装 订 线

彩球消消乐游戏实现报告

班级：计算机一班
姓名：王哲源
学号：1652228

完成日期：2017.3.13

1.题目及其描述

题目主要目的在于通过 CMD 实现彩球消消乐的伪图形化，其中有以下分问题

1.1 利用内部数组生成初始状态，并寻找可消除项

其为本游戏实现的最基本步骤，即随机生成初始状况

1.2 在 1.1 基础上使可下落小球进行下落并填充空位

其为本游戏实现的另一基本步骤，即在出现空位时使小球受重力垂直下落填补空缺并填入新球

1.3 在 1.2 基础上寻找可交换消除项

其为游戏的提示机制，同时可为后面的消除判定提供帮助

1.4 在 1.1 的基础上绘制无分割线网格图

其为实现游戏伪图形化的第一步，即实现最简单的色块打印

1.5 在 1.4 的基础上绘制有分割线网格图

该小题为实现伪图形化界面的进一步改进

1.6 结合实现 1.1 与 1.5

其为实现伪图形化的基础，即支持伪图形化可消除项的显示

1.7 结合实现 1.3 与 1.5

经过本题的实现基本完成游戏版的一个操作周期内容

1.8 游戏实现最终版

通过加入支持鼠标操作，在 cmd 界面下实现最终的游戏版

*1.9 通过读入文件检查判定消除程序正确性

此步仅为检验判定消除程序是否正确的补充小题

1.10 退出

2.整体设计思路

以下对于程序的一些设计思路进行概述，由于 1.9 为过程检查性小题，此处不予以涉及

2.1 关于界面的存储

由于游戏操作界面为 9×9 的二维矩阵，因此此处利用一个 n*m(n,m<=15)的二维数组 map[i][j]表示第 i 行第 j 列位置的球的颜色。球颜色由 1-9 进行表示，空白位置则由 0 作为缺省值。

2.2 关于消除判断

鉴于至少满足三球同色相连时即可判定消除，因此决定一次性判断三个连续的球进行颜色判断，若均相同再判断是否存在更多的可同时消除的球

2.3 关于彩球的下落

鉴于彩球只会自上而下单向下落（而不会出现类似水流左右均分的情况），因此此处只需要对每一列进行一次检查并使其下落即可。

2.4 关于可交换项的提示查询

仅考虑最简单的消除情况，即满足三个连续同色可消除，那么能发生消除的交换有且仅有两个，一个是已存在两个连续的同色球，第三个交换后可补在头/尾处发生消除；另一种情况是头尾满足同色，而两者中间的球经过交换达成同色发生交换，因此可以根据这个条件分两种情况进行判定。

2.5 关于交换后可消除小球的处理

此处注意到，无论是经过交换所可消除的小球，亦或是因随机生成/下落所产生的可消除小球，均满足相同的消除规律，因此可以在执行交换操作之后将本次循环结束，直接返回循环开头对同色彩球的消除，使交换所产生的可消除小球与随机产生的小球共同消除。

3.主要功能实现

以下对整个程序的具体实现进行较为详细的介绍，其中主要以 1.8 游戏版的设计思路为主

3.1 预处理部分

在读入长(n)宽(m)之后，首先对于图进行重置，利用随机函数确定每个格子小球的颜色，同时对矩阵的边框部分赋值-1，以防之后的操作因不注意越界发生非法访问的情况。

3.2 主体函数部分

主题函数通过永真循环作为外部循环，下含消除、下落、填补、查询、交换五个子部分，其中消除、下落、填补三个步骤将进行多次执行直到新的格子内不再存在满足消除条件的彩球。以下进行详细描述。

3.2.1 消除

采用 2.4 中所描述的方法，每当查询到一个新的小球时，对该小球向右/向下的连续三个球颜色查询（向上与向左与此为相同操作因此没有必要进行），若满足同色条件，再进行更多的查询与标记。

