Magic

彩球综合实验报告

学号：2351050

姓名：杨瑞晨

班级：信05/02班

完成日期：24/06/13

Ball

1. 题目分析

1.1 题目综述

本次实验要求设计并实现一个彩球游戏（Magic Ball），通过综合运用 C++ 编程技巧，模拟游戏的各项功能。游戏应包括彩球的生成、运动、碰撞检测等基本功能，并通过伪图形界面进行展示。具体要求如下：

- 游戏区域为 5*5 - 9*9 间的任意矩形，共有 9 种颜色的彩球随机出现，初始随机填满，随机概率相同。
- 可消除项球为实心球，可互换球为空心双圈球，普通球为空心单圈球。球的颜色通过背景色区分。
- 横向或纵向连续三个及以上同色球会消除，每消除一个球计 1 分，如果同时存在横纵向消除，则分别计算。
- 消除后，空位上方的球垂直方向落下，最上方的空位由随机颜色的彩球填满。
- 初始填满后，先判断是否存在可消除项，若有，则立即进行消除、填充，循环往复，直到无可消除项为止（停止前的所有消除项不计分）。
- 无可消除项后，遍历整个数组，将可互换的球用圈标识出来。
- 无可消除项后，标识可互换的球，用户可通过鼠标选择（再按一次则取消选择），再选择邻近的可互换的球即可进行交换；交换后再次进行消除、填充，直到无消除项为止。

1.2 功能分析

菜单项共有10个，分别完成以下功能：

1) 生成初始数组状态并找出初始可消除项；
2) 在 1) 的基础上完成消除初始可消除项、非0项下落、用0填充、再填充新值；
3) 在 2) 的基础上查找相邻的可互换项；
4) 初始化图形界面-无分隔线；
5) 初始化图形界面-有分割线；
6) 初始化图形界面-无分隔线-找出初始可消除项；
7) 初始化图形界面-有分隔线-找出初始可消除项，消除后显示消除提示；
8) 图形界面-有分隔线-支持鼠标判断行列位置；
9) 图形界面-有分隔线-实现人工操作彩球消除，即最终游戏版本；
10) 退出游戏。

1.3 要求与限制

文件由 8 个文件组成， 2 个已给出(斜体标注)：

cmd_console_tools.h

magic_ball.h—头文件，用于引入其他头文件及宏定义，以及声明函数

cmd_console_tools.cpp

magic_ball_main.cpp—main函数

magic_ball_menu.cpp—显示菜单项及选择、完成输入

magic_ball_tools.cpp—各种可能用到的辅助工具函数

magic_ball_base.cpp—内部数组相关函数的实现

magic_ball_graph.cpp—图形界面相关函数的实现

不允许使用除基本结构、函数、数组、指针、引用、结构体以外的内容；

不允许使用 C++ 中的 string 功能；

不允许使用全局变量（全局const变量/宏定义不受限）；

其他要求：

- 使用参数解决差异，如是否有分割线、框线长短；
- 合理安排函数的分解及调用；
- 字体、字号、颜色等无强制要求，但需保证在 1920*1080 分辨率下显示正常；
- 边框为中文字符。

2. 整体设计思路

本实验的目标是设计并实现一个彩球游戏（Magic Ball），实现彩球的生成、移动、碰撞检测和消除等功能，并通过伪图形界面展示游戏过程。为此，我们需要详细规划程序的各个模块，确保每个功能都能高效实现，并且整体结构清晰。

2.1 彩球类设计

彩球的功能（颜色、是否为可消除项、是否为可互换项）可以通过三个二维数组实现（在学习结构体概念后，可将此三个数组作为一个结构体中的项）：

1) ball[10][10]:球的基础定义。取值范围为数字 1-9 ，代表球的不同颜色；
2) clear[10][10]:用于存储球是否为可消除项，1为可消除项；
3) hint[10][10]:用于存储球是否为可互换项，1为可互换项。

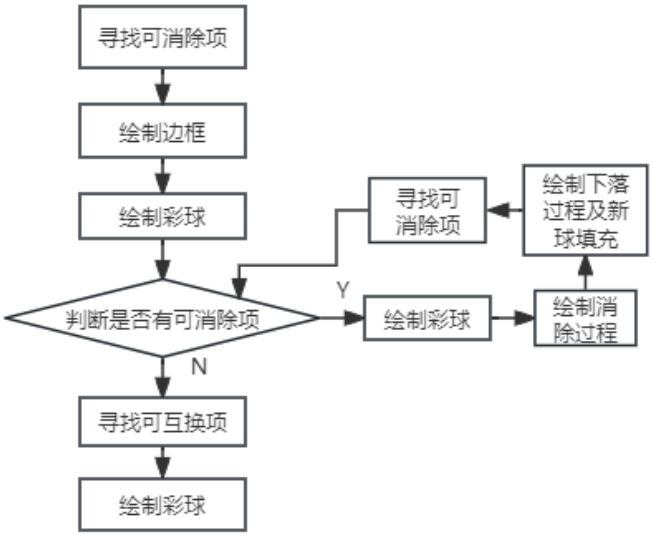
2.2 问题的解决思路

为方便解决问题，我们可以先将彩球抽象为数组，通过数组不同的值表示彩球不同的颜色；对应位置其他数组值为 0/1 分别对应其消除状态与互换状态。这样一来，我们便有了内部数组函数与绘制图形函数的一一对应，其内部逻辑相同，如：

