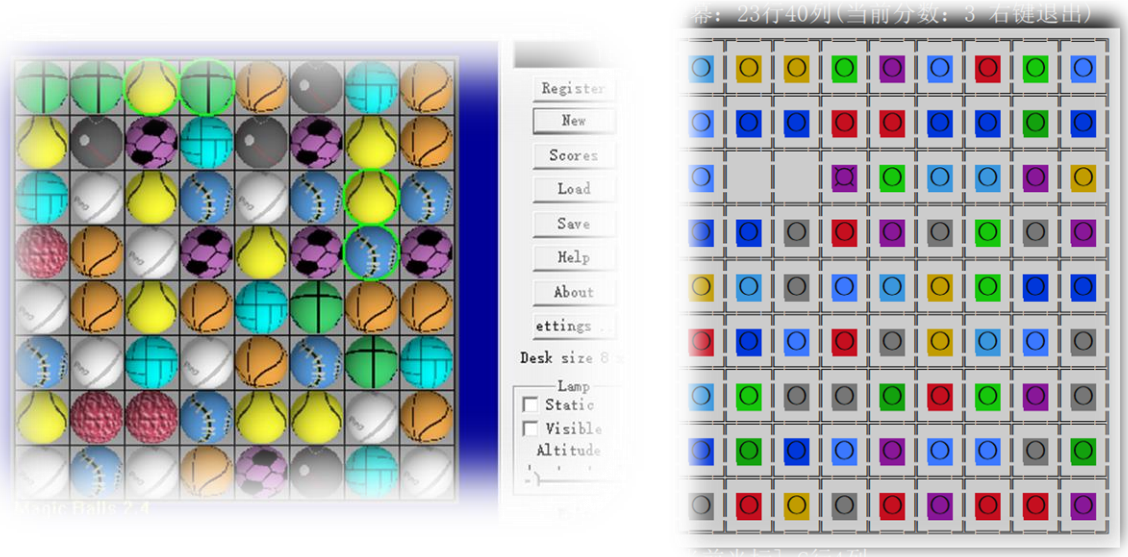
彩球游戏实验报告

2353726 信01 付煜超 2024.6.4完成



1. 彩球游戏实验报告

1.1. 题目及基本要求描述

题目要求按照Windows中的彩球游戏设计一个类似的游戏，这个游戏的设计需要从最基本的数组操作开始，逐步演化成一个完整的伪图形界面以及可以进行彩球消除的功能。此外，还要求使用多个源代码文件来组织和管理代码，以便更好地规划、存放和维护函数。这一系列任务按照要求逐步展开：首先实现数组操作，包括生成数字、消除元素、更新数组等基本逻辑；接着通过编程逐步构建伪图形界面，为之后的操作做准备。任务4567则主要是为任务8和9提供支持，这些任务涉及将游戏元素之间的联系从没有框架到构建有框架的视觉效果，以及模拟初始界面的消除下落过程。任务8要求实现可以使用鼠标控制来选中可交换的彩球，为后续任务提供基础。最后，任务9是将彩球游戏的伪图形玩法完全还原，需要考虑游戏的结束条件、选中和取消选中元素、消除、下落和填充等多个方面，以确保玩家能够完整地体验游戏。这个设计过程将对学习如何合理规划函数的安排、存放和整体代码组织提供宝贵的经验，有助于开发人员了解如何逐步构建一个复杂且完整的程序。

1.1.1. 三级小标题（如果有，宋体，小四，加粗，段前段后间距各1行）

给出题目及基本要求描述，可参考对应题目描述，但不建议照抄，正文采用宋体，黑色，五号字，行距为1.25倍

2. 整体设计思路

在设计一个完整的彩球游戏项目时，细致的规划和分解任务是非常关键的。将整个程序分为多个部分是一个明智的做法，每个部分都应该负责特定的功能或逻辑，以确保代码的模块化和整体性。以下是对你所述内容的进一步扩展，涵盖更多细节和设计思路。

