同济大学

实验报告

都来玩彩球游戏

名　　称：彩球游戏实验报告

班　　级：信 11

学　　号：2351892

姓　　名：陈奕炫

完成日期：2024/5/31

1. 题目及基本描述

1.1.菜单基本要求

本题要求复现部分微软彩球游戏“Magic Ball”的内容，分为九个彩单内容（0 菜单为退出）。其中菜单 1-3 项是内部数组部分的完成，共同完成内部普通数组对画框架、画初始彩球、消除彩球，补零，填充新彩球的基础操作，并其中完成对可交换彩球的判断。4-7 项是对可视化图形界面的基础绘制，而第 8 项则算是对 4-7 项的整合，并为第 9 项完整版做基础。而第 9 项是 c m d 界面图形可视化的完整版，不仅包括以上操作的可视化实现，也要完成球的交换、操作提示以及记分系统。

1.2.多源文件以及头文件的要求

题目要求共有六个源文件，除去 c c t 工具函数源文件，包括 main、menu、tools、base、graph 等五个不同的部分。

其中 main 函数主要通过调用 menu 函数打印初始菜单，并通过返回值在 tools 函数中选择调用不同的函数部分。Tools 函数主要包含菜单 1-9 的实现部分，以及每次行列输入和结束后的等待部分，每一部分做成不同的函数封装起来。Base 函数主要是负责一些基础部分，如对内部数组初始化，下落除 0，判断交换等基础内部操作。最后的 graph 函数则是负责最后图形可视化的完成。

最终包含两个头文件：cct.h 和 magic_ball.h。负责不同源文件之间互相调用函数。

1.3.其他细节要求

程序中一些操作要共用函数，实现的不同行为用参数解决。比如在有无分割线框架和彩球的函数，要通过参数的不同来决定输出的不同位置。还有在鼠标控制游戏的函数中，也要通过参数来完成 8/9 的不同要求。

程序不得使用任何全局变量，只能通过函数传参和指针的方法来操作。

2.整体设计思路

本程序的设计首先考虑到了用户操作的便利性和程序结构的清晰性。通过设计菜单部分，程序建立了一个明确的功能选择界面，使得用户能够轻松地进行各项操作。这种模块化的设计使得程序的功能划分更

加清晰，也方便了后续的调试工作。

在完成了菜单部分的设计后，程序着手实现内部数组的初始化功能。该功能确保了程序在启动时能够生成一个符合游戏规则的初始状态，具体地，程序定义了一个 9×9 的字符数组，并使用随机数填充其中的各个位置，符号‘1’至‘9’代表不同颜色的小球，‘0’表示空位。为防止数组越界，程序还做了相应的边界处理，保证了程序的稳定性。

随后，程序逐步实现了各个功能函数，按照菜单要求的顺序进行设计。首先是判断消除位置的彩球、补零和填充新值的功能函数，这些函数是游戏逻辑的核心部分，保证了游戏的正确进行。其次，程序设计了判断交换的函数，确保了交换操作的合法性和正确性。

最后，程序补充了各种细节和可视化部分的函数实现，提升了用户体验。这些细节包括游戏界面的美化、提示信息的显示等，增强了程序的可用性和吸引力。

3.主要功能的实现

3.1.Menu函数和main函数

程序的执行流程始于 main 函数，其中调用了 menu 函数。在调用 menu 函数之前，main 函数会清除屏幕上的内容，以确保用户在一个清晰的界面中进行选择。一旦 menu 函数被调用，程序会跳转到菜单页面，展示给用户不同的选项。用户可以根据需求选择相应的菜单项。

一旦用户做出选择，菜单函数将所选项的值传回到 main 函数。main 函数利用这个值来调用 tools 函数中的不同部分，以执行用户选择的功能。这种设计使得程序结构清晰，模块化程度高，同时也提供了灵活性，使得程序能够根据用户的需求执行不同的功能。

3.2.Base文件中函数

Base 函数承担了内部数组的初始化和基础操作的任务。

首先，它负责初始化内部数组。这一过程使用了取系统时间作为随机数种子的方法，以确保每次运行程序时得到的随机数序列是不同的，进而将数组填充成 1 到 9 的数字字符的随机数组。这个过程确保了游戏的初始状态是随机的。

对于消除部分，Base 函数中实现了一系列功能。首先是定义了一个新的布尔数组 remove_flag，用于标识哪些彩球可以被消除。通过 check_removable_ball 函数判断每个位置的彩球是否满足消除条件，具体的实现是判断每个位置水平和垂直方向上是否有连续三个相同彩球。之后使用 dropAndFillZero 函数实现了彩球下落且填充新值的操作。