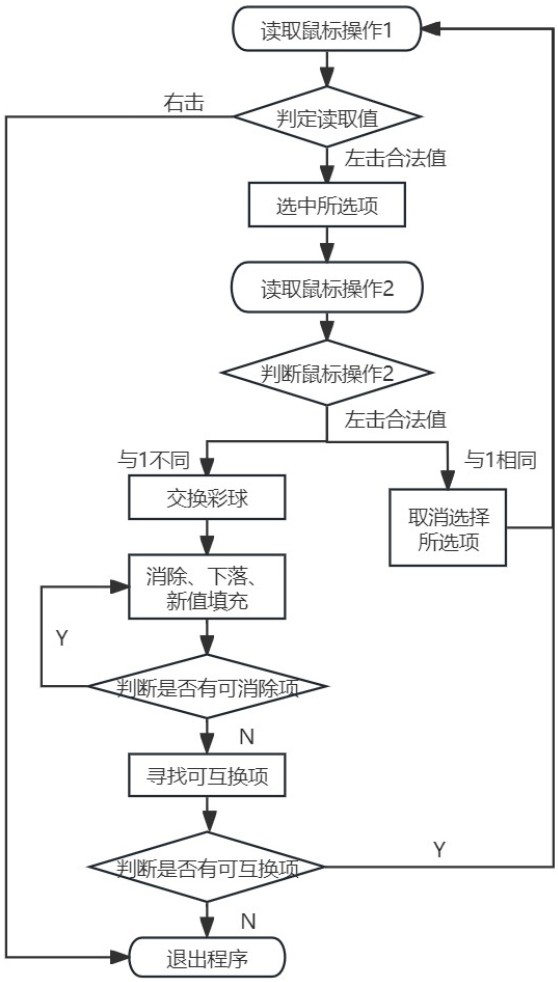
print_stack();	draw_ball();
find_clear();	draw_ball(); -> draw_clear();
find_hint();	draw_ball();
drop();	draw_drop();
fill_numbers();	draw_fill();

为便于编程求解彩球问题，游戏的核心逻辑通过一个主循环控制。该循环可由以下思维框图表示：

1） 初始化部分



2） 主循环部分



2.3 程序实现思路

2.3.1 头文件

头文件用于以下几个作用：

1) 引入其他头文件；
2) 定义 #define 宏定义及 const 型常变量；
3) 存放函数声明以便不同 cpp 文件间的互相访问；

2.3.2 main

main 文件用于初始化屏幕、调用菜单函数并返回菜单项以根据选项调用 magic_ball_menu 的 magic_ball 函数以完成不同的效果。

2.3.3 menu

menu 文件用于：

- 显示菜单并返回选项；
- 处理共同的输入内容；
- 根据 menu 的返回值调用各菜单项实现所需函数。

2.3.4 tools

工具函数组。主要有以下几个功能：

1）延时、等待回车键、处理输入 end 结束；
2）判断数组是否全为 0 ，可用于判断是否存在未消除项及可互换项，以判断游戏是否结束；
3）交换位置，判断交换后是否可构成可消除项，用于寻找可互换项；
4）处理鼠标的函数。

2.3.5 base

内部数组函数组。主要有以下几个功能：

1) 初始化球，生成随机颜色；
2) 打印内部数组，通过不同颜色表示可消除项及可互换项；
3) 寻找可互换项，遍历数组，将可互换项置为 1；
4) 寻找可消除项，遍历数组，将可消除项置为 1；
5) 掉落补位；
6) 填充 0 和重新生成随机数。

2.3.6 graph

伪图形函数组。主要有以下几个功能：

1) 绘制边框，通过参数决定是否有分割线；
2) 绘制球，通过球的空心、实心、双圈分别表示正常球、可消除项和可互换项；
3) 绘制清除过程；
4) 绘制掉落补位过程；
5) 绘制填充 0 和重新生成随机数。

2.3.7 cmd_console_tools

cct_ 系列函数，用于清屏、光标移动、图形化界面等。

3. 主要功能的实现

3.1 寻找可消除项

函数 find_clear() 的主要作用是检查游戏矩阵中是否存在可以消除的彩球，并标记这些彩球的位置，同时更新得分。函数接受五个参数：行数 row、列数 col、表示彩球颜色的二维数组 ball、标记可消除位置的二维数组 clear 以及指向当前得分的指针 score。具体实现如下：