首先，主函数部分在整个程序中扮演着关键的角色。它应该负责处理用户的输入、菜单选择以及确定行列等操作。通过主函数，程序可以根据用户的选择来驱动不同的逻辑流程，例如开始游戏、显示帮助菜单或退出游戏。

其次，菜单函数部分是另一个重要的组成部分。这部分代码应该负责绘制游戏的菜单界面，让用户可以方便地选择不同的游戏选项。通过菜单函数，用户可以与游戏进行交互，选择游戏模式或设置游戏参数等。将菜单选项值返回给主函数可以确保选项的有效处理。

在数组函数部分，我设计了三个12*12大小的数组，用于记录随机数、标记可消除和可交换项，以及存储伪图形界面的颜色信息。这种数组的设计是为了方便管理游戏的逻辑和数据。通过这些数组，可以检测可消除的彩球组合，进行下落操作并填充新的彩球。

接下来，checkdig函数的作用是检查是否存在三个或以上的彩球可以消除。这个步骤至关重要，因为游戏的核心玩法之一就是消除相同颜色的彩球。将符合消除条件的彩球标记后，dropdig函数负责将这些彩球消除并使其他彩球下落，以保持游戏界面的连贯性。最后，新值填充函数用于在空白区域填充新的彩球，准备下一轮的游戏操作。

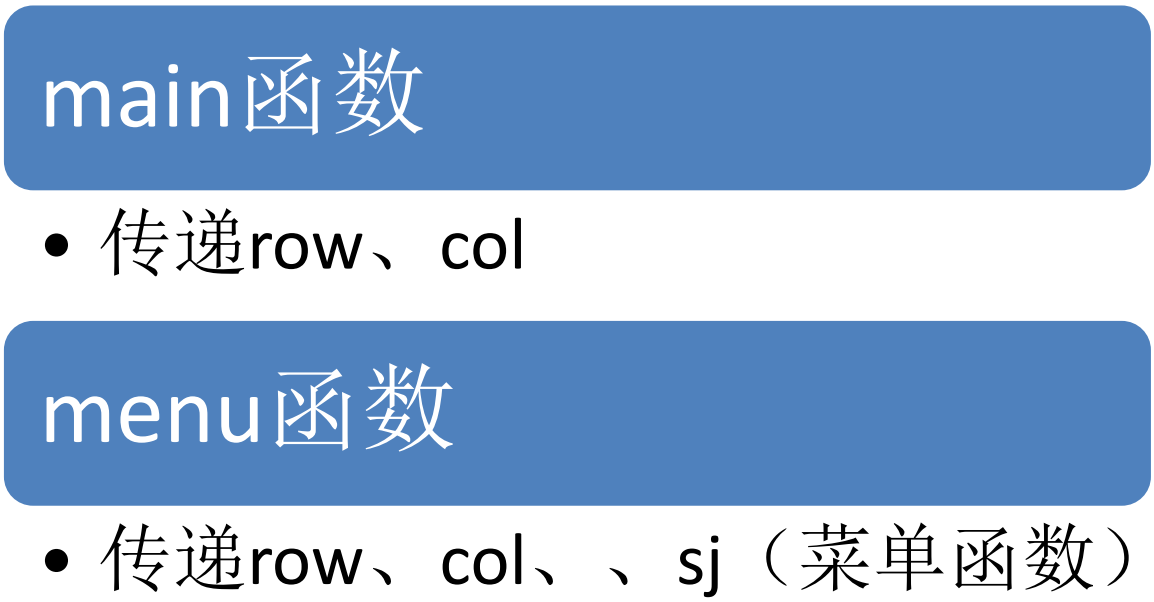
然后是对伪图形界面的设计思路。有内部分界线的界面可以通过在数据基础上进行相应的绘制操作来实现。在打印边框时，您选择了逐步填充四个边角、周围区域和中心图形。这种逐步填充的方式能够确保界面的完整性和美观性。同时，在有边框的界面设计中，不同类型的边框可以通过不同的打印方式来加以区分，这有助于复现demo中的效果。