此外，Base 函数还实现了判断每个位置的彩球是否可以与相邻位置的彩球进行交换的功能部分。这一功能通过假设交换的方法实现，即假设该球与相邻位置的球交换，重新用 check_removable_ball 函数进行判断。根据判断结果，定义了一个布尔数组 canRemove，用于标识哪些位置的彩球可以进行交换操作。

最后，Base 函数中所有的打印功能都由 printInternalArray 函数以及相应的数组来实现。

3.3.可视化graph文件中函数实现

首先，在程序中存在 draw_frame 和 draw_balls 两个函数，它们的主要责任是负责根据内部数组和消除或可交换数组的参数来进行框架和彩球的绘制。 draw_frame 函数负责绘制游戏框架，而 draw_balls 函数则负责根据内部数组的状态来绘制彩球。这两个函数的作用是在界面上清晰地呈现游戏的当前状态，为玩家提供游戏的可视化界面。

接下来，有 draw_eliminate、draw_drop_balls 和 draw_fill_balls 三个函数，它们负责实现下落除零操作的可视化效果。draw_eliminate 函数用于绘制彩球消除的过程，draw_drop_balls 函数用于绘制彩球下落的过程，而 draw_fill_balls 函数则用于绘制新彩球填充的过程。这些函数的作用是通过视觉效果增强玩家对游戏过程的感知，提高游戏的可玩性和趣味性。

在判断是否可以消除的部分，程序继续使用之前定义的 canRemove 数组进行假设移动法来判断。这样的设计使得程序的逻辑清晰，结构简洁，同时也确保了游戏的可玩性和挑战性。

单独有一个 draw_can_swap 函数用于绘制交换球的过程，这一设计使得交换操作在界面上得到了清晰的呈现，为玩家提供了直观的操作反馈。

最后，在鼠标操作方面，程序使用 mouse_ctrl_game 函数来实现。这个函数负责处理鼠标操作事件，并根据玩家的操作进行游戏状态的更新和界面的刷新，从而实现了游戏的交互性和可操作性。

3.4.其他细节功能函数的实现

在程序中，有许多特殊功能的函数被单独封装，这样的设计使得程序结构更加清晰，同时也提高了代码的可维护性和可读性。下面对这些函数逐一进行说明：

1.`forwait` 函数：该函数用于控制程序在不同位置运行时的暂停与下一步的输入。通过设定适当的延迟时间，可以让程序在运行过程中产生一定的停顿，以便用户更好地观察程序的执行过程。=

2. `InitFlag` 函数：该函数用于对两个布尔型数组进行初始化操作，将它们的所有元素置为 false。这样的设计确保了数组在使用之前的初始状态是明确的，避免了未初始化带来的不确定性，增强了程序的稳定性和可靠性。

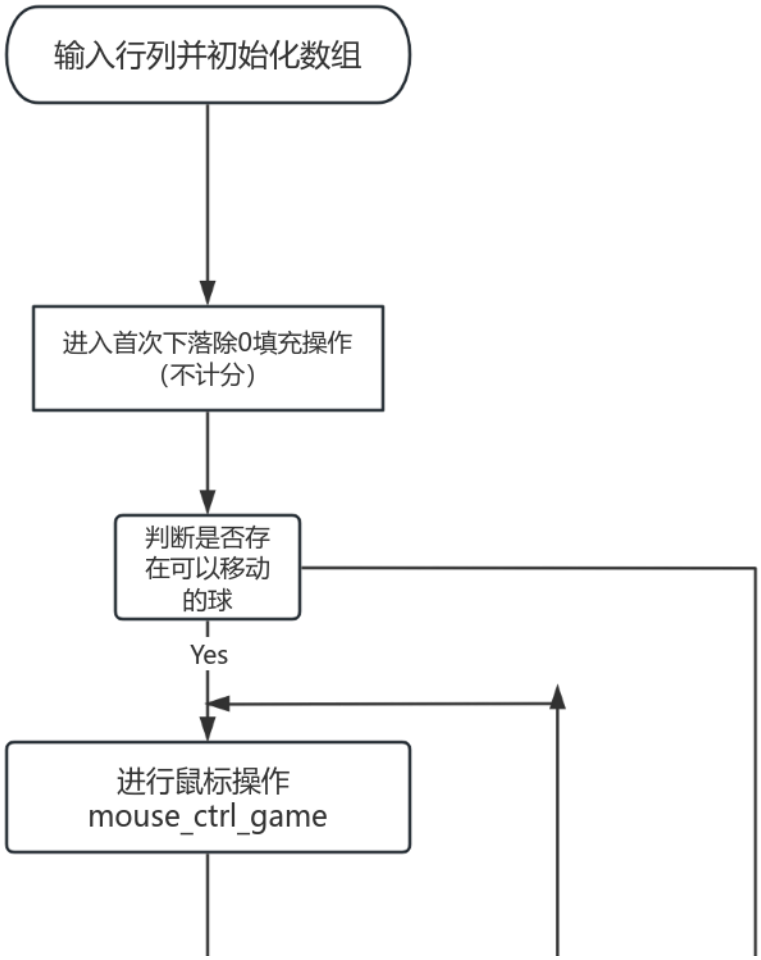
3.`IsActiveArray` 函数：该函数用于判断某个布尔型数组中是否存在 true，如果存在则返回 true，否