3.2.2 下落

对 3.2.1 中标记的球进行消除之后使得 map 上出现多个 0 的位置。此时从最后一行开始向上逐行查询，当碰到空位时，以此为基准向上单向查询，若发现非 0 的彩球，将其放置到之前的空位，并将原位置为 0。之后将空位的指针上移，由被下落的位置继续向上查询直至碰到边界为止。通过该办法处理整张棋盘实现彩球的下落

3.2.3 填充

此处只需将经过 3.2.1 的图进行自上而下的查询，遇到空位后填上新的随机彩球即可。

3.2.4 查询

注意到，此时图中不存在三个及以上的连续彩球，因此我们只需要根据 2.4 中所描述的两种情况，对于横向及纵向分别进行查询即可。当发现可交换的彩球时，将其进行标记即可。

3.2.5 交换

由于在 3.2.4 中我们已经将可交换的彩球位置全部标记出来了，因此在此步中我们可以直接判断鼠标点击的位置是否为这些被标记彩球。若不是则直接拒绝执行即可。但我们应当注意到一件事，即由于我们在 3.2.4 中的标记仅仅是当满足交换就将可交换彩球标记，但到此处时无法判断该交换是否合法——即我们无法确定此时被我们点击的两个彩球是否是 3.2.4 中被我们标记的那一对彩球（同时应当注意到同一个彩球可能同时与其四周的多个彩球满足交换），因此我们在交换前仍需再次检查我们的交换是否合法，即交换后的彩球是否满足消除条件。但此处的判断相较 3.2.1 中的全图扫更为简单，我们只需要假定两彩球发生交换，对交换后的两球六个方向进行搜寻即可，若发现并非合法交换，直接将交换前备份好的两彩球位置复原，并返回禁止交换的信息即可。

3.3 cmd 伪图形界面的输出

由于彩球在发生消除、选中、提示时需要由不同的图形表示，因此我们在主函数部分使用一个 bool 类型的 flag 数组每次对其进行标记，保存消除/提示等信息，在每一步执行时根据情况将数组清空或直接传入各小步的函数之中进行标记或为处理提供便利。同时注意到这里各步均只

是单纯对某些彩球的图形进行改变，因此除选中彩球的图形修改放在鼠标操作函数中，其余图形修改可以共用一个输出函数，根据传入的参数不同来决定输出的图形与底色为什么，并根据 flag 数组的内容直接对相应位置的彩球进行修改即可，无需对整个界面进行重新打印。

装

订

线

4. 调试过程遇到的问题

以下将对调试过程中遇到的几个主要问题进行概述

4.1 提示交换显示不完全

该问题是在一开始数组界面测试的时候发现的。主要表现为当出现两个相同颜色连续的球时，若右侧存在可被交换的球则可正常显示，而左侧则不会显示。经过检查函数发现是自己最初时情况没被考虑完整。在做程序整体思路初步设计时仅考虑到对于同色球仅需要向右查找是否有同色球可以减少向左查询的时间，但并未考虑当出现同色球时左右两侧均可能出现第三个可以被同时消去的球。经过修改程序可以同时查询左右侧的情况，但如果出现两个连续彩球与一个同色彩球间夹着不同色彩球又无法被判断——即又有一种情况被忽略了。实际上可以被交换的彩球单侧一共有三个方向，而非只有两个（这与 2.4 中提及的另一种情况不同但很容易理所当然认为是相类似的）。

*4.2 使用文件读入时遇到键入回车符提示报错

在实现 1.9 时，采用了 freopen 对输入重定向至文件。但当执行到程序中间要求键入回车时程序会发生弹窗报错，经过检查得知当使用 freopen 对输入重定向时 cmd 窗口不再支持键入（此项同时会对 system(“pause”)造成影响），应当再加入 freopen("CON", "r", stdin);将输入重新定向回控制台窗口才能支持继续从 cmd 读入，否则则会出现上述所说的弹窗报错。

