同济大学计算机系

彩球游戏实验报告

学　　号 1850772

姓　　名 张哲源

专　　业 计算机科学与工程

授课老师 沈坚老师

目录

1. 题目

彩球游戏综合演示

1.1. 题目要求

1. 内部数组，随机生成初始5个球
2. 内部数组，随机生成60%的球，寻找移动路径
3. 内部数组，完整版
4. 画出n*n的框架（无分隔线），随机显示5个球
5. 画出n*n的框架（有分隔线），随机显示5个球
6. n*n的框架，60%的球，支持鼠标，完成一次移动
7. cmd图形界面完整版

2. 整体设计思路

整体设计思路如下(题目7为例)

1. 底层构造二维数组,对二维数组进行一系列的操作,然后再打印出来,反应这个过程
2. BFS层序遍历找寻移动路径,同时判断能否正常返回,层序遍历的过程中可以利用队列之间层级的关系得到移动的路径,这样的移动路径比较短
3. 鼠标移动先通过转换鼠标光标为对应的数组坐标,然后通过记录两次点击的方法,记录终点和起点,再放入BFS 中判断是否能正常移动,并且移动得到路径。
4. 消除,先判断能不能消除,再消除,因为要考虑到一个球进入可能消除多个球的问题,所以要先判断消除的情况,再做消除的操作,而且先判断的话也可以得到分数,消除数量,为其他功能的实现提供了遍历
5. 更新变化后的信息

3. 主要功能的实现

题目7为例:

核心功能有画边框图,撒球,图中画球(这三个功能为一开始的准备)

中间循环部分

鼠标操作(内套BFS得到移动路径,画移动路径,撒下三个球,统计计数,更新)

鼠标右键或游戏结束,结束循环

(游戏结束规则:剩三个空格)(判断鼠标行为,右键即可退出)

游戏结束

```
void function7()
{
    int map[MAXROW][MAXCOL] = { 0 };
    coord size;
    coord now;
    coord pos = { 0,1 };
    int next[3] = { 0 };
    InputTips(size.x, size.y);//这一步得到行和列
    SeedMap(map, size,5);
    cct_cls();
    cct_getxy(now.x, now.y);
    cct_setfontsize("新宋体", 32);
    cct_setconsoleborder(size.x * 10, size.y * 4, size.x * 10, size.y * 4);
    Draw_Board_And_Line(map, size, pos,1);
    Draw_Ball_In_Map(map, size, pos);
    Mouse_Operate(map, size, pos);

}
```

一开始的准备

```
while (!GameOver(map,size))
{
    Draw_Ball_Next(next, update);//更新下一个出现的栏
    int opt = cct_read_keyboard_and_mouse(now.x, now.y, opt_kind, kbd1, kbd2);

    cheak = { now.x - pos.x,now.y - pos.y };
    int flag = 0;//判断是不是在中间的位置
    if (opt == CCT_MOUSE_EVENT)
    {
        cheak.x /= 2;//横坐标是空两格,保留两格,为了方便和y一起,除二操作
        if (cheak.x % 2 == 1 && cheak.y % 2 == 1)
        {
            cheak.x /= 2;//中间隔了1格
            cheak.y /= 2;
            //不越界即为正确
            if (cheak.x >= 0 && cheak.x < size.x && cheak.y >= 0 && cheak.y < size.y)
                flag = 1;
        }

        /*************************有效的鼠标操作*****************************/
        if (flag)//如果是在选中范围内
        {
            /*********************打印鼠标坐标 ******************************/
            cct_gotoxy(pos.x, pos.y + size.y * 2 + 2);//到最下面一行打印
            cout << "[当前鼠标]" << char(cheak.y + 'A') << "行" << cheak.x + 1 << "列";
            cout << "  ";
```

鼠标坐标的转换

```
/*************************选中球和移动球*************************/
if (opt_kind == MOUSE_LEFT_BUTTON_CLICK)//按下左键
{
    cct_showstr(pos.x, pos.y + size.y * 2 + 1, "[点击]

    if (map[cheak.y][cheak.x])//有球,让球变选择
    {
        if (record.x != -1)//先还原记录中的,没有就不还原
            Print_Ball(map, size, record, pos, normal);

        //再用打印表示选中的球,并且标记此球,把此球作为起点
        Print_Ball(map, size, cheak, pos, selected);
        record = cheak;//记录一下当前的选中
        begin = cheak;//选另一个球作为起点
    }
    else//没球也清除
    {
        if (record.x != -1)//上一个点不为空就恢复
        {
            Print_Ball(map, size, record, pos, normal);
            if (begin.x != -1)//如果有起点
                end = cheak;//终点也有了
        }
    }
```