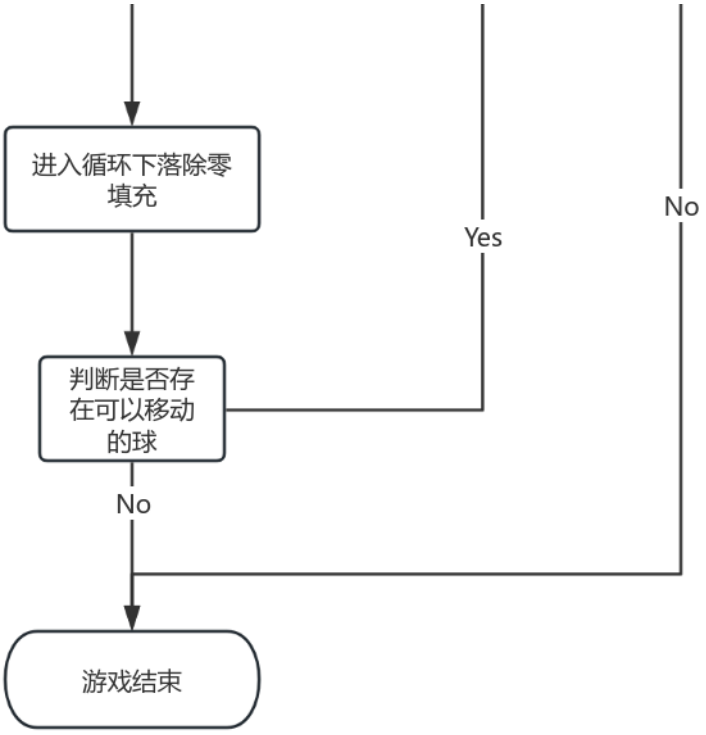
则返回 false。这个函数的作用是用来检查某个数组是否处于活跃状态，从而决定程序的后续操作。这样的设计增加了程序的灵活性和可扩展性。

4.`Get_pos` 函数：该函数用于通过一个球在数组中的位置，返回其在可视化绘制中应有的坐标。这个函数的作用是将程序中的抽象位置信息转换为可视化界面的实际坐标，从而实现了球在界面上的正确显示和位置定位。

5.`BallColor` 函数：该函数用于通过一个球的编号来返回它应有的颜色。这个函数的主要目的是为了避免显示一些不宜显示的球，如白色球等。通过这个函数，程序可以根据球的编号确定其应该显示的颜色。