最后，实现伪图形的游戏的设计需要考虑到交互性和游戏性。首先要判断初始情况是否存在可消除的彩球组合，如果有则进行消除操作并让其他彩球下落，否则提示玩家下一步的操作。在选中可交换项后，通过判断交换后是否存在可消除的组合，来确定交换是否有效。这种循环式的判断和操作是游戏设计中常见的模式，确保玩家在游戏中可以不断进行有效的交互操作。

在游戏进行过程中，需要不断检测是否有可消除项，处理玩家输入，更新游戏状态并判断游戏是否结束。这些步骤相互联系，构成了一个完整的游戏流程。通过合理设计和划分任务，可以更有效地开发出具有良好游戏体验的彩球游戏。因此，在实现游戏的过程中，应该注重代码的组织结构、模块化设计和逻辑合理性，以确保项目的可维护性和可扩展性。

3. 主要功能的实现

可以画出主要结构的流程框图，也可以其它描述方式，正文采用宋体，黑色，五号字，行距为1.25



dig_ini函数

- 设置数组每个数值

checkclear函数

- 检查可消除项

checkdig函数

- 给相应的值上色

dropdig函数

- 除0下落操作

supplydig函数

- 填充新值

ini_struct函数

- 画出伪图形界面的基本框架

draw_struc函数

- 实现伪图形界面的游戏

graph_check_clear函数

- 负责检查是否有可交换项

4. 调试过程碰到的问题

碰到的问题及解决方法，如果问题较多，选主要的即可，一般不超过3个，正文采用宋体，黑色，五号字，行距为1.25倍

首先，我需要找到一种方法来判断填充新值后数组是否可以进行消除。为了解决这个问题，我设定了一个循环来进行判断。然而，在调试过程中，我花了一段时间思考如何从循环中排除杂项，以确保循环只进行数值的改变。

其次，我在判断可交换项时遇到了困难。由于缺乏有效的判断方法，我只能一个一个地列举出可能

的情况，例如0x00。我穷举了12种情况，这在调试期间导致了巨大的工作量。每当发现问题时，我都不得不重新检查每个判断条件。

第三个主要问题是在第9题中实现伪图形时。在选中第一个图形后，有时候我发现无法选中第二个图形，尽管已经有了可交换的提示。在判断过程中，它显示无法进行交换。经过长时间的搜索，我最终发现问题出在之前编写的消除判断条件的边界条件上。我在处理x方向时遗漏了一种情况。

5. 心得体会

在我解决问题的过程中，我经历了一系列的调试和问题排查。这个过程中，我遇到了一些挑战和困难，但也从中获得了宝贵的经验和教训。

首先，第一个主要问题是如何判断填充新值后的数组是否还能消除。这个问题需要通过一个循环来判断，但是在建立循环的过程中，我花费了相当长的时间。我意识到，为了保证循环只包含数值的改变，我需要将杂项排除在循环之外。为了解决这个问题，我进行了一些调试和思考。最终，我通过合理的逻辑和条件判断，成功将杂项剔除出循环。这个问题的解决使我深刻认识到在编写循环时，确保正确性和简洁性的重要性。

其次，第二个主要问题是在判断可交换项时遇到了困难。缺乏有效的判断方法使得我不得不一个一个地列举可能的情况。这个问题增加了调试的工作量，并且容易导致遗漏或错误的判断。为了解决这个问题，我进行了详细的调试和检查。每当出现问题时，我都会仔细回顾判断条件，并确保每一种情况都被正确地考虑到。这个问题的解决使我意识到，对于没有直接的判断方法的情况，合理地列举和检查可能的情况是一种有效的方法。