点击操作得到起点坐标和终点坐标

```
/*****************得到起点和终点做移动判断*****************/
if (begin.x != -1 && end.x != -1)
{
    //cct_showstr(0,size.y*2+4,"可以移动");
    if (BFS(map, size, begin, end, path))
    {
        //打印可以移动的提示信息
        cct_gotoxy(pos.x, pos.y + size.y * 2 + 1);//到最下面一行打印
        cout << "[提示]可以从(" << char(begin.y + 'A') << "行" << begin.x + 1 << "列)";
        cout << "移动到(" << char(end.y + 'A') << "行" << end.x + 1 << "列)";
        //cout << "                                    ";
        //移动过程
        Draw_Move(map, size, pos, path, 150);//先根据路径画出移动轨迹
        Move(map, begin, end);//然后把内部的数组起点和终点更新了
        int score = Get_Score(map, size, end);//移动之后判断一下能不能消除,得到得分
        sum += score;

        clear[map[end.y][end.x]] += Get_Eliminate(map, size, end);//对应的下标累加消除的个数

        Eliminate(map, size, end);//消除下

        SeedMap(map, size, 1, next[0]);//移动完之后撒种子
        SeedMap(map, size, 1, next[1]);
        SeedMap(map, size, 1, next[2]);

        Draw_Ball_In_Map(map, size, pos);//移动后是否消除要重新画一下
        Draw_Score_Board(sum,update);//更新分数
        Count_Ball(map, size, count);//重新计数球
        Draw_Ball_In_Count(size, count, clear,update);//更新当前球的数量

        next[0] = 1 + rand() % 7;//移动完之后重新生成下一次的种子
        next[1] = 1 + rand() % 7;
        next[2] = 1 + rand() % 7;

    }
```

移动判断和移动后的更新

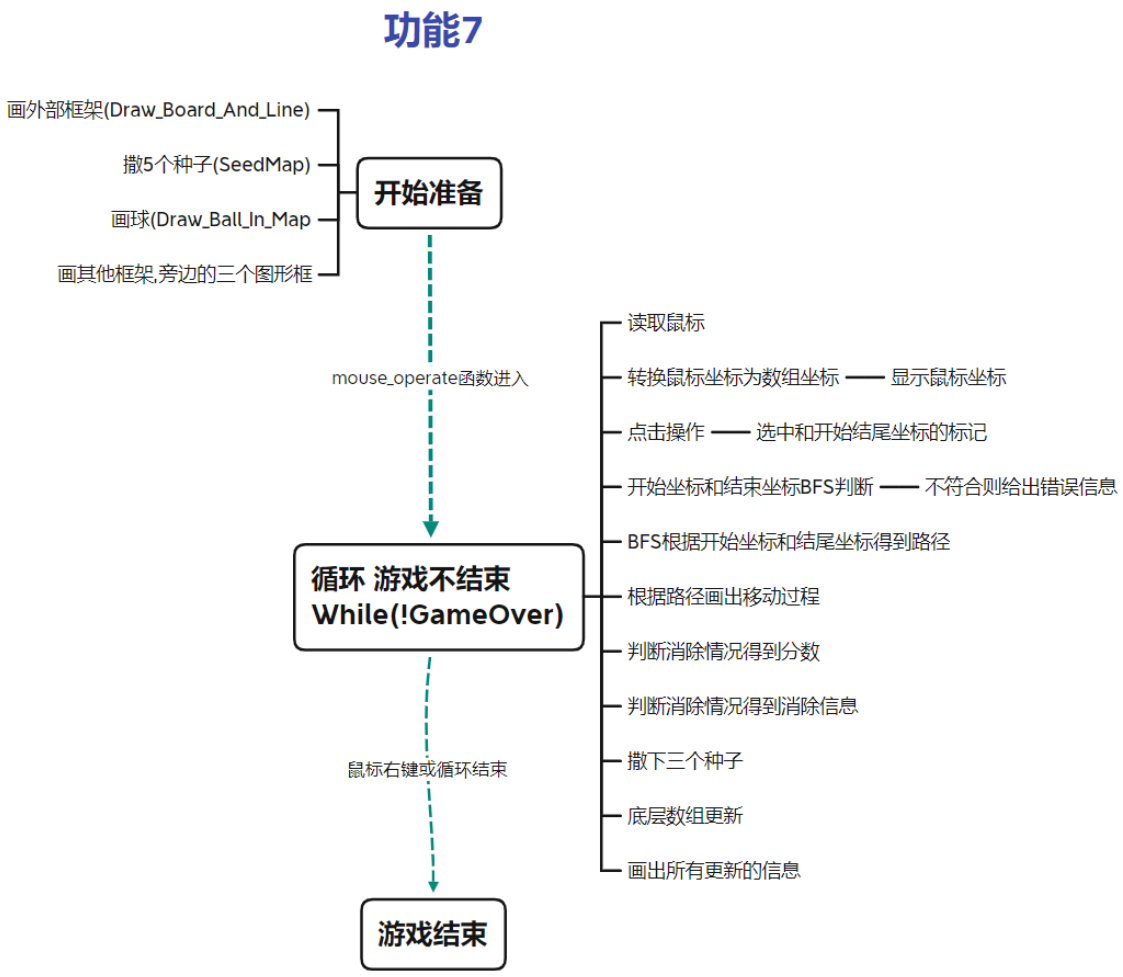
```
                    else
                    {
                        cct_gotoxy(pos.x, size.y * 2 + 1 + pos.y);//不能移动则不撤
                        cout << "[错误] 无法从" << char(begin.y + 'A') << begin.x + 1;
                        cout << "移动到" << char(end.y + 'A') << end.y + 1;
                    }

                    begin = { -1,-1 };//恢复起点终点,方便下一次判断
                    end = { -1,-1 };//恢复起点终点,方便下一次判断
                    //memset(path, -1, sizeof(coord));//恢复path,其实也可以不要这一步,BFS里面最后一步
                }
            }
            else if (opt_kind == MOUSE_RIGHT_BUTTON_CLICK)//右键退出
                break;
        }
    }
}
cct_showstr(pos.x, size.y * 2 + pos.y+1, "游戏结束                          ");

cct_disable_mouse();//设置禁用鼠标
```

