装 订 线

消消乐游戏

---C++程序设计实验报告

姓名：童佳燕

学号：1551445

班级：计算机科学与技术1班

指导教师：沈坚

完成日期：2017年3月5日

1. 实验题目：消消乐游戏

1.1 游戏规则

（1）游戏区域：5*5-9*9可选，行列由用户输入；

（2）初始状态：9种不同颜色的彩球随机出现，占满全部空间；

（3）操作方式：鼠标分别单机源位置和目标位置，使两者进行位置交换；

（4）得分规则： 任意行或任意列同色球满3个及以上，或同时满足行列要求的，可消除，得分为消除分数；

用户操作前的消除不得分；

（5）积分补充： 消除后，上方的小球按垂直方向下落，上方空余位置随机补充彩球至满，如果下落即补充后满足消除要求，则自动消除并积分；

（6）提示：每一轮自动提示可移动消除的彩球；

（7）游戏结束：无任何移动可消除彩球则游戏结束；

1.2 程序设计要求

（1）显示要求：可移动位置的彩球，被选中的彩球要有不同的显示效果；

彩球下落时，要有动画效果沿着通路进行移动；

消除时要有相应的动画效果；

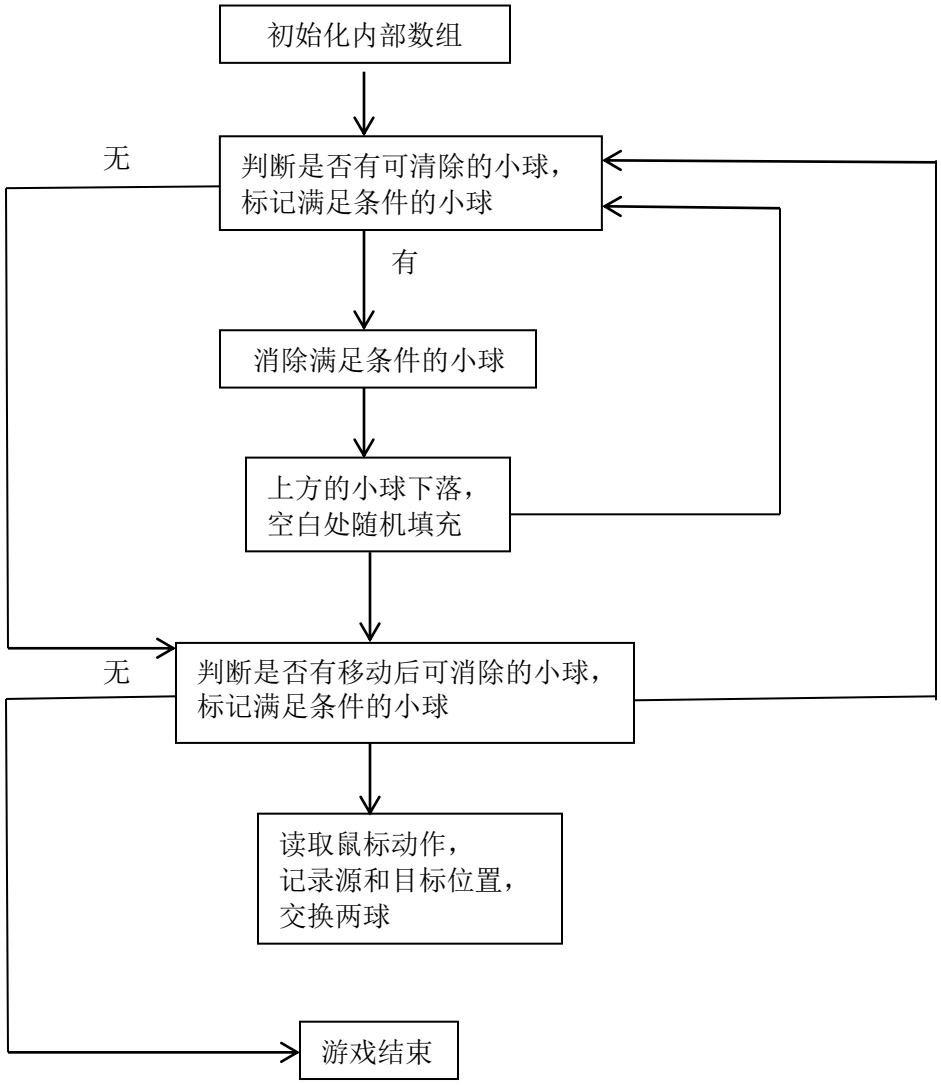
（2）不允许使用任何全局变量/指针/数组，允许使用全局的宏定义和常变量；

（3）尽可能公用函数，减少函数冗杂，所有函数尽量控制在50行左右；

2. 整体设计思路

编写程序过程中，先模拟一遍整个运行过程，拎出的那些步骤，再根据步骤需求考虑如何解决，想到解决方法之后，再次提取，找出需要的，小球的属性，将这些属性作为类型小球的数据成员，最后再一步一步倒推，测试是否成立。

在模拟过程中，我提取的主要步骤如下图：



根据流程图，已经确定的小球的属性包括，值（1-9），用于标记是否可消除的标记1，以及用于标记是否移动可消除的标记2；