第三个主要问题是在实现伪图形时出现的。在选中第一个图形后，有时候无法选中第二个图形，尽管存在可交换的提示。经过艰苦的调试过程，我最终发现问题出在我之前编写的消除判断条件的边界条件上。我在处理 x 方向时漏掉了一种情况。这个问题的解决需要我对代码进行细致的检查，并修复了条件判断的错误。我从中学到了对于边界情况的处理要格外小心，并且要在编写代码之前进行全面的思考和测试。

整个调试过程中，我意识到了调试的重要性和艰巨性。解决问题需要耐心和细致的思考，以及具备良好的逻辑能力和技术基础。此外，我还学会了如何充分利用调试工具和技巧，例如打印变量的值、逐行调试、追踪函数调用等。这些工具和技巧对于定位和修复问题非常有帮助。

在整个调试过程中，我体会到了解决问题的乐趣和成就感。每当我成功解决一个问题时，我都能感受到自己技能的提升和成长。此外，我也深刻认识到在开发过程中，问题是不可避免的，而且从中学习并提升自己的能力是重要的。

总结起来，通过解决这些调试中出现的问题，我不仅解决了具体的技术问题，还提高了自己的思考能力、逻辑能力和分析能力。我相信这些经验和教训将对我的未来工作和学习有着积极的影响。在面对类似的问题时，我将更加自信和有能力去解决它们。同时，我也会继续学习和探索新的调试方法和技巧，以提高自己的技术水平。