不能正常移动的处理

流程框架图



4. 调试过程碰到的问题

问题 1:一个球进入会多个消除的问题

考虑到一个球进来,可能会消除多个球,比如说两行或者多行交叉了,所以把消除函数和判断函数拆开成了两个部分,先判断 后消除,把函数按照单个方向抽了出来.

```
int  Can_Eliminate(int map[MAXROW][MAXCOL], map_size size, coord pos)
{
    coord direction[4] = {
        { 1, 0},//横向
        { 0, 1},//纵向
        { 1,-1},//左斜
        { 1, 1} //右斜
    };
    //判断能不能消除,四种情况有一种就可以消除
    return (Can_Eliminate_Dir(map, size, pos,direction[0]) +
            Can_Eliminate_Dir(map, size, pos,direction[1]) +
            Can_Eliminate_Dir(map, size, pos,direction[2]) +
            Can_Eliminate_Dir(map, size, pos,direction[3]));
}
```

```
void Eliminate(int map[MAXROW][MAXCOL], map_size size, coord pos)
{
    coord direction[4] = {
        { 1, 0},//横向
        { 0, 1},//纵向
        { 1,-1},//左斜
        { 1, 1} //右斜
    };
    coord clear_path[4][9];
    //横向消除
    int Row = Can_Eliminate_Dir(map, size, pos, direction[0], clear_path[0]);

    //纵向消除
    int Col = Can_Eliminate_Dir(map, size, pos, direction[1], clear_path[1]);

    //左斜消除
    int LHT = Can_Eliminate_Dir(map, size, pos, direction[2], clear_path[2]);

    //右斜消除
    int RHS = Can_Eliminate_Dir(map, size, pos, direction[3], clear_path[3]);

    if(Row)
        Eliminate_Dir(map, clear_path[0]);

    if (Col)
        Eliminate_Dir(map, clear_path[1]);

    if (LHT)
        Eliminate_Dir(map, clear_path[2]);

    if (RHS)
        Eliminate_Dir(map, clear_path[3]);
}
```

问题 2:BFS 移动路劲的优化

一开始移动想着DFS就行了,反正能够到达终点,但是这样下来色块跑的非常慢,于是做了优化

```
typedef struct Node
{
    int x;
    int y;
    int parent_layer;
    int layer;//用来做bfs算法
}node;
```

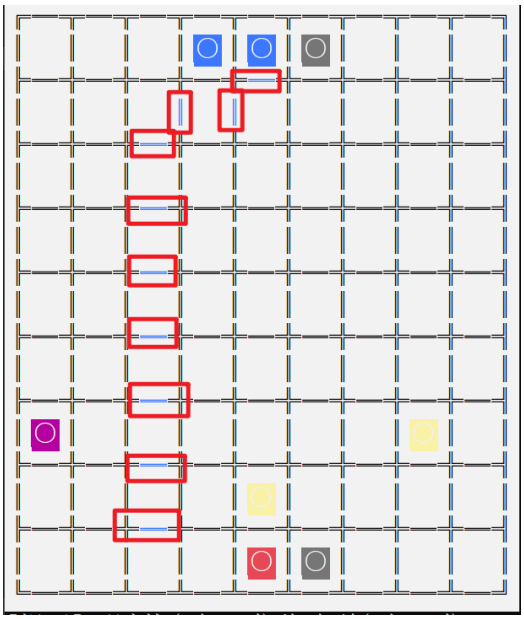
```
if (reached && path != NULL)
{
    //最后一层置-1
    path[queue[rear].layer + 1] = { -1,-1};
    while (rear != -1)//倒着往前装填到path里面
    {
        path[queue[rear].layer] = { queue[rear].x,queue[rear].y };
        rear = queue[rear].parent_layer;//顺着层一层一层到开始的位置
    }
}
```