5. 总结部分

5.1 关于文件读入的收获

由于题 1.9 要求，程序需要从文件读入数据，且文件名并非事先定好的。通过查找 freopen 函数的参数得知，其原型为 FILE *freopen(const char *path, const char *mode, FILE *stream)，而其中 const char *path 即为以字符数组所构成的文件路径——其中.\表示与源程序在同一目录，而..\则表示的是上一级目录，因此可以通过构造一个字符数组 str 来指定文件路径。同时，freopen 返回值为 NULL 时则表示文件打开失败，因此可以根据此进行文件读取提示。

5.2 对程序最初构思的反思

在 4.1 中已提及，由于自身在程序编写初期思考的过于大意，导致在编写时出现了一些情况的遗漏。所幸这次问题出现的时间为编写前期，并未对后期图形化界面编写造成过大影响。这一方面暴露了自身对于问题思考的不完整性，也再一次的证明了程序分步编写的重要性。在编写较为长的代码时不能一股脑写到头，而应该将大问题拆分为无数个小问题，通过逐步编写的方式最

后将所有子步骤拼接为最终的程序。这样一来简化了问题的思考和编写难度，另一方面也能及时发现自己某一部分的思维漏洞，避免了由于初期考虑不够周全而导致之后的整个程序不可避免的大型改动。

5.3 对图形界面输出的反思

这次程序的很大一部分来源于前次小球的程序，包括移动、下落以及鼠标操作的部分过程。但小球形状的修改过程几乎完全是重写的。虽然前次小球程序中也涉及鼠标选中及移动操作，但之前对于界面的操作基本是执行一步之后对整张图清空重新打印的，而本次由于涉及小球的不停移动该办法不再适用（变化过于频繁界面会闪烁的非常严重）。而后重新编写根据小球固定位置点对点修改的程序时发现实际复杂度并非想象的那么难，且界面不再像小球那样每次操作闪烁让人难受。因此对于图形界面的修改除非必要情况，实际上对点修改操作是优于全图重打的。这样美观程度提升了，对于程序的执行效率也是有提高的。在将来的编写中该问题应该特别予以注意，不能图方便而放弃了界面美观度而采用最为无脑的方式。