1.清除 clear 数组

函数开始时，遍历整个 clear 数组，将所有元素设置为 0，表示初始状态下没有任何可消除的彩球。

2.检查行消除

函数检查每一行是否有三个或更多连续相同颜色的彩球。如果找到这样的序列，则将这些位置在 clear 数组中标记为 1，并更新得分。具体步骤如下：

- 遍历每一行，对于每一个元素 ball[i][j]，检查其与后面两个元素 ball[i][j+1] 和 ball[i][j+2] 是否相同。
- 如果相同，则继续向后检查，直到遇到不同的颜色或到达行末尾，并将这些相同颜色的位置在 clear 数组中标记为 1，同时增加得分。
- 跳过已检查过的部分，继续检查剩余部分。

3.检查列消除

类似地，函数接着检查每一列是否有三个或更多连续相同颜色的彩球。如果找到这样的序列，则将这些位置在 clear 数组中标记为 1，并更新得分。

3.2 值/球的下落操作

函数 drop() 的主要作用是处理游戏矩阵中被消除的彩球，并将上方的彩球掉落填补空位。函数接受四个参数：行数 row、列数 col、表示彩球颜色的二维数组 ball 以及标记消除位置的二维数组 clear。具体实现如下：

1.将被消除的彩球位置设置为0

函数首先遍历整个 clear 数组，将所有标记为 1 的位置对应的 ball 数组元素设置为 0，表示这些位置的彩球已被消除。

2.处理掉落补位

接着，函数遍历每一列，从下到上处理每一个位置，将非零的彩球掉落到该列的最下方空位。具体步骤如下：

- 对于每一列 j，设置写入位置 write_row 初始值为最后一行 row - 1。
- 从下到上遍历每一行 i，如果当前位置 ball[i][j] 非零，则将其值移动到 write_row 位置，并将 clear[write_row][j] 置为 0。
- 如果 write_row 和 i 不同，表示当前位置有空位，将其值设为 0，并将 clear[i][j] 置为 1。
- 每次移动后，更新写入位置 write_row，向上移动一行。

3.3 寻找可互换项

函数 find_hint() 的主要作用是检查游戏矩阵中是否存在可以通过一次交换形成匹配的彩球，并标记这些位置。函数接受四个参数：行数 row、列数 col、表示彩球颜色的二维数组 ball 和标记提示位置的二维数组 hint。具体实现如下：

1.清除 hint 数组

函数开始时，遍历整个 hint 数组，将所有元素设置为 0，表示初始状态下没有任何可通过交换形成匹配的提示。

2.检查每个元素与右边和下边相邻元素的交换是否能形成匹配

接着，函数遍历每个元素，检查其与右边和下边相邻元素的交换是否能形成匹配。如果可以形成匹配，则将这些位置在 hint 数组中标记为 1。具体步骤如下：

- 遍历每一行和每一列，对于每一个元素 ball[i][j]，分别检查其与右边元素 ball[i][j+1] 和下边元素 ball[i+1][j] 的交换是否能形成匹配。
- 使用 match 函数进行匹配检查，如果交换后可以形成匹配，则在 hint 数组中标记这两个位置为 1。

其中， match() 函数用于检查游戏矩阵中两个位置的彩球交换后是否能形成三个或更多相同颜色的彩球，从而确定该交换是否有效。函数接受六个参数：行数 row、列数 col、表示彩球颜色的二维数组 ball 以及两个交换位置的坐标 x1、y1 和 x2、y2。具体实现如下：

1.交换两个位置的值，以模拟玩家进行交换操作后的状态。

2.检查行匹配

接着，函数遍历每一行，检查是否存在三个或更多连续相同颜色的彩球。如果存在这样的序列，则设置 match 变量为 true，表示该交换有效。

3.检查列匹配

类似地，函数接着遍历每一列，检查是否存在三个或更多连续相同颜色的彩球。如果存在这样的序列，则设置 match 变量为 true。

4.恢复交换前的状态，将两个位置的彩球值交换回原来的状态，以确保游戏矩阵不被改变。

5.返回结果，函数返回布尔值 match，指示该交换是否能形成有效的匹配。

3.4 图形化操作

3.4.1 球的位置

通过计算，可以得到彩球实际输出的坐标值，将其设为 ballX, ballY, 其中，参数 divide 用于区别是否绘制了分割线。

```
int ballX = X + j * (divide + 1) * 2;
int ballY = Y + i * (divide + 1);
```

3.4.2 球的绘制

可以使用 cct_showstr() 函数来绘制不同状态的彩球。

```
const char* str = "○";("●", "◎"同理)
cct_showstr(ballX, ballY, str, *(ball[i] + j), COLOR_BLACK, 1);
```