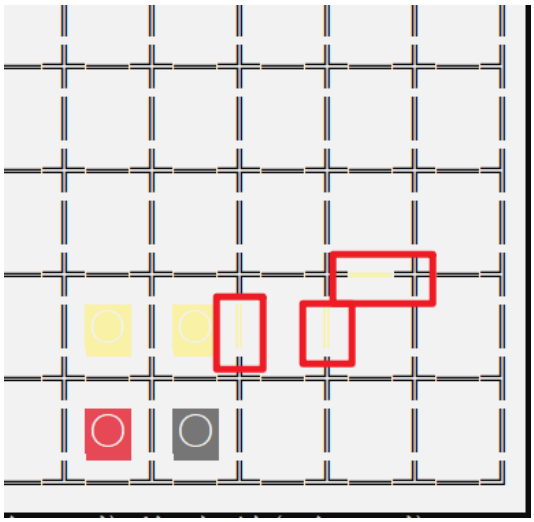
给BFS的结点里面做了有点向链表一样的东西,用这个来记录当前一层和上一层,

到时候只需要根据层保存到路径中去就可以了

移动基本上是直达的，此处为了方便记录路径，在程序中做了修改用颜色标记一下

（源程序没有格子的标记）





问题 3：鼠标操作的问题

整个最困难的感觉就是鼠标操作了

首先就是鼠标这个循环建立在哪里，调来调去，发现只能放到读取鼠标的外面

```
422    coord score_pos = { pos.x + 50 ,pos.y + 5
423
424    Count_Ball(map, size, count);
425    cct_enable_mouse();//使用鼠标
426    Draw_Ball_Next(next);
427    Draw_Score_Board(sum);
428    Draw_Ball_In_Count(size,count,clear);
429
430    while (!GameOver(map,size))
431    {
432        Draw_Ball_Next(next, update);//更新下
433        int opt = cct_read_keyboard_and_mouse
434
435        cheak = { now.x - pos.x,now.y - pos.y
436        int flag = 0;//判断是不是在中间的位置
437        if (opt == CCT_MOUSE_EVENT)
```

第二个就是坐标的转换，发现横向是纵向的两倍，这一点可以从控制台上面看出来，但是当时调的时候忘记了这个，都怪打印全角的空白框，让我一直以为是一个坐标. 想明白之后，所有的坐标打印都很好操作

```
void Print_Ball(int map[MAXROW][MAXCOL], map_size size, coord pos, coord board_pos, elem type)
{
    switch (type)
    {
        case normal://这里用15减去当前颜色可以避免原来的数组为0的时候,也可以打印出白色
            cct_showstr(pos.x * 4 + board_pos.x + 2, pos.y * 2 + board_pos.y + 1, CIRCLE, 15-map[pos
            break;
        case selected:
            cct_showstr(pos.x * 4 + board_pos.x + 2, pos.y * 2 + board_pos.y + 1, CHOOSE, 15-map[pos
            break;
        case box:
            cct_showstr(pos.x * 4 + board_pos.x + 2, pos.y * 2 + board_pos.y + 1, BLANK, COLOR_HWHIT
            break;
        default:
            break;
    }
    cct_setcolor();//恢复颜色,防止不必要的东西出现
}
```

在图中打印球

```
cheak = { now.x - pos.x,now.y - pos.y };
int flag = 0;//判断是不是在中间的位置
if (opt == CCT_MOUSE_EVENT)
{
    cheak.x /= 2;//横坐标是空两格,保留两格,为了方便和y一起,除二操作
    if (cheak.x % 2 == 1 && cheak.y % 2 == 1)
    {
        cheak.x /= 2;//中间隔了1格
        cheak.y /= 2;
        //不越界即为正确
        if (cheak.x >= 0 && cheak.x < size.x && cheak.y >= 0 && cheak.y < si
            flag = 1;
    }

    /*************************有效的鼠标操作*************************
    if (flag)//如果是在选中范围内
```

