWHITE, COLOR_HWHITE);
            else
                cct_showstr(x + 2 + 2*j,y + 1
+  i , "○", shuzu[i][j], COLOR_HWHITE);
        }
    }
    cct_setcolor();
    cout << endl << endl;
}

void g_init_c(int x, int y, int line, int col)
{
    int nx, ny;
    cct_showstr(x, y, "╔", COLOR_HWHITE, 0);
    cct_showstr(x + 2, y, "═══╦", COLOR_HWHITE,
0, col-1);
    cct_getxy(nx,ny);
    cct_showstr(nx, ny, "═╗", COLOR_HWHITE,
0);
    for (int i = 0; i < 2*line-1; i++) {
        if (i % 2 == 0) {
            cct_showstr(x, y + 1+i, "║  ",
COLOR_HWHITE, 0, col);
            cct_getxy(nx, ny);
            cct_showstr(nx, ny, "║",
COLOR_HWHITE, 0);
        }
        else {
            cct_showstr(x, y + 1 + i, "╠",
COLOR_HWHITE, 0);
            cct_showstr(x+2, y + 1 + i, "═══╬",
COLOR_HWHITE, 0,col-1);
            cct_getxy(nx, ny);
            cct_showstr(nx, ny, "═╣",
COLOR_HWHITE, 0);
        }
    }
    cct_showstr(x, y+2*line, "╚", COLOR_HWHITE,
0);
    cct_showstr(x + 2, y+2*line, "═══╩",
COLOR_HWHITE, 0, col - 1);
    cct_getxy(nx, ny);
    cct_showstr(nx, ny, "═══╝", COLOR_HWHITE,
0);
}

void g_init_c2(int x, int y, int shuzu[9][9], int
line, int col)
{
    for (int i = 0; i < line; i++) {
        for (int j = 0; j < col; j++) {
            if (shuzu[i][j] != 0)
                cct_showstr(x + 2 + 4 * j, y +
1 + 2 * i, "○", shuzu[i][j], COLOR_HWHITE);
        }
    }
    cct_setcolor();
}

//移动一格
void move_a_step(int nr, int nc, int er, int
ec,int color)
{
    int dr = er - nr, dc = ec - nc;
    if (dr) {
        Sleep(200);
        cct_showstr(4 * nc + 2, 2 * nr + 2, "
", COLOR_HWHITE, COLOR_HWHITE);
        cct_showstr(4 * nc + 2 , 2 * nr + 2+dr,
"◎",color, COLOR_HWHITE);
        Sleep(200);
        cct_showstr(4 * nc + 2, 2 * nr + 2+dr, "
═", COLOR_HWHITE, 0);
        cct_showstr(4 * nc + 2, 2 * nr + 2+2*dr,
"◎", color, COLOR_HWHITE);
    }
    if (dc) {
        Sleep(200);
        cct_showstr(4 * nc + 2, 2 * nr + 2, "
", COLOR_HWHITE, COLOR_HWHITE);
        cct_showstr(4 * nc + 2+2*dc, 2 * nr + 2,
"◎", color, COLOR_HWHITE);
        Sleep(200);
        cct_showstr(4 * nc + 2+2*dc, 2 * nr + 2,
```