6. 附件：程序（此处以 1.8 为主）

游戏主体函数部分：

```
void work_game(int(*map)[Maxm], const int n, const int m)
{
    bool flag[Maxn][Maxm];
    int tot = 0;
    reset_map(map, n, m);
    output_cmd_reset(map, n, m, 8);

    bool lock = 1;
    setcolor(hout, 0, 7);
    while (1)
    {
        while (1)
        {
            setcolor(hout, 0, 7);
            gotoxy(hout, 0, 13 + 2 * (n - 5));
            printf("
");
            gotoxy(hout, 0, 13 + 2 * (n - 5));
            printf("正在寻找可消除项...");
            if (!check(map, n, m, flag))
            {
                gotoxy(hout, 0, 13 + 2 * (n - 5));
                printf("
");
                gotoxy(hout, 0, 13 + 2 * (n - 5));
                printf("无可消除项\n");
                break;
            }
            output_cmd_change(1, map, n, m, flag);

            if (!lock)
                output_score(tot, n, m, flag);

            gotoxy(hout, 0, 13 + 2 * (n - 5));
            printf("
");
            gotoxy(hout, 0, 13 + 2 * (n - 5));
            printf("\n");
            Sleep(200);
            output_cmd_change(0, map, n, m, flag);
            gotoxy(hout, 0, 13 + 2 * (n - 5));
            printf("
");
            gotoxy(hout, 0, 13 + 2 * (n - 5));

            printf("正在进行下落操作...");
            output_cmd_alter(map, n, m, flag);
```

```
                    fill_map(map, n, m, flag);
                    gotoxy(hout, 0, 13 + 2 * (n
- 5));
                    printf("
");
                    gotoxy(hout, 0, 13 + 2 * (n
- 5));
                    printf("正在进行填充操
作...");
                    output_cmd_change(2,  map,
n, m, flag);
               }

               Sleep(1000);
               gotoxy(hout, 0, 13 + 2 * (n - 5));
               printf("
");
               gotoxy(hout, 0, 13 + 2 * (n - 5));
               printf("正在查询可行交换...");
               if (!find(map, n, m, flag))
               {
                    gotoxy(hout, 0, 13 + 2 * (n
- 5));
                    printf("
");
                    gotoxy(hout, 0, 13 + 2 * (n
- 5));
                    printf("无可行交换");
                    Sleep(500);
                    break;
               }
               output_cmd_change(3, map, n, m,
flag);
               gotoxy(hout, 0, 13 + 2 * (n - 5));
               printf("
");
               gotoxy(hout, 0, 13 + 2 * (n - 5));
               if (lock)
                    lock = 0;

               if   (!mouse_work(map,   n,   m,
flag))
                    break;
          }

          gotoxy(hout, 0, 13 + 2 * (n - 5));
          printf("
");
          gotoxy(hout, 0, 13 + 2 * (n - 5));
          printf("游戏结束!!!!\n");
     }
     打印部分：
     void     output_cmd_change(int     step,
int(*map)[Maxm], const int n, const int m,
bool(*flag)[Maxm])
     {
          char fig[3];
          if (!step)
               strcpy(fig, "  ");
          else if (step == 1)
               strcpy(fig, "★");
          else if (step == 2 || step == 4)
               strcpy(fig, "○");
          else //step==3
               strcpy(fig, "◎");
          For(i, 1, n) For(j, 1, m)
          {
               if (!flag[i][j])
                    continue;
               gotoxy(hout, 4 * j - 2, 2 * i);
               setcolor(hout,      step      ?
col[map[i][j]] : 15, step ? 0 : 15);
               printf("%s", fig);
               Sleep(step == 4 ? 0 : 200);
          }
          setcolor(hout, 0, 7);
     }
     void   output_cmd_reset(int(*map)[Maxm],
const int n, const int m, const int mode)
     {
          const int cols = 35 + 2 * (m - 2);
          const int lines = 20 + 2 * (n - 5);