坐标转换

第三个就是如何消除选中和得到起点终点,这个逻辑有点绕.

```
if (opt_kind == MOUSE_LEFT_BUTTON_CLICK)//按下左键
{
    cct_showstr(pos.x, pos.y + size.y * 2 + 1, "[点击]

    if (map[cheak.y][cheak.x])//有球,让球变选择
    {
        if (record.x != -1)//先还原记录中的,没有就不还原
            Print_Ball(map, size, record, pos, normal);

        //再用打印表示选中的球,并且标记此球,把此球作为起点
        Print_Ball(map, size, cheak, pos, selected);
        record = cheak;//记录一下当前的选中
        begin = cheak;//选另一个球作为起点
    }
    else//没球也清除
    {
        if (record.x != -1)//上一个点不为空就恢复
        {
            Print_Ball(map, size, record, pos, normal);
            if (begin.x != -1)//如果有起点
                end = cheak;//终点也有了
        }
    }

    /***************得到起点和终点做移动判断***************/
    if (begin.x != -1 && end.x != -1)
    {
```

心得体会

第二次大作业,感觉要比第一次大作业顺利多了,起码知道了如何循序渐进的调,而且知道了哪些模块是不好写的,但是作为一个强迫症,我一直想让我的代码变得逻辑性很清晰,函数尽可能干净,但是最后写鼠标的时候还是无法避免的写成了屎山,分析了下原因感觉是因为,整个过程只有一个循环,而且不能使用全局变量,所以导致函数写起来可能要有很多参数，其次就是因为相似功能没有在一个函数设置不同的参数里实现，导致抽离的时候很难抽干净功能块。

(我承认我比较菜,我没有找到正确打开方式)

两个方面总结:

1. 程序结构的分离和函数功能的抽调

程序分为7个文件,不同的.cpp文件里面写不同的东西,如何做到逻辑清晰,功能按类别堆放感觉对后期调试非常重要,我把所有的画图类的东西都放到了CONSOLE里面,然后底层的逻辑功能都放到了tools里面,头文件里面用/*******/把这些函数做分隔,然后7个功能的主函数放到base里面,通过main跳转

这样就使得写模块的时候,现在头文件里面放好,一边写一个功能,一边能找到自己想要的功能,遇到没有的功能,能够马上写出来再填补到功能里面去.而且对函数的抽离也提供了极大的遍历.所以在写的时候，前面的功能都能顺利的实现，基本上是没有什么功能就现写什么功能，到鼠标那块部分有点卡。抽的不是很干净。

2. 函数功能逐渐的改写和增加

还记得一开始上课时讲函数时,就说了形式参数这个问题,以及函数功能的增加,自己在编写的时候经常能遇到后面要写一个函数,和之前的函数作用很像,或者要从之前的函数里面做修改,再用到我的函数里面去.自己在完成这个程序的时候,发现经常是一个函数一开始写是一个样子,但是经过几个功能增加了之后,这个函数就是另外一个样子了

过程中对函数的功能缝缝又补补,调的过程中,感觉有好多一些时间都花在了之前没想到这个功能,然后后续有需求,再改函数上面了,感觉一开始理清楚思路真的很重要,后续再写的时候就可以很好的拿之前的函数来用,而不是再对之前的函数做修改,然后在回过头再检查之前的函数能不能在之前的模块里正常运行了.

本学习的数据结构在BFS求路径那个地方还是用到了，本来想写一堆队列操作的，后来发现数组够用了（数组真是万能的东西……）

整个作业连写带调花了整整3天时间,写完还是比较有成就感的,从一开始一个简单的二维数组,到现在能够实现完整的游戏功能,内心还是非常有成就感的.

6. 附件：源程序

(base.cpp 以第七个功能的代码为例)

```
void function7()
{
    int map[MAXROW][MAXCOL] = { 0 };
    coord size;
    coord now;
    coord pos = { 0,1 };
    int next[3] = { 0 };
    InputTips(size.x, size.y);//这一步得到行和列
    SeedMap(map, size,5);
    cct_cls();
    cct_getxy(now.x, now.y);
    cct_setfontsize("新宋体", 32);
    cct_setconsoleborder(size.x * 10, size.y * 4, size.x * 10, size.y * 4);
    Draw_Board_And_Line(map, size, pos,1);
    Draw_Ball_In_Map(map, size, pos);
    Mouse_Operate(map, size, pos);
```