```
"‖", COLOR_HWHITE, 0);
        cct_showstr(4 * nc + 2+4*dc,2 * nr + 2,
"◎", color, COLOR_HWHITE);
    }
}
//图形移动一步
bool move_path(int line, int col, int
shuzu[][9],int flag2)
{
    int X = 0, Y = 0;
    int ret, maction, flag = 0;
    int keycode1, keycode2;
    static int start_c, start_r, end_c, end_r;
    char result[9][9] = { };
    int path_x[100], path_y[100], path_length =
0;
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            result[i][j] = '0';
        }
    }
    while (1) {
        ret = cct_read_keyboard_and_mouse(X, Y,
maction, keycode1, keycode2);
        if (ret == CCT_MOUSE_EVENT) {
            cct_gotoxy(0, 2 * line + 2);
            cout << "[当前光标] ";
            if ((X % 4 == 2 || X % 4 == 3) && X
< (2 + 4 * col) && Y % 2 == 0 && Y>0 && Y < (2 +
line * 2)) {//鼠标有效
                int nc = (X + 2) / 4;
                int nl = Y / 2;
                cout << (char)('A' + nl - 1)
<< "行" << nc << "列            ";
                if (maction ==
MOUSE_LEFT_BUTTON_CLICK && shuzu[nl - 1][nc - 1])
{
                    if (flag == 1)
                        cct_showstr(start_c *
4 + 2, start_r * 2 + 2, "○",
shuzu[start_r][start_c], COLOR_HWHITE);
                    if (flag2)
                        cct_showstr(end_c * 4
+ 2, end_r * 2 + 2, "○", shuzu[end_r][end_c],
COLOR_HWHITE);
                    start_c = nc - 1;
                    start_r = nl - 1;
                    cct_showstr(nc * 4 - 2, nl
* 2, "◎", shuzu[start_r][start_c],
COLOR_HWHITE);
                    cct_setcolor();
                    flag = 1;
                }
                if (maction ==
MOUSE_LEFT_BUTTON_CLICK && !shuzu[nl - 1][nc - 1]
&& flag == 1) {
                    end_c = nc - 1;
                    end_r = nl - 1;
                    ret = findway(shuzu,
result, start_r, start_c, end_r, end_c, line,
col, path_x, path_y, path_length);
                    if (ret) {
                        cct_gotoxy(0, 2 *
line + 2);
                        cout << "[提示]可以从
["<< (char)('A' + start_r)<< start_c+1<<"]移动到
["<< (char)('A' + end_r)<< end_c + 1<<"]";
                        for (int i =
path_length - 1; i > 0; i--)

    move_a_step(path_x[i], path_y[i], path_x[i -
1], path_y[i - 1], shuzu[start_r][start_c]);
                        shuzu[end_r][end_c] =
shuzu[start_r][start_c];

    shuzu[start_r][start_c] = 0;
                        cct_setcolor();
                        cct_gotoxy(0, 2 *
line + 2);
                        return 1;
                    }
                    else {
                        cct_gotoxy(0, 2 *
line + 2);
                        cout << "[错误] 无法
从[" << (char)('A' + start_r) << start_c + 1 <<
"]移动到[" << (char)('A' + end_r) << end_c + 1 <<
"]";
                    }
                }
                if (maction ==
MOUSE_RIGHT_BUTTON_CLICK) {
                    cct_gotoxy(0, 2 * line +
2);
                    return  0;
                }
            }
            else
                cout << "位置非法
";
        }
    }
}

void tongji(int shuzu[][9],int line,int col,int
ret[8])
{
    for (int i = 0; i < line; i++) {
        for (int j = 0; j < col; j++) {
            ret[shuzu[i][j]]++;
        }
    }
}
void column_1(int shuzu[][9],int rand3[], int
```

```
 score,int line,int col,int x = 40, int y = 1)
{
    cct_showstr(x, y,      "╔═══════════╗",
COLOR_HWHITE, COLOR_BLACK);
    cct_showstr(x, y + 1, "║得分：      ║",
COLOR_HWHITE, COLOR_BLACK);
    cct_showint(x + 8, y + 1, score,
COLOR_HWHITE, COLOR_BLACK);
    cct_showstr(x, y+2,    "╚═══════════╝",
COLOR_HWHITE, COLOR_BLACK);

    cct_showstr(x, y + 4, "╔═╦═╦═╗",
COLOR_HWHITE, COLOR_BLACK);
    cct_showstr(x, y + 5, "║ ║ ║ ║",
COLOR_HWHITE, COLOR_BLACK);
    cct_showstr(x, y + 6, "╚═╩═╩═╝",
COLOR_HWHITE, COLOR_BLACK);
    cct_showstr(x+2, y + 5, "○", rand3[0],
COLOR_HWHITE);
    cct_showstr(x + 6, y + 5, "○", rand3[1],
COLOR_HWHITE);
    cct_showstr(x + 10, y + 5, "○", rand3[2],
COLOR_HWHITE);