3.4.3 球消除的动画模拟

定义两个符号用于绘制消除动画。

```
const char* str1 = "〇";
const char* str2 = "□";
```

设置循环，并适当添加延时，即可模拟消除动画效果。

```
if (*(clear[i] + j) == 1) {
    for (int k = 0; k < 5; k++) {
        cct_showstr(ballX, ballY, str2, *(ball[i] + j), COLOR_BLACK, 1);
        Sleep(20);
        cct_showstr(ballX, ballY, str1, *(ball[i] + j), COLOR_BLACK, 1);
        Sleep(50);
    }
    cct_showstr(ballX, ballY, "  ", COLOR_HWHITE, COLOR_HWHITE, 1);
    Sleep(20);
}
```

4. 调试过程碰到的问题

4.1 读取彩球坐标

在调试 read_mouse() 函数时，我们发现了一些潜在的问题，这些问题可能会影响游戏的正常运行和用户体验。具体问题包括：

1.光标位置判断不准确：

光标位置的合法性判断主要通过计算光标在游戏矩阵中的位置，并检查该位置是否在矩阵范围内。然而，由于边框的存在，光标可能会被误判为非法位置。

通过优化光标位置的合法性判断逻辑，确保准确计算光标在矩阵中的位置，并正确判断其合法性。

```
bool legal = false;
int boardY = (X - 1) / 4;
```

```
int boardX = (Y - 2) / 2;
if ((X - 1) % 4 == 0 || (X - 1) % 4 == 3 || (Y - 2) % 2 == 1) // 边框处
    legal = false;
else if (boardX >= 0 && boardX < row && boardY >= 0 && boardY < col)
    legal = true;
```

2.光标位置显示混乱：
当光标在非法位置时，显示的提示信息可能会覆盖合法选择的信息，导致用户困惑。
通过输出文字后面加适量空格，确保在每次显示新的提示信息前，清除旧的提示信息。
3.鼠标操作不响应
在某些情况下，鼠标点击操作可能会不响应，导致用户无法进行正常操作。
通过优化鼠标事件处理逻辑，确保每次鼠标点击操作都能被正确处理。

```
if (maction == MOUSE_LEFT_BUTTON_CLICK) {
    cct_gotoxy(0, length - 3);
    if (hint[boardX][boardY] == 1) {
        *x = boardX, *y = boardY;
        /* 当前选择 */
    }
    else {/*不能选择*/}
}
```

4.2 游戏开始前的分数计算

分数计算功能通过在 find_clear() 函数中传递进整形变量 score 的地址实现，每成功消除一个球 score 便自加：

```
int score = 0;
find_clear(row, col, ball, clear, &score);
void find_clear(int row, int col, int(*ball)[NUM_MAX], int(*clear)[NUM_MAX], int *score)
{clear[i][k] = 1; *score = *score + 1;}
```

然而在实际运行中，用户操作前初始生成的彩球中若有可消除项且成功消除也会计入进总分里，与游戏的实际行为不符。

通过定义 fake_score 变量，在用户操作前传递此变量便可规避该问题：

```
find_clear(row, col, ball, clear, &fake_score);
```

5. 心得体会

5.1 心得体会与经验总结

- 在调试过程中，学会利用输出日志和断点来检查程序的运行状态和变量的变化情况。通过这些调试技巧，能够快速发现和解决程序中的错误，确保程序的正确性和稳定性；
- 在开始编码之前，详细的设计思路和模块划分显得尤为重要。通过对游戏逻辑、图形界面、用户交互和分数计算等模块进行详细的规划，使得后续的编码过程更加清晰和有条理；
- 将程序划分为多个模块，每个模块负责特定的功能，不仅提高了代码的可读性和可维护性，还方便了调试和测试。尤其是在调试过程中，可以逐个模块地排查问题，快速定位和修复。
- 反复出现的常量应使用 const 常变量/#define 宏定义的方式写于头文件，便于后续维护；
- 合理注释，便于后期维护时自己和他人对代码的理解

- 在实现过程中，充分利用了已有的函数和逻辑，提高了代码的复用性。同时，通过优化代码逻辑和减少冗余操作，提高了程序的运行效率。

5.2 对复杂问题编写的体会

- 复杂问题往往组成庞杂，开始写之前须构建好主体的思路，否则无从下手。也不要想到哪就写到哪，不利于整体结构的构建；
- 善于利用不同的 cpp 文件化繁为简，善于使用头文件完成函数声明及宏定义与常变量的声明；
- 将问题模块化，分解为不同的部分，依据主要功能分成大的模块，再拆分成小的模块，最终组合在一起实现复杂功能；
- 遇到思路卡壳时，不妨绘制流程图以更好理解当前问题。

5.2.1 前后小题的关联关系

此次作业便是对各题前后联系与代码的有效重新利用的最好证明，有效减少了工作量。之前的问题循序渐进，逐渐深入，循循善诱，在后续的使用中只需适当增添参数即可。合理的代码复用，可大大减轻复杂程序的编写难度。