（console.cpp）

```
} void Draw_Board_And_Line(int map[MAXROW][MAXCOL], map_size size, coord pos,int mouse)
{
    cct_setcursor(CURSOR_INVISIBLE);//让光标不可见
    coord cmdsize, buffersize;
    cct_getconsoleborder(cmdsize.x, cmdsize.y, buffersize.x, buffersize.y);
    cct_gotoxy(0, 0);
    cout << "屏幕;" << buffersize.y << "行" << buffersize.x << "列" ;
    if (mouse)
        cout << "(鼠标右键退出)";
    cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
    cct_gotoxy(pos.x, pos.y);//到给定的位置打印表
    //打印表头
    cout << "  ╔";
    for (int j = 1; j < size.x; j++)
        cout << "═" << "╦";
    cout << "═" << "╗";//打印第一行的头
    //打印内部框架部分
    for (int i = 1; i <= size.y; i++)
    {
        cct_gotoxy(pos.x, pos.y + i * 2 - 1);
        cout << "  ║";
        for (int j = 0; j < size.x; j++)
            cout << BLANK << "║";
        cct_gotoxy(pos.x, pos.y + i * 2);
        if (i < size.y)//没到底
        {
            cout << "  ╠";
            for (int j = 1; j < size.x; j++)
                cout << "═" << "╬";
            cout << "═" << "╣";
        }
        else//打印最底部的一层框
        {
            cout << "  ╚";
            for (int j = 1; j < size.x; j++)
                cout << "═" << "╩";
            cout << "═" << "╝";
        }
    }
    cct_gotoxy(0, size.y * 2 + 2);//到图片的最下面两行
    cct_setcolor();//恢复颜色,
    //cct_setcursor(CURSOR_VISIBLE_NORMAL);//恢复鼠标
    cout << endl;
void Draw_Ball_In_Map(int map[MAXROW][MAXCOL], map_size size, coord pos)
{
    cct_setcursor(CURSOR_INVISIBLE);
    coord temp;
    for (temp.x = 0; temp.x < size.x; temp.x++)
    {
        for (temp.y = 0; temp.y < size.y; temp.y++)
        {
            if (map[temp.y][temp.x])
            {
                Print_Ball(map, size, temp, pos, normal);
            }
            else
                Print_Ball(map, size, temp, pos, box);
        }
    }
    cct_gotoxy(pos.x, pos.y + size.y * 2 + 2);//到图片的最下面两行
    cct_setcolor();//恢复颜色,
    //cct_setcursor(CURSOR_VISIBLE_NORMAL);//恢复鼠标
```

```
    cout << endl;
}
    }
void Mouse_Operate(int map[MAXROW][MAXCOL], map_size size, coord pos)
{
    int kbd1, kbd2;//键盘操作的两个参数,充数用的
    int opt_kind;//定义操作的类型
    int sum = 0;//总分
    int update = 1;//是否更新元素
    int count[8] = { 0 };
    int clear[8] = { 0 };//消除的元素
    int next[3];
    next[0] = 1 + rand() % 7;
    next[1] = 1 + rand() % 7;
    next[2] = 1 + rand() % 7;
    coord now;    //当前鼠标坐标
    coord cheak;//用于做检查的坐标,对应数组
    coord record = { -1,-1 };//用于记录当前位置,取消选择时方便恢复当前球的颜色
    coord begin = { -1,-1 };//用于做移动的起点
    coord end = { -1,-1 };//用于做移动的终点
    coord path[81];//移动的路径,最多应该有81次,9*9;
    coord score_pos = { pos.x + 50 ,pos.y + 50 };

    Count_Ball(map, size, count);
    cct_enable_mouse();//使用鼠标
    Draw_Ball_Next(next);
    Draw_Score_Board(sum);
    Draw_Ball_In_Count(size,count,clear);

    while (!GameOver(map,size))
    {
        Draw_Ball_Next(next, update);//更新下一个出现的栏
        int opt = cct_read_keyboard_and_mouse(now.x, now.y, opt_kind, kbd1, kbd2);

        cheak = { now.x - pos.x,now.y - pos.y };
        int flag = 0;//判断是不是在中间的位置
        if (opt == CCT_MOUSE_EVENT)
        {
            cheak.x /= 2;//横坐标是空两格,保留两格,为了方便和y一起,除二操作
            if (cheak.x % 2 == 1 && cheak.y % 2 == 1)
            {
                cheak.x /= 2;//中间隔了1格
                cheak.y /= 2;
                //不越界即为正确
                if (cheak.x >= 0 && cheak.x < size.x && cheak.y >= 0 && cheak.y < size.y)
                    flag = 1;
            }

            /*****************************有效的鼠标操作*******************************/
            if (flag)//如果是在选中范围内
            {
                /************************打印鼠标坐标 *******************************/
                cct_gotoxy(pos.x, pos.y + size.y * 2 + 2);//到最下面一行打印
                cout << "[当前鼠标]" << char(cheak.y + 'A') << "行" << cheak.x + 1 << "列";
                cout << "  ";

                /*****************************选中球和移动球*******************************/
                if (opt_kind == MOUSE_LEFT_BUTTON_CLICK)//按下左键