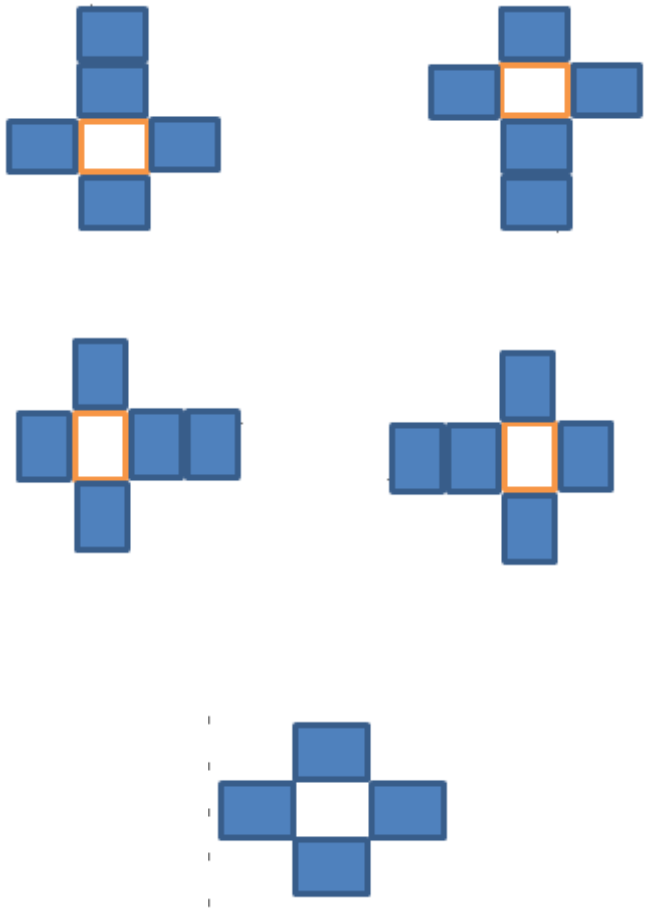
2. 主要功能的实现

2.1 判断是否可消除

从[0][0]开始遍历，判断是否满足ball[i][j-1].value==ball[i][j].value==ball[i][j+1].value，如果满足则对该三球进行erase=1的标记，下一次循环从j+1开始；同理对于列向的判断。

2.2 判断是否移动可消除

通过对demo的观察发现，移动可消除的情况总共有五种，分别是：



其中第一种到第四种分别又有三种情况，分别进行判断，标记，将可以移动的小球的remove值变动为1，为下面进行小球下落提供判断依据。

2.3 小球下落

从底部开始向上遍历，遇到remove=1的小球，进入循环，寻找它上方没有被消掉的小球，记录位置ii，也就是说ii位置的小球需要移动到i位置，其中ii小球的value值赋i位置，i位置的小

球“正常化”，即所有属性值，除value之外全都恢复到初始值；而ii位置的小球则需要“接替”i位置小球的属性，如remove=1；下次循环可以从上一轮的ii处开始，因为已经能够确定i-ii之间都是“空”的；

在这个过程中，需要注意的是ii>=0这个隐含条件，需要加入判断以及处理。

记录了小球移动的起点和终点位置，利用循环，加入延时，即可出现小球移动的动画效果。

3.调试过程碰到的问题

3.1 属性更新不彻底

测试过程中，发现可消除提示时有上一轮的残留，也就是某小球在上一轮是被判断为可移动消除的小球，即remove=1，而在下一轮实际上已经不是可移动消除的小球；而程序的输出是以remove为判断之一，对remove=1的小球做特殊处理输出；结合两者，考虑到，一，是remove的值没有及时更新；二，在输出的时候只更新了remove=1的小球的输出。