          const int size = 36 - 4 * (n - 5);
          setconsoleborder(hout, cols, lines);
          setfontsize(hout, L"新宋体", size);

          setcolor(hout, 0, 7);
          gotoxy(hout, 0, 0);
          if (mode == 8)
               printf("(右键-退出)当前总分:
0\n");
          else
               printf("\n");

          setcolor(hout, 15, 0);
          printf("┏");
          For(i, 1, m - 1)
               printf("━┳");
          printf("━┓\n");

          For(i, 1, n - 1)
          {
               printf("┃");
               For(j, 1, m - 1)
               {
                    if (!map[i][j])
                    {
                         printf("  ┃");
                         continue;
                    }
                    setcolor(hout,
col[map[i][j]], 0);
```

```
                printf("○");
                setcolor(hout, 15, 0);
                printf("┃");
            }
            if (!map[i][m])
                printf("  ");
            else
            {
                setcolor(hout,
col[map[i][m]], 0);
                printf("○");
                setcolor(hout, 15, 0);
            }
            printf("┃\n");

            printf("┃");
            For(j, 1, m - 1)
                printf("━╋");
            printf("━┫\n");
        }
        printf("┃");
        For(j, 1, m - 1)
        {
            if (!map[n][j])
            {
                printf("  ┃");
                continue;
            }
            setcolor(hout,  col[map[n][j]],
0);
            printf("○");
            setcolor(hout, 15, 0);
            printf("┃");
        }
        if (!map[n][m])
            printf("  ");
        else
        {
            setcolor(hout,  col[map[n][m]],
0);
            printf("○");
            setcolor(hout, 15, 0);
        }
        printf("┃\n");

        printf("┗");
        For(i, 1, m - 1)
            printf("━┻");
        printf("━┛\n");
    }
    void   output_cmd_alter(int(*map)[Maxm],
const int n, const int m, bool(*flag)[Maxm])
    {
        opFor(i, n, 1) For(j, 1, m)
        {
            if (!flag[i][j])
                continue;
            int prei = i, curi = i - 1;
            while (curi)
            {
                while  (flag[curi][j]  &&
curi)
                    --curi;
                if (!curi)
                    break;
                int tc = col[map[curi][j]];
                map[prei][j]            =
map[curi][j];
                flag[prei][j] = 0;
                map[curi][j] = 0;
                flag[curi][j] = 1;
                For(k, curi + 1, prei)
                {
                    gotoxy(hout, 4 * j - 2,
2 * (k - 1));
                    setcolor(hout, 15, 0);
                    printf("  ");
                    gotoxy(hout, 4 * j - 2,
2 * (k - 1) + 1);
                    setcolor(hout, tc, 0);
                    printf("○");
                    Sleep(50);
                    gotoxy(hout, 4 * j - 2,
2 * (k - 1) + 1);
                    setcolor(hout, 15, 0);
                    printf("━");
                    gotoxy(hout, 4 * j - 2,
2 * k);
                    setcolor(hout, tc, 0);
                    printf("○");
                    Sleep(50);
                }
                --prei;
                --curi;
            }
        }
        setcolor(hout, 0, 7);
    }
    void output_score(int &tot, const int n,
const int m, bool(*flag)[Maxm])
    {
        For(i, 1, n) For(j, 1, m)
            if (flag[i][j])
                ++tot;
        setcolor(hout, 0, 7);
        gotoxy(hout, 21, 0);
        printf("%d", tot);
    }
```