3.5.cmd 完整版实现流程图





4.调试过程碰到的问题

4.1. cmd完整版中的循环消除问题

在第九项的完成过程中，每个地方的循环消除问题很容易出现 bug。首先在程序开始的时候，首先要进行消除判断（这个过程不能计分），然后才能开始鼠标的点击操作，这时候每次用鼠标交换彩球后都要进行循环消除判断操作，并且在循环的过程中要进行各种 bool 数组的重置。在循环中，使用了 action 这一 int 值来操纵每一部分的运行：值为-1 时退出大循环，为 1 时进行循环消除操作，为 0 时只进行基础循环。

```
while (action>=0) {

    canRemoveAfterSwap(ball_arr, remove_flag, canRemove, row, col);
    if (isActiveArray(canRemove, row, col)) {
        draw_can_swap(ball_arr, canRemove, row, col);
        action = 0;
    }
    else {
        draw_can_swap(ball_arr, canRemove, row, col);
        break;//action = -1;
    }
    action = mouse_ctrl_game(ball_arr, canRemove, row, col, true);
    while (action==1) {
        //先进行消除
        check_removable_ball(ball_arr, remove_flag, row, col);
        score+=draw_eliminate(ball_arr, remove_flag, row, col);
        cct_gotoxy(14, 0);
        cout << "(当前分数：" << score << " 右键退出)                " << endl;
        draw_drop_balls(ball_arr, row, col);
        draw_fill_balls(ball_arr, row, col);
        check_removable_ball(ball_arr, remove_flag, row, col);
        if ((isActiveArray(remove_flag, row, col))) {
            continue;
        }
        else {
            canRemoveAfterSwap(ball_arr, remove_flag, canRemove, row, col);
            if (isActiveArray(canRemove, row, col)) {
                action = 0;
            }
            else {
                action = -1;
            }
            draw_can_swap(ball_arr, canRemove, row, col);
        }
    }
    initFlag(remove_flag, row, col);
    initFlag(canRemove, row, col);

}
```

4.2. 鼠标右键退出在cmd窗口中出现弹窗的问题

在 mouse_ctrl_game 这个用鼠标操纵游戏的函数中由于右键在 cmd 运行时，会出现以下弹窗。

而这与我们用鼠标单击右键退出程序的要求有矛盾，虽然仍然可以正确退出，但在体验感上有所欠缺。

猜测这个地方是因为程序本身和 cmd 同时读入鼠标右键造成的，有了这个猜测，遂进行验证：

```
        break;
    case MOUSE_RIGHT_BUTTON_CLICK:          //按下右键
        loop = 0;
        action = -1;
        Sleep(100);
        break;
```

像这样，当点击右键后，延时一会，就可以解决同时读入的问题，此时单击右键后在 cmd 中的弹窗也不会再出现。

5.心得体会

通过这次作业，我收获了以下心得体会。

模块化设计提升效率与可维护性：将程序划分为 main、menu、tools、base、graph 等多个模块，每个模块专注于特定任务，不仅让代码结构清晰明了，还极大提高了开发效率。在后期调试和维护时，这种模块化设计使我可以迅速定位问题所在，针对性地进行修改，无需在庞大的代码库中迷失方向。

函数封装增强代码复用性：通过精心设计的函数封装，如`check_removable_ball`、`dropAndFillZero`等，我学会了如何将复杂的逻辑封装在独立的函数中，使其具有高度的通用性。这不仅减少了代码冗余，还使得相同的逻辑可以在不同的场景下重复利用，如判断消除和交换逻辑的复用，大大提高了代码的灵活性和可扩展性。

参数化设计提升函数灵活性：在实现过程中，我深刻理解到通过参数化设计来控制函数行为的灵活性。例如，`draw_balls`和`draw_frame`函数通过接受不同的参数来决定是否绘制分割线或不同状态下的彩球，这种做法不仅简化了代码，还使得函数能够在多种场景下适应不同的需求。

全局变量禁用与数据传递：遵循不使用全局变量的原则，全靠函数参数和返回值传递数据，虽然在初期增加了编码的复杂度，但长期来看，这样的设计显著降低了代码间的耦合度，减少了潜在的错误来源，提高了程序的稳定性和可测试性。

明确函数职责：每个函数应有明确单一的职责，这有助于减少代码间的相互依赖，便于理解和调试。例如，`Base`文件中的函数专注于内部数组的操作，而`graph`文件的函数则专注于可视化处理，这样的分工让代码逻辑更为清晰。

参数化增强通用性：利用参数来控制函数的行为，使同一函数能在不同情境下发挥不同作用。比如`draw_can_swap`函数通过接收交换球的信息，灵活地在界面上展示交换过程，这种设计模式让函数变得更加通用和强大。

设计模式的应用：在处理类似消除判断和交换判断等复杂逻辑时，采用策略模式或状态模式等设计模

式，将逻辑封装在独立的类或函数中，使得逻辑变更时只需替换策略或状态对象，而不必改动调用处的代码，提高了系统的可维护性和扩展性。

综上所述，通过这次实验，我不仅掌握了如何高效地设计和实现一个模块化的程序，还深刻理解了函数封装和复用在实际编程中的重要性。这些心得将对我未来的学习和开发工作产生深远的影响。