而实际上，BUG在这两个方面都涉及到了。

程序过程中，首先发现小球的属性的更新很凌乱，没有一个统一的更新，很容易遗漏；因此，修改过程中，在循环的头部，对所有小球的属性做了统一的更新。

其次，在变更remove之前，也就是在判断新一轮可移动消除之前，首先对输出做了更新，将上一轮移动可消除的小球的输出恢复到正常图案。

3.2 字符串未设置尾零

做第九小题的过程中，需要进行字符串连接“test\”，用户输入的学号字符串，“.dat”，先开了一个字符串数组file[30]，将上述三个字符串连接进去时，发现编译器并没有报错，然而运行没有结果。

我使用的排查方法时将下面一整块的先注释掉，然后一句一句的向上，填一句cout<<“a”；观察从什么位置开始，编译器就被迫停下了。测试发现，在strcat_s(file,学号数组)之前，cout语句可以被正常执行，而这之后，就执行出错。确定目标之后，很容易就联想到尾零的问题。于是给file数组初始化了一个尾零，问题才得以解决。

实际上，这个问题主要是因为学号是用字符数组存储的，是不会自动添加尾零的；而字符数组的操作需要遇到尾零才会停下。

4.心得体会

4.1 命名

这个程序中首次使用结构体数据，对结构体需要的属性进行定义，声明。而一个属性的命名应该是能够清楚的代表相应属性的功能。

事实上，我在程序设计的初期，对ball结构体的属性erase,remove常常感到困惑，两者容易混淆。而去查看前面的代码，弄清楚代表什么属性的时间成本，本是可以省下的，就因为命名不够清晰。

准备做的改进是，学习优秀程序员的源程序命名；扩展自身英语水平。

4.2 函数拆分

不得不说的，这一次大作业在函数拆分过程中比上期末大作业时，要更加细化，相对也要更加合理，利用率也有所提高。基本上需要做到一个过程一个函数，才能尽可能减少程序的冗杂。适当的添加参数，利用参数解决细微的差别引起的细微的功能变化。

4.3 属性变化流程图

程序中结构体ball对应的属性有5个，若干个函数分别都会对里面的成员进行操作；

当操作的函数一多，很容易丢了对一个属性变化的追踪。在后期测试检查时，纸笔还是很有必要的。

画出函数的流程图，在边上注明该函数对哪些对象的**属性进行了什么操作。流程图一画，就很清楚在哪些状态，这些对象的属性分别是怎样的情况，对于后期排查错误很有效。

5.附件：源程序

```
void eight(struct Ball(*ball)[10], struct Size *size)
{
    struct Place position[2] = { {0,0},{0,0} };
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    struct Place place;
    int &x = place.x;
    int &y = place.y;
    int begin_leap = 0,sum=0;//begin_leap用于控制是否开始记录
    int *ptr_sum = &sum;//sum用于分数记录
    input_size(size);
    init(ball, *size);
    print_withline(ball, *size);
    getxy(hout, x, y);
    while (1)
    {
        while (1)
        {
            /*每次进入都重写所有项的值，避免上一轮遗留下未改变的值，虽然有浪费，
            但程序更加直观，更安全*/
```

```
            for (int i = 0; i < size->rows; i++)
                for (int j = 0; j < size->cols; j++)
                {
                    ball[i][j].erase = 0;
                    ball[i][j].h_leap = 0;
                    ball[i][j].v_leap = 0;
                    ball[i][j].remove = 0;
                }
            if (!judge(ball, *size))
            {
                gotoxy(hout, 0, y + 1);
                setcolor(hout, COLOR_BLACK, COLOR_WHITE);
                cout << "\n初始已无可消除项" << endl;
                break;
            }
            else
            {
                sign(ball, *size);
                for (int i = 0; i < size->rows; i++)
                    for (int j = 0; j < size->cols; j++)
                        if (ball[i][j].erase == 1)
                            showstr(hout, 2 + j * 4, (i + 1) * 2, "◆", ball[i][j].value + 7,
COLOR_WHITE);
                eliminate(ball, *size,ptr_sum,begin_leap);//
                gotoxy(hout, 0, y);
                setcolor(hout, 0, COLOR_HWHITE);
                cout << "得分：" << *ptr_sum << endl;
                fall_block(ball, *size);
                fill(ball, *size);
            }
        }
        begin_leap+=1;//出了初始消除阶段后，开始记录消除球的个数
        if (!prompt_block(ball, *size))
            break;
        get_mouse(ball, *size, position,y);//获得鼠标点击的两颗球的位置（在数组中的位置）
        exchange(ball, *size, position);//交换位置

    }
    gotoxy(hout, 0, y + 3);
    cout << "game over! " << endl;
}
```

```
int judge(struct Ball(*ball)[10], struct Size size)
{
    int leap = 0;
    for (int i = 1; i < size.rows - 1; i++)
        for (int j = 0; j < size.cols; j++)
            if (ball[i][j].value == ball[i + 1][j].value &&ball[i][j].value == ball[i - 1][j].value)
            {
                leap = 1;
                ball[i][j].v_leap = 1;
            }
    for (int i = 0; i<size.rows; i++)
        for (int j = 1; j<size.cols - 1; j++)
            if (ball[i][j].value == ball[i][j + 1].value&&ball[i][j].value == ball[i][j - 1].value)
            {
```