}

void column_2(int shuzu[][9], int xiaochu[8], int
score, int line, int col, int x = 40, int y = 1)
{
    int ret[8] = { 0 };
    tongji(shuzu, line, col, ret);
    cct_showstr(x, y + 8, "╔═══════════
═══════╗", COLOR_HWHITE, COLOR_BLACK);
    for (int i = 0; i < 8; i++)
    {
        cct_showstr(x, y + 9 + i, "╠
╣", COLOR_HWHITE, COLOR_BLACK);
        if (i == 0)
            cct_showstr(x + 2, y + 9 + i, "○",
COLOR_HWHITE, COLOR_HWHITE);
        else {
            cct_showstr(x + 2, y + 9 + i, "○",
i, COLOR_HWHITE);
        }
        cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
        cout << setfill('0') << setw(2) <<
ret[i] << "/(" << fixed << setprecision(2) <<
setw(5) << ((double)ret[i] / (line * col)) * 100
<< "%) 消除-"<<xiaochu[i];
    };
    cct_showstr(x, y + 17, "╚═══════════
═══════╝", COLOR_HWHITE, COLOR_BLACK);
    cct_setcolor();
}

void choice_f()
{
    cct_enable_mouse();
    int line, col, n;
   bool flag=0;
    int shuzu[9][9] = { 0 };
    cct_cls();
    input(line, col);
    n = int(line * col * 0.6);
    init(shuzu, line, col, n);
    cct_setcursor(CURSOR_INVISIBLE);
    cct_showstr(0, 0, "鼠标移动，左键单击选择，
右键退出");
    g_init_c(0, 1 , line, col);
    g_init_c2(0, 1, shuzu, line, col);
    move_path(line, col, shuzu,flag=0);
}

void choice_g()
{
    cct_enable_mouse();
    int line, col,score=0;
    int shuzu[9][9] = { 0 };
    int rand3[3] = { 0 }, xiaochu[8] = { 0 };
    bool ret_full = 1,flag2=0;
    cct_cls();
    input(line, col);
    init(shuzu, line, col, 5);
    cct_setcursor(CURSOR_INVISIBLE);
    cct_showstr(0, 0, "鼠标移动，左键单击选择，
右键退出");
    g_init_c(0, 1, line, col);
    g_init_c2(0, 1, shuzu, line, col);
    while (1) {
        int n_score;
        for (int i = 0; i < 3; i++) {
            rand3[i] = rand() % 7 + 1;
        }
        if (ret_full == 0)
            break;
        column_1(shuzu, rand3, score, line,
col);
        column_2(shuzu, xiaochu, score, line,
col);

        if (move_path(line, col, shuzu, flag2))
{
            flag2= 0;
            n_score = cal_score(shuzu, line,
col, xiaochu, 1);
            score += n_score;
            if (!n_score) {
                ret_full = generate(shuzu,
rand3, line, col, 1);
                n_score = cal_score(shuzu,
line, col, xiaochu, 1);
                score += n_score;
                flag2 = 1;
```

```
                }
            }
            else
                return;
        }
        cct_gotoxy(0, 2 * line + 2);
        cct_setcolor();
        return;
}

//内部数组，初始生成 n 小球
void init(int shuzu[9][9], int line, int col, int n)
{
    int r_line, r_col, r_value, i = 0;
    while (i < n)
    {
        r_line = rand() % line;
        r_col = rand() % col;
        r_value = rand() % 7;
        if (shuzu[r_line][r_col] == 0) {
            shuzu[r_line][r_col] = r_value + 1;
            i++;
        }
    }
}
//生成 3 个小球,并判断是否数组满，满则返回 0
bool generate(int shuzu[][9], int rand3[], int line, int col,int g)
{
    int r_line, r_col, k = 0;
    bool flag = 0;
    for (int i = 0; i < line; i++) {
        for (int j = 0; j < col; j++) {
            if (shuzu[i][j] == 0) {
                flag = 1;
            }
        }
    }
    while (k < 3 && flag)
    {
        flag = 0;
        for (int i = 0; i < line; i++) {
            for (int j = 0; j < col; j++) {
                if (shuzu[i][j] == 0) {
                    flag = 1;
                }
            }
        }
        if (flag) {
            r_line = rand() % line;
            r_col = rand() % col;
            if (shuzu[r_line][r_col] == 0) {
                shuzu[r_line][r_col] = rand3[k];
                if (g) {
                    cct_showstr(4 * r_col + 2, 2 *
r_line + 2, "○", rand3[k], COLOR_HWHITE);
                }
                k++;
            }
        }
    }
    flag = 0;
    for (int i = 0; i < line; i++) {
        for (int j = 0; j < col; j++) {
            if (shuzu[i][j] == 0) {
                flag = 1;
            }
        }
    }
    return flag;
}