6.附件：源程序

```
void menu_9()
{
int row, col;
input_row_col(&row, &col);
char ball_arr[9][9] = { 0 };
bool remove_flag[9][9] = { 0 };
bool canRemove[9][9] = { 0 };
int score = 0;
int action=0;//0 为默认，1 为进行循环消除，-1 为退出
initArray(ball_arr, row, col);

cct_cls();
cct_setfontsize("新宋体", 34);
cct_setconsoleborder(60, 25);
cout << "屏幕：" << row + 6 << "行 40 列";
cout << "(当前分数：" << score << "  右键退出)                  " << endl;

draw_frame(row, col, true);
check_removable_ball(ball_arr, remove_flag, row, col);
draw_balls(ball_arr, remove_flag, row, col, true, true);

while (true) {
    check_removable_ball(ball_arr, remove_flag, row, col);
    if (isActiveArray(remove_flag, row, col)) {
        draw_eliminate(ball_arr, remove_flag, row, col);
        Sleep(50);
        draw_drop_balls(ball_arr, row, col);
        Sleep(50);
        draw_fill_balls(ball_arr, row, col);
    }
    else {
        break;
    }
}

while (action>=0) {

    canRemoveAfterSwap(ball_arr, remove_flag, canRemove, row, col);
    if (isActiveArray(canRemove, row, col)) {
```

```
            draw_can_swap(ball_arr, canRemove, row, col);
            action = 0;
        }
        else {
            draw_can_swap(ball_arr, canRemove, row, col);
            break;//action = -1;
        }
        action = mouse_ctrl_game(ball_arr, canRemove, row, col, true);
        while (action==1) {

            //先进行消除

    check_removable_ball(ball_arr, remove_flag, row, col);

    score+=draw_eliminate(ball_arr, remove_flag, row, col);
            cct_gotoxy(14, 0);
            cout << "(当前分数：" << score << " 右键退出)                " << endl;
            draw_drop_balls(ball_arr, row, col);
            draw_fill_balls(ball_arr, row, col);

    check_removable_ball(ball_arr, remove_flag, row, col);
            if ((isActiveArray(remove_flag, row, col))) {
                continue;
            }
            else {

    canRemoveAfterSwap(ball_arr, remove_flag, canRemove, row, col);
                if (isActiveArray(canRemove, row, col)) {
                    action = 0;
                }
                else {

                    action = -1;

                }

    draw_can_swap(ball_arr, canRemove, row, col);
            }
        }
        initFlag(remove_flag, row, col);
        initFlag(canRemove, row, col);

    }
    cct_gotoxy(14, 0);
    cout << "(无可消除项，游戏结束！最终得分为：" << score << ")     ";
    cct_gotoxy(0, 3 + 2 * row);
```

```
    //对数组进行初始化
    void initArray(char
ball_arr[][9],int row,int col)
    {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j< col; j++) {
            ball_arr[i][j] = rand() % 9
+ 1+'0';
        }
    }
}
    void draw_frame(int row, int col,
bool hasSplitLine)
    {

    cct_setcolor(COLOR_WHITE,
COLOR_BLACK);
    if (hasSplitLine) {
        cout<< "╔";
        for (int i = 0; i < col; i++) {
            cout << "═";
            Sleep(10);
            if (i != col - 1) {
                cout << "╤";
                Sleep(10);
            }

        }
        cout << "╗" << endl;;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col;
j++) {
                cout << "│  ";
                Sleep(10);
            }
            cout << "│" << endl;
            if (i != row - 1) {
            cout << "╟";
            cout << "─";
            Sleep(10);
                for (int k = 0; k <
col - 1; k++) {
                    cout << "┼";
                    cout << "─";
                    Sleep(10);
                }
                cout << "╢" << endl;
            }

        }
        cout << "╚";
        for (int i = 0; i < col; i++) {
            cout << "═";
            Sleep(10);
            if (i != col - 1) {
                cout << "╧";
                Sleep(10);
            }
        }
        cout << "╝" << endl;;
    }
    else {
        cout << "╔";
        for (int i = 0; i < col; i++) {
```

```
            cout << "═";
            Sleep(10);
        }
        cout << "┐";
        for (int i = 0; i < row; i++) {
            cct_gotoxy(0, i + 2);
            cout << "│";
            cct_gotoxy(2 * (col + 1), i + 2);
            cout << "│";
            Sleep(10);
        }
        cout << endl;
        cout << "└";
        for (int i = 0; i < col; i++) {
            cout << "═";
            Sleep(10);
        }
        cout << "┘";
    }
    cct_setcolor();

}
    void draw_balls(char ball_arr[][9], bool remove_flag[][9],int row, int col, bool hasSplitLine, bool isShowisActiveArray)