装 订 线

```
                    leap = 1;
                    ball[i][j].h_leap = 1;
               }
     return leap;
}
```

```
void sign(struct Ball(*ball)[10], struct Size size)
{
     for (int i = 1; i < size.rows - 1; i++)
          for (int j = 0; j < size.cols; j++)
               if (ball[i][j].v_leap == 1)
               {
                    ball[i - 1][j].erase = 1;
                    ball[i][j].erase = 1;
                    ball[i + 1][j].erase = 1;
               }
     for (int i = 0; i<size.rows; i++)
          for (int j = 1; j < size.cols - 1; j++)
               if (ball[i][j].h_leap == 1)
               {
                    ball[i][j - 1].erase = 1;
                    ball[i][j].erase = 1;
                    ball[i][j + 1].erase = 1;
               }
}
```

```
int prompt_1(struct Ball(*ball)[10], struct Size size)
{
     int leap = 0;
     for (int i = 0; i < size.rows - 2; i++)
          for (int j = 0; j < size.cols; j++)
          {
               if (ball[i][j].value == ball[i + 1][j].value)
               {
                    if (ball[i + 3][j].value == ball[i][j].value && i < size.rows - 3)
                    {
                         leap = 1;
                         ball[i + 2][j].remove = 1;
                         ball[i + 3][j].remove = 1;
                    }
                    if (ball[i + 2][j + 1].value == ball[i][j].value && j < size.cols - 1)
                    {
                         leap = 1;
                         ball[i + 2][j].remove = 1;
                         ball[i + 2][j + 1].remove = 1;
                    }
                    if (ball[i + 2][j - 1].value == ball[i][j].value && j > 0)
                    {
                         leap = 1;
                         ball[i + 2][j].remove = 1;
                         ball[i + 2][j - 1].remove = 1;
                    }
               }
          }
     return leap;
```