在设计和实现每个函数时，我都考虑到了其与其他函数的关联性。例如，在实现 find_clear 函数时，考虑到了后续 drop 函数对矩阵状态的更新需求，因此确保 clear 数组的正确性和完整性。同时，在实现 match 函数时，考虑到了 find_hint 函数的提示逻辑，通过统一的匹配判断逻辑，确保了代码的一致性和可靠性。

5.2.2 代码重用的实现

在实现过程中，通过模块化编程和函数封装，实现了代码的重用。例如，在 draw_ball 和 draw_clear 函数中，通过定义统一的绘制逻辑和符号，避免了重复代码的出现。这样不仅提高了代码的可读性和可维护性，还减少了开发和调试的工作量。

5.3 可改进的方向

此次编程过程中还有一些可以改进的地方：

- 当前的代码中，彩球的状态和位置信息分别保存在不同的数组中，这样在调用函数时需要传递多个参数，增加了函数调用的复杂性。可以通过将彩球及其状态抽象为一个结构体，来简化函数的参数传递，提高代码的可读性和维护性；
- 在当前的代码中，许多地方使用了写死的延时 Sleep(time)，这使得动画效果有时不够流畅，用户操作时需要间隔适当时间才能正常读取下一次操作。可以通过优化延时逻辑，使用动态延时来改善用户体验。

6. 附件：源程序

6.1 magic_ball_main

```
int main()
{
	while (1) {
		//
		const int select = menu();
		cout << endl;
		if (select == '0')
			return 0;
		magic_ball(select);//根据菜单项调用不同函数组合
		end();//输入end结束并返回菜单键
	}
	return 0;
}
```

6.2 magic_ball_menu

6.2.1 菜单项输出

```
cchar menu()
{
	//清屏
	//此处省略菜单项
	char ret;
	while (1) {
		ret = _getch();
		//错误处理
	cout << ret << endl;
	return ret;
}
```

6.2.2 输入函数

```
void input(int* row, int* col, char select)
{
	while (1) {
		cout << "请输入行数(5-9):" << endl;
		cin >> *row;
		//错误处理
	}
	while (1) {
		cout << "请输入列数(5-9):" << endl;
		cin >> *col;
		//错误处理
	}
}
```