                {
                    cct_showstr(pos.x, pos.y + size.y * 2 + 1, "[点击]");

                    if (map[cheak.y][cheak.x])//有球,让球变选择
                    {
                        if (record.x != -1)//先还原记录中的,没有就不还原
                            Print_Ball(map, size, record, pos, normal);

                        //再用打印表示选中的球,并且标记此球,把此球作为起点
                        Print_Ball(map, size, cheak, pos, selected);
                        record = cheak;//记录一下当前的选中
                        begin = cheak;//选另一个球作为起点
                    }
                    else//没球也清除
                    {
                        if (record.x != -1)//上一个点不为空就恢复
                        {
                            Print_Ball(map, size, record, pos, normal);
                            if (begin.x != -1)//如果有起点
```

```
                                end = cheak;//终点也有了
                            }
                        }

                        /****************得到起点和终点做移动判断***************/
                        if (begin.x != -1 && end.x != -1)
                        {

    //cct_showstr(0,size.y*2+4,"可以移动");
                            if (BFS(map, size, begin, end, path))
                            {
                                //打印可以移动的提示信息

    cct_gotoxy(pos.x, pos.y + size.y * 2 + 1);//到最下面一行打印
                                cout << "[提示]可以从(" << char(begin.y + 'A') << "行" << begin.x + 1 << "列)";
                                cout << "移动到(" << char(end.y + 'A') << "行" << end.x + 1 << "列)";
                                //cout << "
";
                                //移动过程
                                Draw_Move(map, size, pos, path, 150);//先根据路径画出移动轨迹
                                Move(map, begin, end);//然后把内部的数组起点和终点更新了
                                int score = Get_Score(map, size, end);//移动之后判断一下能不能消除,得到得分
                                sum += score;


    clear[map[end.y][end.x]] += Get_Eliminate(map, size, end);//对应的下标累加消除的个数

                                Eliminate(map, size, end);//消除下

                                SeedMap(map, size, 1, next[0]);//移动完之后撒种子
                                SeedMap(map, size, 1, next[1]);
                                SeedMap(map, size, 1, next[2]);

    Draw_Ball_In_Map(map, size, pos);//移动后是否消除要重新画一下

    Draw_Score_Board(sum,update);//更新分数
                                Count_Ball(map, size, count);//重新计数球

    Draw_Ball_In_Count(size, count, clear,update);//更新当前球的数量

                                next[0] = 1 + rand() % 7;//移动完之后重新生成下一次的种子
                                next[1] = 1 + rand() % 7;
                                next[2] = 1 + rand() % 7;

                            }
                            else
                            {

    cct_gotoxy(pos.x, size.y * 2 + 1 + pos.y);//不能移动则不撒
                                cout << "[错误]无法从" << char(begin.y + 'A') << begin.x + 1;
                                cout << "移动到" << char(end.y + 'A') << end.y + 1;
                            }

                            begin = { -1,-1 };//恢复起点终点,方便下一次判断
                            end = { -1,-1 };//恢复起点终点,方便下一次判断
                            //memset(path, -1, sizeof(coord));//恢复path,其实也可以不要这一步,BFS里面最后一步做了-1
                        }
                    }
                    else if (opt_kind == MOUSE_RIGHT_BUTTON_CLICK)//右键退出
                        break;
                }
            }
    }
    cct_showstr(pos.x, size.y * 2 + pos.y+1, "游戏结束                    ");

    cct_disable_mouse();//设置禁用鼠标
    }
void Draw_Move(int map[MAXROW][MAXCOL], map_size size, coord pos,coord path[],int time)
{
    cct_setcursor(CURSOR_INVISIBLE);//让光标不
```

```
可见
    int color =15-map[path[0].y][path[0].x];//起点的颜色
    coord cur = path[0];//当前坐标
    coord next = { -1,-1 };//下一步的坐标
    coord dir = { -1,-1 };//下一步的方向
    for (int i = 1; path[i].x != -1; i++,cur=next)
    {
        next = { path[i].x,path[i].y };
        dir.x = next.x - cur.x;