    {
    if (hasSplitLine==false) {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col;
    {
                int color = ballColor(ball_arr[i][j]);
                if (isShowisActiveArray && remove_flag[i][j]) {
                    cct_showstr(2 * (1 + j), 2 + i, "●", color, COLOR_BLACK, 1);
                }
                else {
                    cct_showstr(2 * (1 + j), 2 + i, "○", color, COLOR_BLACK, 1);
                }
                Sleep(10);
            }
        }
        cct_gotoxy(0, 3 + row);
    }
    else {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                int color = ballColor(ball_arr[i][j]);

                if (isShowisActiveArray && remove_flag[i][j]) {
                    cct_showstr(2 + 4 * j, 2 + 2 * i, "●", color,
```

```
                        }
                    Sleep(10);
                }
            }
            cct_gotoxy(0, 3 + 2*row);
        }
        cct_setcolor();

}
    int mouse_ctrl_game(char
ball_arr[][9],bool canRemove[][9],int
row,int col,bool game)
    {
    int X = 0, Y = 0;
    int ret, maction;
    int keycode1, keycode2;
    bool loop = true;
    bool hasBeClick[9][9] = { 0 };
    int action = 0;
    char* p1 = 0;
    char* p2 = 0;
    cct_enable_mouse();

    cct_setcursor(CURSOR_INVISIBLE);//光
标不显示
    while (loop) {
        ret =
cct_read_keyboard_and_mouse(X, Y,
maction, keycode1, keycode2);
        if (ret == CCT_MOUSE_EVENT) {

            cct_gotoxy(0, 3 + 2 * row);
            int x = X / 4;
            int y = Y / 2 - 1;

            if ((((X - 2) % 4 == 0) ||
(X - 3) % 4 == 0)
                && (Y > 0) && (Y % 2
== 0)

    &&x>=0&&x<col&&y>=0&&y<row)
            {
                cout << "[当前光标]  "
<< char(y+'A') << "行" << x+1<<"列    ";
                cct_gotoxy(0, 3 + 2 *
row);
                switch (maction) {
                    case
MOUSE_LEFT_BUTTON_CLICK:            //按
下左键
                        if
(canRemove[y][x]&&isActiveArray(hasBeCli
ck,row,col)==false ) {

    cct_gotoxy(0, 3 + 2 * row);
                            cout
<< "当前选择" << char(y + 'A') << "行"
<< x + 1 << "列                ";

    Sleep(400);
                            if
(!game) {

    loop = false;
```

```
                                    }
                                    else
{

    hasBeClick[y][x] = true;

    p1 = &ball_arr[y][x];
                                    }

    draw_click(y,
x,ballColor(ball_arr[y][x]), "选取");
                                }
                                else if
(canRemove[y][x] &&
isActiveArray(hasBeClick, row,
col)&&hasBeClick[y][x]==0) {

    hasBeClick[y][x] = true;
                                    p2 =
&ball_arr[y][x];
                                    bool
isReSelect = false;

    isReSelect = draw_swap(ball_arr,
hasBeClick, row, col);
                                    if
(isReSelect) {

    //draw_click(y, x,
ballColor(ball_arr[y][x]), "选取");
```

```
    initFlag(hasBeClick, row, col);

    //hasBeClick[y][x]=true;
                                }
                                else
{

    swapBall(p1, p2);
    action = 1;
    initFlag(hasBeClick, row, col)
    loop = false;
                                }

                            }
                            else if
(canRemove[y][x] &&
isActiveArray(hasBeClick, row, col) &&
hasBeClick[y][x]) {

    draw_click(y, x,
ballColor(ball_arr[y][x]), "取消选取");

    hasBeClick[y][x] = false;
                            }
                            else {

    cct_gotoxy(0, 3 + 2 * row);
                                cout
<< "不能选择" << char(y + 'A') << "行"
<< x + 1 << "列                ";
```

```
        Sleep(500);
                                }



                            break;
                        case
MOUSE_RIGHT_BUTTON_CLICK:           //按
下右键
                            loop = 0;
                            action = -1;
                            Sleep(100);
                            break;
                        case
MOUSE_NO_ACTION:
                        default:
                            break;
                    }
                }
                else {
                    cout << "[当前光标]  "
<< "位置非法";
                }
            }
    }
    cct_gotoxy(0, 3 + 2 * row);
    cct_disable_mouse();  //禁用鼠标
    cct_setcursor(CURSOR_VISIBLE_NORMAL)
;   //打开光标
    return action;
    }
```