6.2.3 菜单项处理函数

```
void magic_ball(char select)/
{
	cct_cls();
	int row, col;
	input(&row, &col, select);
	int ball[NUM_MAX][NUM_MAX] = { 0 };
	int clear[NUM_MAX][NUM_MAX] = { 0 };
	int hint[NUM_MAX][NUM_MAX] = { 0 };
	int score = 0;
	initial(row, col, ball, select);

	//此处省略菜单项 1-7

		if (select == '8' || select == '9') {
			cct_gotoxy(16, 0);
			cout << "(当前分数：" << score << " 右键退出)";
			int fake_score = 0;
			find_clear(row, col, ball, clear, &fake_score);
			draw_border(row, col, divide);
			draw_ball(row, col, divide, ball, clear, hint);
			while (!is_all_zero(row, col, clear)) {
				draw_ball(row, col, divide, ball, clear, hint);
				draw_clear(row, col, divide, ball, clear);
				cct_gotoxy(16, 0);
				//cout << "(当前分数：" << score << " 右键退出)";
				draw_drop(row, col, divide, ball, clear);
				Sleep(200);
				fill_numbers(row, col, ball, clear);
				draw_fill(row, col, divide, ball, clear);
				find_clear(row, col, ball, clear, &fake_score);
			}
			find_hint(row, col, ball, hint);
			draw_ball(row, col, divide, ball, clear, hint);
			//鼠标操作
			int X1 = -1, Y1 = -1, X2 = -1, Y2 = -1;
			while (1) {
				X1 = Y1 = X2 = Y2 = -1;
				int quit = read_mouse(row, col, ball, clear, hint, length, select, &X1, &Y1);
				if (quit == -1) {
					cct_gotoxy(0, length - 3);
					return;
				}
				if (X1 != -1 && Y1 != -1) {
					if (select == '8') {
						cct_gotoxy(0, length - 3);
						return;
					}
```

```
                        cct_showstr(2 + Y1 * (divide
+ 1) * 2, 2 + X1 * (divide + 1), "◎", ball[X1][Y1],
COLOR_HWHITE, 1);
                        cct_setcolor();
                        read_mouse(row, col, ball,
clear, hint, length, select, &X2, &Y2);
                        if (X2 == X1 && Y2 == Y1) {
                            cct_showstr(2 + Y1 *
(divide + 1) * 2, 2 + X1 * (divide + 1), "◎",
ball[X1][Y1], COLOR_BLACK, 1);
                            cct_setcolor();
                            cct_gotoxy(0, length -
3);
                            cout << "取消选择" <<
char(X1 + 'A') << "行" << Y1 + 1 << "列    ";
                            Sleep(400);
                            continue;
                        }
                        if (X2 != -1 && Y2 != -1) {
                            if (match(row, col,
ball, X1, Y1, X2, Y2) && ((fabs(X1 - X2) == 1 && Y1 ==
Y2) || (fabs(Y1 - Y2) == 1 && X1 == X2))) {
                                cct_gotoxy(0,
length - 3);
                                cout << "交换" <<
char(X1 + 'A') << "行" << Y1 + 1 << "列";
                                cout << " <=> " <<
char(X2 + 'A') << "行" << Y2 + 1 << "列";
                                Sleep(400);
                                hint[X1][Y1] = 0;
                                hint[X2][Y2] = 0;
                                int temp =
ball[X1][Y1];
                                ball[X1][Y1] =
ball[X2][Y2];
                                ball[X2][Y2] =
temp;
                                draw_ball(row,
col, divide, ball, clear, hint);
                                Sleep(200);
                                find_clear(row,
col, ball, clear, &score);
                                draw_ball(row,
col, divide, ball, clear, hint);
                                draw_clear(row,
col, divide, ball, clear);
                                cct_gotoxy(16, 0);
                                cout << "(当前分
数：" << score << " 右键退出)";
                                draw_drop(row,
col, divide, ball, clear);
                                Sleep(200);
                                fill_numbers(row,
col, ball, clear);
                                draw_fill(row,
col, divide, ball, clear);
                                find_clear(row,
col, ball, clear, &score);
                                while
(!is_all_zero(row, col, clear)) {
                                    draw_ball(row, col, divide, ball, clear, hint);
                                    draw_clear(row, col, divide, ball, clear);
                                    cct_gotoxy(16, 0);
                                    cout << "(当前分数：" << score << " 右键退出)";
                                    draw_drop(row, col, divide, ball, clear);
                                    Sleep(200);
                                    fill_numbers(row, col, ball, clear);
                                    draw_fill(row, col, divide, ball, clear);
                                    find_clear(row, col, ball, clear, &score);
                                }
                                find_hint(row,
col, ball, hint);
                                if
(is_all_zero(row, col, hint)) {
                                    cct_gotoxy(0,
0);
                                    cout << "无可
消除项，游戏结束！最终分数：" << score << "      ";
                                    cct_gotoxy(0,
length - 3);
                                    break;
                                }
                                draw_ball(row,
col, divide, ball, clear, hint);
                            }
                            else {
                                cct_showstr(2 + Y1
* (divide + 1) * 2, 2 + X1 * (divide + 1), "◎",
ball[X1][Y1], COLOR_BLACK, 1);
                                cct_setcolor();
                                cct_gotoxy(0,
length - 3);
                                cout << "不能交换"
<< char(X1 + 'A') << "行" << Y1 + 1 << "列";
                                cout << " <=> " <<
char(X2 + 'A') << "行" << Y2 + 1 << "列";
                                Sleep(400);
                                continue;
                            }
                        }
                    }
                }
            }
            cct_gotoxy(0, length - 3);
            cct_setcursor(CURSOR_VISIBLE_NORMAL);
        }
    }
```

6.3 magic_ball_tools

6.3.1 小工具函数

```
bool is_all_zero(int row, int col, int
arr[NUM_MAX][NUM_MAX])
{
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (arr[i][j] == 1) {
```

```
                return false; // 发现1，返回false
            }
        }
    }
    return true; // 遍历完未发现1，返回true
}
void end()
{
     int X, Y;
     cct_getxy(X, Y);
     while (1) {
          char t[4] = { 0 };
          cct_gotoxy(X, Y);
          cout << "本小题结束，请输入End继续:
";
          cct_gotoxy(30, Y);
          cin.getline(t, 4);
          //得到 End
     }
}
```

6.3.2 判断交换匹配函数

```
bool match(int row, int col, int
ball[NUM_MAX][NUM_MAX], int x1, int y1, int x2, int y2)
{
     int temp = ball[x1][y1];
     ball[x1][y1] = ball[x2][y2];
     ball[x2][y2] = temp;
     bool match = 0;
     for (int i = 0; i < row; i++) {
          for (int j = 0; j < col - 2; j++) {
               if (ball[i][j] == ball[i][j + 1] &&
ball[i][j] == ball[i][j + 2]) {
                    match = 1;
               }
          }
     }
     for (int j = 0; j < col; j++) {
          for (int i = 0; i < row - 2; i++) {
               if (ball[i][j] == ball[i + 1][j] &&
ball[i][j] == ball[i + 2][j]) {
                    match = 1;
               }
          }
     }
     ball[x2][y2] = ball[x1][y1];
     ball[x1][y1] = temp;
     return match;
}
```