装 订 线

```
}
```

```
int prompt_5(struct Ball(*ball)[10], struct Size size)
{
    int leap = 0;
    for(int i=0;i<size.rows;i++)
        for(int j=1;j<size.cols-1;j++)
            if (ball[i][j - 1].value == ball[i][j + 1].value)
            {
                if (ball[i - 1][j].value == ball[i][j - 1].value &&i>=1)
                {
                    leap = 1;
                    ball[i][j].remove = 1;
                    ball[i - 1][j].remove = 1;
                }
                if (ball[i + 1][j].value == ball[i][j - 1].value && i<size.cols-1)
                {
                    leap = 1;
                    ball[i][j].remove = 1;
                    ball[i+1][j].remove = 1;
                }
            }
    for(int i=1;i<size.rows-1;i++)
        for(int j=0;j<size.cols;j++)
            if (ball[i-1][j ].value == ball[i+1][j].value )
            {
                if (ball[i][j-1].value == ball[i+1][j].value &&j>=1)
                {
                    leap = 1;
                    ball[i][j - 1].remove = 1;
                    ball[i][j].remove = 1;
                }
                if (ball[i + 1][j].value == ball[i][j+1].value &&j<size.cols -1)
                {
                    leap = 1;
                    ball[i][j].remove = 1;
                    ball[i + 1][j].remove = 1;
                }
            }
    return leap;
}
```

```
void eliminate(struct Ball(*ball)[10], struct Size size,int *ptr_sum,int begin_leap)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    for (int i = 0; i<size.rows; i++)
        for (int j = 0; j < size.cols; j++)
            if (ball[i][j].erase == 1)
            {
                showstr(hout, 2 + j * 4, (i + 1) * 2, "※", ball[i][j].value + 7, COLOR_WHITE);
                if(begin_leap!=0)
                    (*ptr_sum)++;
                Sleep(200);
                showstr(hout, 2 + j * 4, (i + 1) * 2, "  ", COLOR_HWHITE, COLOR_WHITE);
```

```
            }
}
```

```
void fall_block(struct Ball(*ball)[10], struct Size size)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    for (int j = 0; j<size.cols; j++)
        for (int i = size.rows - 1; i >= 0; i--)
        {
            if (ball[i][j].erase == 1)
            {
                int ii = i - 1;
                for (; ii >= 0; ii--)
                    if (ball[ii][j].erase == 0)
                        break;
                if (ii >= 0)
                {
                    ball[i][j].erase = 0;
                    ball[i][j].value = ball[ii][j].value;
                    ball[ii][j].erase = 1;

                    ball[i][j].h_leap = 0;
                    ball[i][j].v_leap = 0;
                    ball[i][j].remove = 0;

                    for (int k = ii; k < i; k++)
                    {
                        Sleep(60);
                        showstr(hout, 2 + j * 4, (k + 1) * 2, "  ", COLOR_HWHITE, COLOR_WHITE);
                        Sleep(30);
                        showstr(hout, 2 + j * 4, (k + 2) * 2, "○", ball[i][j].value + 7,
COLOR_WHITE);
                    }

                }
            }
        }
}
```

```
int prompt_block(struct Ball(*ball)[10], struct Size size)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    int leap = 0;
    for (int i = 0; i<size.rows; i++)
        for (int j = 0; j<size.cols; j++)
            showstr(hout, 2 + j * 4, (i + 1) * 2, "○", ball[i][j].value + 7, COLOR_WHITE);
    if (prompt_1(ball, size) + prompt_2(ball,size) + prompt_3(ball, size) + prompt_4(ball, size) +
prompt_5(ball, size))
    {
        leap = 1;
        for(int i=0;i<size.rows ;i++)
            for(int j=0;j<size.cols;j++)
                if(ball[i][j].remove ==1)
                    showstr(hout, 2 + j * 4, (i + 1) * 2, "◎", ball[i][j].value + 7, COLOR_WHITE);
    }
```

```
    return leap;
}
```

```
void get_mouse(struct Ball(*ball)[10], struct Size size,struct Place position[2],int Y)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    const HANDLE hin = GetStdHandle(STD_INPUT_HANDLE);

    int loop = 1;
    int click_begin = 0;
    enable_mouse(hin);
    int x,y,action;
    setcursor(hout, CURSOR_INVISIBLE);
    while (loop)
    {
        action = read_mouse(hin, x, y);//读取鼠标的动作和位置
        gotoxy(hout, 0, Y+1);
        setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
        int y_i = (y - 1) / 2;//0-rows-1
        int x_j = x / 4;//0-cols-1
        if ((y - 1) / 2 + 1 > 0 && (y - 1) / 2 + 1 <= size.rows && (x / 4 + 1) > 0 && (x / 4 + 1)
<= size.cols)
        {
            cout << "[当前坐标]" << char(y_i + 'A') << "行" << x_j + 1 << "列";
            switch (action)
            {
            case MOUSE_RIGHT_BUTTON_CLICK:
                end();
                break;
            case MOUSE_LEFT_BUTTON_CLICK:
                if (ball[y_i][x_j].remove ==1&& click_begin == 0)
                {
                    position[0] = {x_j,y_i};
                    showstr(hout, 2 + position[0].x * 4, position[0].y * 2 + 2, "●",
ball[y_i][x_j].value + 7, COLOR_WHITE);
                    click_begin++;
                    break;
                }
                else if (ball[y_i][x_j].remove == 1&& abs(y_i-position[0].y
+x_j-position[0].x)==1)//
                {
                    position[1] = { x_j,y_i };
                    loop = 0;
                    break;
                }

            }
        }
    }
}
```