鼠标操作部分：

```
    bool      mouse_init(Point      &mouse,
int(*map)[Maxm], const int n, const int m,
bool(*flag)[Maxm])
```

```
    {
        Point pos;
        pos.x = pos.y = 0;
        enable_mouse(hin);
        setcursor(hout, CURSOR_INVISIBLE);
        while (1)
        {
            int       mouse_action       =
read_mouse(hin, pos.x, pos.y);
            gotoxy(hout, 0, 16 + 2 * (n - 7));
            setcolor(hout, 0, 7);
            if ((pos.x >= 2 && pos.x <= 4 *
m - 1 && ((pos.x % 4 == 2) || (pos.x % 4) == 3))
                && (pos.y >= 2 && pos.y <=
2 * n && !(pos.y % 2)))
            {
                printf("[当前光标位置]：%c
行%d 列", pos.y / 2 + 'A' - 1, (pos.x + 2) / 4);
                if    (mouse_action    ==
MOUSE_RIGHT_BUTTON_CLICK)
                    return 0;
                if    (mouse_action    ==
MOUSE_LEFT_BUTTON_CLICK)
                {
                    mouse  =  {  pos.y  /
2,(pos.x + 2) / 4 };
                    return 1;
                }
            }
        }
    }
    bool  mouse_work(int(*map)[Maxm],  const
int n, const int m, bool(*flag)[Maxm])
    {
        bool select = 0;
        Point mouse, pre;
        while (1)
        {
           if (!mouse_init(mouse, map, n, m,
flag))
                return 0;
            if     (!map[mouse.x][mouse.y]
&& !select)
                continue;
            if (!flag[mouse.x][mouse.y])
            {
                gotoxy(hout, 0, 13 + 2 * (n
- 5));
                printf("
");
                gotoxy(hout, 0, 13 + 2 * (n
- 5));
                printf("不能选择这个位置
");
                continue;
            }
            if (!select)
            {
                select = 1;
                gotoxy(hout, 4 * mouse.y -
2, 2 * mouse.x);
                setcolor(hout,
col[map[mouse.x][mouse.y]], 0);
                printf("□");
                pre = mouse;
                setcolor(hout, 0, 7);
                continue;
            }
            //select == 1
            if   ((mouse.x   ==   pre.x   &&
abs(mouse.y - pre.y) == 1) ||
                (mouse.y   ==   pre.y   &&
abs(mouse.x - pre.x) == 1))
            {
                if (check_swap(map, n, m,
pre, mouse))
                {
                    if (mouse.x < pre.x ||
mouse.y < pre.y)
                        Swap(mouse, pre);
                    int deltax = mouse.x -
pre.x, deltay = 2 * (mouse.y - pre.y);
                    output_cmd_change(4,
map, n, m, flag);

                    setcolor(hout, 15, 0);
                    gotoxy(hout, 4 * pre.y
- 2, 2 * pre.x);
                    printf("  ");
                    gotoxy(hout,    4    *
mouse.y - 2, 2 * mouse.x);
                    printf("  ");
                    setcolor(hout, 15, 0);
                    gotoxy(hout, 4 * pre.y
- 2 + deltay, 2 * pre.x + deltax);
                    printf("○");
                    Sleep(200);

                    setcolor(hout,
col[map[pre.x][pre.y]], 0);
                    gotoxy(hout, 4 * pre.y
- 2, 2 * pre.x);
                    printf("○");
                    setcolor(hout,
col[map[mouse.x][mouse.y]], 0);
                    gotoxy(hout,    4    *
mouse.y - 2, 2 * mouse.x);
                    printf("○");
                    setcolor(hout, 15, 0);
                    gotoxy(hout, 4 * pre.y
- 2 + deltay, 2 * pre.x + deltax);
                    printf(deltax ? "—" :
```

```
"|");
                    Sleep(50);

                    break;
                }
                else
                {
                    gotoxy(hout, 0, 13 + 2
* (n - 5));
                    printf("
");
                    gotoxy(hout, 0, 13 + 2
* (n - 5));
                    printf("无法进行交换
");
                    continue;
                }
            }
            else
            {
                setcolor(hout,
col[map[pre.x][pre.y]], 0);
                gotoxy(hout, 4 * pre.y - 2,
2 * pre.x);
                printf("◎");
                setcolor(hout,
col[map[mouse.x][mouse.y]], 0);
                gotoxy(hout, 4 * mouse.y -
2, 2 * mouse.x);
                printf("□");
                pre = mouse;
                setcolor(hout, 0, 7);
                continue;
            }
        }
        return 1;
    }
    template <typename T>
    void Swap(T &x, T &y)
    {
        T tmp = x;
        x = y;
        y = tmp;
    }
    bool try_swap(int(*map)[Maxm], int n, int
m, int tc, const int X, const int Y)
    {
        int cnt = 0;

        int curX = X - 1;
        while (curX && cnt<2)
        {
            if (col[map[curX][Y]] != tc)
                break;
            cnt++;
            --curX;
        }
        curX = X + 1;
        while (curX <= n && cnt < 2)
        {
            if (col[map[curX][Y]] != tc)
                break;
            cnt++;
            ++curX;
        }
        if (cnt >= 2)
            return 1;

        cnt = 0;
        int curY = Y - 1;
        while (curY && cnt < 2)
        {
            if (col[map[X][curY]] != tc)
                break;
            cnt++;
            --curY;
        }
        curY = Y + 1;
        while (curY <= m && cnt < 2)
        {
            if (col[map[X][curY]] != tc)
                break;
            cnt++;
            ++curY;
        }
        if (cnt >= 2)
            return 1;

        return 0;
    }
    bool check_swap(int(*map)[Maxm], int n,
int m, Point A, Point B)
    {
        int cA = col[map[A.x][A.y]], cB =
col[map[B.x][B.y]];
        Swap(map[A.x][A.y], map[B.x][B.y]);
        if (!try_swap(map, n, m, cA, B.x, B.y)
&& !try_swap(map, n, m, cB, A.x, A.y))
        {
            Swap(map[A.x][A.y],
map[B.x][B.y]);
            return 0;
        }
        return 1;
    }
```