bool findway(int shuzu[][9], char result[][9], int sr, int sc, int
er, int ec, int line, int col, int path_x[], int path_y[], int&
path_length) {
    int dr[] = { -1, 1, 0, 0 };
    int dc[] = { 0, 0, -1, 1 };
    bool visited[9][9] = { false };
    int queue_x[100], queue_y[100];
    int front = 0, rear = 0; // 队列的头尾指针
    int parent_x[9][9];
    int parent_y[9][9];
    for (int i = 0; i < line; ++i) {
        for (int j = 0; j < col; ++j) {
            parent_x[i][j] = -1;
            parent_y[i][j] = -1;
        }
    }
    queue_x[rear] = sr;
    queue_y[rear] = sc;
    rear++;
    visited[sr][sc] = true;

    while (front < rear) {
        int x = queue_x[front];
        int y = queue_y[front];
        front++;

        if (x == er && y == ec) {
            int px = x, py = y;
            while (px != -1 && py != -1) {
                path_x[path_length] = px;
                path_y[path_length] = py;
                path_length++;
                int tmp_x = parent_x[px][py];
                int tmp_y = parent_y[px][py];
                px = tmp_x;
                py = tmp_y;
            }
            return true;
        }

        for (int i = 0; i < 4; ++i) {
            int nr = x + dr[i];
            int nc = y + dc[i];

            if (nr >= 0 && nr < line && nc >= 0 && nc
< col && !visited[nr][nc] && shuzu[nr][nc] == 0) {
                queue_x[rear] = nr;
                queue_y[rear] = nc;
                rear++;
                visited[nr][nc] = true;
                parent_x[nr][nc] = x;
                parent_y[nr][nc] = y;
            }
        }
```

```
	}
	return false;
}


//计算分数，并将连续 5 个以上置零，返回值是是否有计分活动
int cal_score(int shuzu[][9], int line, int col,int xiaochu[8],int g)
{
	int score = 0, flag1 = 0, flag2 = 0,count_total = 0,count1, count2, color;
	int i1, j1, i2, j2;;
	for (i1 = 0; i1 < line; ++i1) {
		for (j1 = 0; j1 <= col - 5; ++j1) {
			count1 = 1;
			if (shuzu[i1][j1] != 0) {
				while (j1 + count1 < col && shuzu[i1][j1 + count1] == shuzu[i1][j1])
					count1++;
				if (count1 >= 5) {
					count_total += count1;
					flag1 = 1;
					color = shuzu[i1][j1];
					break;
				}
			}
		}
		if (flag1)
			break;
	}//行
	for (j2 = 0; j2 < col; ++j2) {
		for (i2 = 0; i2 <= line - 5; ++i2) {
			count2 = 1;
			if (shuzu[i2][j2] != 0) {
				while (i2 + count2 < line && shuzu[i2 + count2][j2] == shuzu[i2][j2])
					count2++;
				if (count2 >= 5) {
					if (flag1)
						count_total = count_total + count2 - 1;
					else
						count_total += count2;
					flag2 = 1;
					color = shuzu[i2][j2];
					break;
				}
			}

		}
		if (flag2)
			break;
	}//列
	if (flag1) {
		for (int k = 0; k < count1; ++k) {
			shuzu[i1][j1 + k] = 0;
			if (g) {
				Sleep(20);
				cct_showstr(4 * (j1 + k) + 2, 2 * i1 + 2, "  ", COLOR_HWHITE, COLOR_HWHITE);
			}
		}
	}
	if (flag2) {
		for (int k = 0; k < count2; ++k) {
			shuzu[i2 + k][j2] = 0;
			if (g) {
				Sleep(20);
				cct_showstr(4 * j2 + 2, 2 * (i2 + k) + 2, "  ", COLOR_HWHITE, COLOR_HWHITE);
			}
		}
	}
	//置零
	if (count_total) {
		xiaochu[color] += count_total;
		score = (count_total - 1) * (count_total - 2);
	}
	return score;
}
```