6.3.3 读取鼠标函数

```
int read_mouse(int row, int col, int(*ball)[NUM_MAX],
int(*clear)[NUM_MAX], int(*hint)[NUM_MAX], const int
length, char select, int* x, int* y)
{
     int X = 0, Y = 0;
     int ret, maction;
     int loop = 1;
     int keycode1, keycode2;
     cct_enable_mouse();
     cct_setcursor(CURSOR_INVISIBLE);     //关闭光标
     char colLabels[] = "ABCDEFGHI";
     while (1) {
          ret = cct_read_keyboard_and_mouse(X, Y,
maction, keycode1, keycode2);
          if (ret == CCT_MOUSE_EVENT) {
               cct_gotoxy(0, length - 3);
               cout << "[当前光标] ";
               bool legal = 0;
               int boardY = (X - 1) / 4;
               int boardX = (Y - 2) / 2;
               if ((X - 1) % 4 == 0 || (X - 1) % 4 ==
3 || (Y - 2) % 2 == 1) { // 边框处
                    legal = 0;
               }
               else if (boardX >= 0 && boardX < row &&
boardY >= 0 && boardY < col) {
                    legal = 1;
               }
               if (legal) {
                    cout << setw(2) <<
colLabels[boardX] << "行" << setw(2) << boardY + 1 << "
列              ";
               }
               else
                    cout << "位置非法";

               if (legal) {
                    //左键选择
                    if (maction ==
MOUSE_LEFT_BUTTON_CLICK) {
                         cct_gotoxy(0, length - 3);

                         if (hint[boardX][boardY] ==
1) {
                              *x = boardX;
                              *y = boardY;

                              cout << "当前选择" <<
colLabels[boardX] << "行" << boardY + 1 << "列       ";
                              Sleep(400);
                              cct_enable_mouse();
                              break;
                         }
                         else {
                              cout << "不能选择" <<
colLabels[boardX] << "行" << boardY + 1 << "列       ";
                              Sleep(200);
                              cct_enable_mouse();
                              continue;
                         }
                    }
                    //右键退出
                    if (maction ==
MOUSE_RIGHT_BUTTON_CLICK)
                         return -1;
               }
          } //end of if (CCT_MOUSE_EVENT)
     } //end of while(1)
     cct_disable_mouse(); //禁用鼠标
     return 0;
}
```

6.4 magic_ball_base

```
void print_stack(int row, int col, int(*ball)[NUM_MAX],
int(*clear)[NUM_MAX], int(*hint)[NUM_MAX])//打印数栈
```

```
{
     cout << endl;
     cout << "  |";
     for (int i = 0; i < col; i++) {
          cout << " " << i + 1 << " ";
     }
     cout << endl;
     cout << "--+-";
     for (int i = 0; i < col; i++) {
          cout << "---";
     }
     cout << endl;
     for (int i = 0; i < row; i++) {
          cout << char('A' + i) << " |";
          for (int j = 0; j < col; j++) {
               cout << " ";
               int X, Y;
               cct_getxy(X, Y);
               //可消除项填色
               if (*(clear[i] + j) == 1) {
                    const int bg_color =
COLOR_HYELLOW;
                    const int fg_color = COLOR_HBLUE;
                    cct_showch(X, Y, '0' + *(ball[i] +
j), bg_color, fg_color, 1);
               }
               else if (*(hint[i] + j) == 1) {
                    const int bg_color = COLOR_HPINK;
                    const int fg_color = COLOR_YELLOW;
                    cct_showch(X, Y, '0' + *(ball[i] +
j), bg_color, fg_color, 1);
               }
               else
                    cout << *(ball[i] + j);
               cct_setcolor();
               cout << " ";
          }
          cout << endl;
     }
     cout << endl;
}
void find_clear(int row, int col, int(*ball)[NUM_MAX],
int(*clear)[NUM_MAX], int *score)
{
     //clear 全部置 0
     //检查行
     for (int i = 0; i < row; i++) {
          for (int j = 0; j < col - 2; ) {
               if (ball[i][j] == ball[i][j + 1] &&
ball[i][j] == ball[i][j + 2]) {
                    int k = j;
                    while (k < col && ball[i][k] ==
ball[i][j]) {
                         clear[i][k] = 1;
                         *score = *score + 1;
                         k++;
                    }
                    j = k;  // 跳过已检查过的部分
               }
               else
                    j++;
          }
     }
     //检查列，同理
}
void drop(int row, int col, int(*ball)[NUM_MAX],
int(*clear)[NUM_MAX])
{
     //ball 全部置 0
     for (int j = 0; j < col; j++) {
          int write_row = row - 1;//写入位置初始为最后
一行
          for (int i = row - 1; i >= 0; i--) {
               if (ball[i][j] != 0) {
                    ball[write_row][j] = ball[i][j];//
值交换
                    clear[write_row][j] = 0;//状态置0
                    if (write_row != i) {
                         ball[i][j] = 0;//发现0，0上移
                         clear[i][j] = 1;//状态置1
                    }
                    write_row--;//写入位置上移
               }
          }
     }
}
void fill_numbers(int row, int col,
int(*ball)[NUM_MAX], int(*clear)[NUM_MAX])
{
     for (int j = 0; j < col; j++) {
          for (int i = 0; i < row; i++) {
               if (ball[i][j] == 0)
                    ball[i][j] = rand() % 9 + 1;
          }
     }
}
void find_hint(int row, int col, int (*ball)[NUM_MAX],
int (*hint)[NUM_MAX])
{
     //hint 全部置 0
     for (int i = 0; i < row; i++) {
          for (int j = 0; j < col; j++) {
               //检查列
               if (j + 1 < col && match(row, col,
ball, i, j, i, j + 1)) {
                    hint[i][j] = 1;
                    hint[i][j + 1] = 1;
               }
               //检查行，同理
          }
     }
}
```