        dir.y = next.y - cur.y;
        //先清除当前框的彩色
        cct_showstr(cur.x * 4 + pos.x + 2, cur.y * 2 + pos.y + 1, BLANK, COLOR_HWHITE, COLOR_HWHITE);
        Sleep(time);
        //然后向下一步方向位置移动一格,打印出彩色的方块
        cct_showstr(cur.x * 4 + pos.x + 2 + dir.x * 2, cur.y * 2 + pos.y + 1 + dir.y, CHOOSE,  color, COLOR_HWHITE);
        Sleep(time);
        //然后在当前位置要把横线补上
        if (dir.y == 0)//平移补竖,上下补横
            cct_showstr(cur.x * 4 + pos.x + 2 + dir.x * 2, cur.y * 2 + pos.y + 1 + dir.y, "‖", COLOR_HWHITE,  color);
        else
            cct_showstr(cur.x * 4 + pos.x + 2 + dir.x * 2, cur.y * 2 + pos.y + 1 + dir.y, "═", COLOR_HWHITE,  color);
        //然后到下一格继续打印新的球
        cct_showstr(next.x * 4 + pos.x * 2 + 2, next.y * 2 + pos.y + 1, CHOOSE, COLOR_HWHITE, color);
    }
} void Draw_Ball_In_Count(map_size size,int count[],int clear[], int update, coord pos)
{
    cct_setcursor(CURSOR_INVISIBLE);//让光标不可见
    if (!update)//如果不更新就打印边框
    {
        //宽度是12行,8列
        Draw_Board({ 12,8 }, COUNT_POS);
        for (int i = 0; i < 8; i++)
        {
            cct_showstr(pos.x + 2, pos.y + i+1, CIRCLE, 15-i, COLOR_HWHITE);
            cct_showstr(pos.x + 4, pos.y + i + 1, ":", COLOR_HWHITE, COLOR_BLACK);
        }
        cct_setcolor();//恢复颜色
    }
    for (int i = 0; i < 8; i++)
    {
        cct_gotoxy(pos.x + 5, pos.y + 1 + i);
        cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
        cout << setw(2) << setfill('0') << count[i] << "/" << "(";
        cout << setiosflags(ios::fixed) << setprecision(2);
        cout << setw(5) << (0.0 + count[i]) / (size.x * size.y) * 100;
        cout<< "%)" << "消除-" <<resetiosflags(ios::fixed)<< clear[i];
    }
    cct_setcolor();
    cout << endl << endl;

}
void Draw_Ball_Next(int next[], int update, coord pos)
{
    cct_setcursor(CURSOR_INVISIBLE);//让光标不可见
    if (!update)//如果不更新就打印边框
    {
        cct_showstr(pos.x, pos.y, "╔═══╦═══╗", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x, pos.y + 1, "‖", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x + 4, pos.y + 1, "‖", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x + 8, pos.y + 1, "‖", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x + 12, pos.y + 1, "‖", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x, pos.y + 2, "╚═══╩═══╝", COLOR_HWHITE, COLOR_BLACK);

    }
    cct_showstr(pos.x + 2, pos.y + 1, CIRCLE, 15-next[0], COLOR_HWHITE);
    cct_showstr(pos.x + 6, pos.y + 1, CIRCLE, 15-next[1], COLOR_HWHITE);
    cct_showstr(pos.x + 10, pos.y + 1, CIRCLE, 15-next[2], COLOR_HWHITE);
    cct_setcolor();
}
void Draw_Score_Board(int score, int update, coord pos)
{
    cct_setcursor(CURSOR_INVISIBLE);//让光标不可见
```

```
    if (!update)//如果不更新就打印边框
    {
        cct_showstr(pos.x, pos.y, "╔═══════════╗",COLOR_HWHITE,COLOR_BLACK);
        cct_showstr(pos.x, pos.y + 1, "║", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x + 2, pos.y + 1, "得分:      ", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x + 12, pos.y + 1, "║", COLOR_HWHITE, COLOR_BLACK);
        cct_showstr(pos.x, pos.y+2, "╚═══════════╝", COLOR_HWHITE, COLOR_BLACK);
    }
    cct_gotoxy(pos.x + 7, pos.y + 1);
    cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
    cout << score;
    cct_setcolor();
    }
void Print_Ball(int map[MAXROW][MAXCOL], map_size size, coord pos, coord board_pos, elem type)
{
    switch (type)
    {
        case normal://这里用15减去当前颜色可以避免原来的数组为0的时候,也可以打印出白色
            cct_showstr(pos.x * 4 + board_pos.x + 2, pos.y * 2 + board_pos.y + 1, CIRCLE, 15-map[pos.y][pos.x], COLOR_HWHITE);
            break;
        case selected:
            cct_showstr(pos.x * 4 + board_pos.x + 2, pos.y * 2 + board_pos.y + 1, CHOOSE, 15-map[pos.y][pos.x], COLOR_HWHITE);
            break;
        case box:
            cct_showstr(pos.x * 4 + board_pos.x + 2, pos.y * 2 + board_pos.y + 1, BLANK, COLOR_HWHITE, COLOR_HWHITE);
            break;
        default:
            break;
    }
    cct_setcolor();//恢复颜色,防止不必要的东西出现
    }
```