6.5 magic_ball_graph

```
void draw_border(int row, int col, bool divide, int
sleep_time)
{
    cct_gotoxy(0, 1);
     cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
     //绘制边框，中文字符
    cct_setcolor();
}
void draw_ball(int row, int col, bool divide,
int(*ball)[NUM_MAX], int(*clear)[NUM_MAX],
int(*hint)[NUM_MAX], int sleep_time)
```

```
{
    cct_gotoxy(0, 1);
    const int X = 2;
    const int Y = 2;
    const char* str1 = "●";
    const char* str2 = "◎";
    const char* str3 = "○";
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            int ballX = X + j * (divide + 1) * 2;
            int ballY = Y + i * (divide + 1);
            if (*(clear[i] + j) == 1) {
                cct_showstr(ballX, ballY, str1,
*(ball[i] + j), COLOR_BLACK, 1);
            }
            else if (*(hint[i] + j) == 1) {
                cct_showstr(ballX, ballY, str2,
*(ball[i] + j), COLOR_BLACK, 1);
            }
            else {
                cct_showstr(ballX, ballY, str3,
*(ball[i] + j), COLOR_BLACK, 1);
            }
            delay(sleep_time);
        }
    }
    cct_setcolor();
}
void draw_clear(int row, int col, bool divide,
int(*ball)[NUM_MAX], int(*clear)[NUM_MAX])
{
    cct_gotoxy(0, 1);
    const int X = 2;
    const int Y = 2;
    const char* str1 = "○";
    const char* str2 = "□";
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            int ballX = X + j * (divide + 1) * 2;
            int ballY = Y + i * (divide + 1);
            if (*(clear[i] + j) == 1) {
                for (int k = 0; k < 5; k++) {
                    cct_showstr(ballX, ballY, str2,
*(ball[i] + j), COLOR_BLACK, 1);
                    Sleep(20);
                    cct_showstr(ballX, ballY, str1,
*(ball[i] + j), COLOR_BLACK, 1);
                    Sleep(50);
                }
                cct_showstr(ballX, ballY, "  ",
COLOR_HWHITE, COLOR_HWHITE, 1);
                Sleep(20);
            }
        }
    }
    cct_setcolor();
}
void draw_drop(int row, int col, bool divide,
int(*ball)[NUM_MAX], int(*clear)[NUM_MAX])
{
    cct_gotoxy(0, 1);
    const int X = 2;
    const int Y = 2;
    const char* str = "○";
    //ball 全部置 0
    for (int j = 0; j < col; j++) {
        int write_row = row - 1;/
        for (int i = row - 1; i >= 0; i--) {
            int ballX = X + j * (divide + 1) * 2;
            int ballY = Y + i * (divide + 1);
            if (ball[i][j] != 0) {
                ball[write_row][j] = ball[i][j];
                clear[write_row][j] = 0;
                if (write_row != i) {
                    cct_showstr(ballX, ballY, "  ",
COLOR_HWHITE, COLOR_HWHITE, 1);
                    Sleep(10);
                    cct_showstr(ballX, Y + write_row *
(divide + 1), str, *(ball[i] + j), COLOR_BLACK, 1);
                    Sleep(60);
                    ball[i][j] = 0;/
                    clear[i][j] = 1;/
                }
                write_row--;
            }
        }
    }
    cct_setcolor();
}
void draw_fill(int row, int col, bool divide,
int(*ball)[NUM_MAX], int(*clear)[NUM_MAX])
{
    cct_gotoxy(0, 1);
    const int X = 2;
    const int Y = 2;
    const char* str = "○";
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            int ballX = X + j * (divide + 1) * 2;
            int ballY = Y + i * (divide + 1);
            if (*(clear[i] + j) == 1) {
                cct_showstr(ballX, ballY, str,
*(ball[i] + j), COLOR_BLACK, 1);
                Sleep(50);
            }
        }
    }
    cct_setcolor();
}
```