6. 附件：源程序

注意排版格式，对齐方式，可以采用两列排版，正文采用宋体，小五，行距为单倍行距

不需要全部源程序， main函数、菜单选择、输入输出错误等常规内容建议舍弃，挑部分核心代码贴上来即可。

```
void dig_ini(int row, int col, int sj)
{
    srand(static_cast<unsigned int>(time(nullptr)));


    char dig[12][12] = { 0 };
    char color[12][12] = { 0 };

    for (int i = 1; i <= row; i++)
    {
        for (int j = 1; j <= col; j++)
        {
            dig[i][j] = 1 + rand() % (9);
        }
    }
}

void dropdig(int row, int col, char dig[12][12], char color[12][12], int x, int sj)
{
    cout << endl;
    int row0 = 0;
    char letter = 0;
    if (x == 1)
    {
        for (int j = 1; j <= col; j++)
        {
            int times = 0;
            for (int i = 1; i <= row; i++)
            {

                if (color[i][j] == 1)
                {
                    color[i][j] = 0;
                    for (int k = i - 1; k > times; k--)
                    {
                        dig[k + 1][j] = dig[k][j];
                        color[k + 1][j] = color[k][j];
                    }
                    times++;
                    dig[times][j] = 0;
                    color[times][j] = 1;
                }
            }
        }
        row0 = 0;
        for (int i = 0; i <= col; i++, row0++)
        {
            if (i == 0)
                cout << "  |";
```

```
            else
                cout << std::right << setw(3) << row0;
        }
        cout << endl;
        for (int i = 0; i <= col; i++)
        {
            if (i == 0)
                cout << "--+";
            else
                cout << "---";
        }
        cout << "-" << endl;
        letter = 'A';
        for (int i = 1; i <= row; i++)
        {
            cout << letter << " |";
            for (int j = 1; j <= col; j++)
            {
                if (color[i][j] == 1)
                {
                    cout << "  ";
                    cct_setcolor(14, 4);
                    cout << int(dig[i][j]);
                    cct_setcolor();
                }
                else
                {
                    cout << std::right << setw(3) << int(dig[i][j]);
                }
            }
            cout << endl;
            letter++;
        }
        supplydig(row, col, dig, color, sj);
    }
    if (x == 0)
    {
        cout << "初始无可消除项，本小题无法测试，请再次运行" << endl;
        cout << "本小题结束，请输入End继续...";
        int x = 0, y = 0;
        cct_getxy(x, y);
        cin.clear();
        cin.ignore(1024, '\n');
        while (1)
        {
            char jieshu[2048] = { 0 };
            for (int i = 0; ; i++)
            {
                jieshu[i] = getchar();
                if (jieshu[i] == '\n')
                {
                    break;
                }
            }
```

```
            if (strlow(jieshu) == 0)
            {
                cct_cls();
                break;
            }
            else
            {
                cct_gotoxy(x, y);
                for (int i = 0; i < 3; i++)
                {
                    jieshu[i] = 0;
                }
                cct_gotoxy(x, y);
                cout << "   " << endl;
                cout << "输入错误，请重新输入";
                cct_gotoxy(x, y);
                continue;
            }
        }
    }
}

if (maction == MOUSE_LEFT_BUTTON_CLICK && color[int(Y / 2)][int((X + 2) / 4)] == 1 && X1 == 0 && Y1
== 0)
{
                cct_gotoxy(0, 2 * row + 2);
                output = 'A' + Y / 2 - 1;
                cout << "当前选择" << output << "行" << int((X + 2) / 4) << "列
";
                Sleep(500);
                X1 = int((X + 2) / 4);
                Y1 = int(Y / 2);
                cct_showstr(4 * X1 - 2, 2 * Y1, str, piccolor[Y1][X1], 7, 1, 2);
                cct_setcolor();
}
//选中第一个区域
else if (maction == MOUSE_LEFT_BUTTON_CLICK && color[int(Y / 2)][int((X + 2) / 4)] == 1 && X1 != 0 &&
Y1 != 0)
{
                X2 = int((X + 2) / 4);
                Y2 = int(Y / 2);
                output = 'A' + Y2 - 1;
                if (X1 == X2 && Y1 == Y2)
                {
                    output = 'A' + Y2 - 1;
                    cct_showstr(4 * X1 - 2, 2 * Y1, str, piccolor[Y1][X1], 0, 1, 2);
                    cct_setcolor();
                    cct_gotoxy(0, 2 * row + 2);
                    cout << "当前选择" << output << "行" << int((X + 2) / 4) << "列
";
                    cct_setcolor();
                    Sleep(200);
                    X1 = 0;
                    X2 = 0;
```

```
                    Y1 = 0;
                    Y2 = 0;
                    continue;
                }
                else if ((abs(X1 - X2) == 1 && abs(Y1 - Y2) == 0) || (abs(X1 - X2) == 0 && abs(Y1 - Y2) == 1) && color[Y2][X2] == 1)
{

                int m = dig[Y2][X2], n = piccolor[Y2][X2];
                dig[Y2][X2] = dig[Y1][X1];
                dig[Y1][X1] = m;
                piccolor[Y2][X2] = piccolor[Y1][X1];
                piccolor[Y1][X1] = n;
                //交换两组数据

                xiao = 0;
                for (int i = 1; i <= row; i++)
                {
                                for (int j = 1; j <= col; j++)
                                {
                                    color[i][j] = 0;
                                }
                }
                //初始化颜色
                for (int i = 1; i <= row; i++)
                {
                                for (int j = 1; j <= col - 2; j++)
                                {
                                    int k = 1;
                                    while (1)
                                    {
                                        if (dig[i][k + j] != dig[i][j])
                                            break;
                                        else
                                            k++;
                                    }
                                    if (k >= 3)
                                    {
                                        for (int m = 0; m < k; m++)
                                        {
                                            color[i][m + j] = 1;
                                        }
                                        xiao = 1;
                                    }
                                }
                }
                //确定行数需要上色的数字
                for (int i = 1; i <= col; i++)
                {
                                for (int j = 1; j <= row - 2; j++)
                                {
                                    int k = 1;
                                    while (1)
                                    {
```

```
                                    if (dig[j + k][i] != dig[j][i])
                                        break;
                                    else
                                        k++;
                                }
                                if (k >= 3)
                                {
                                    for (int m = 0; m < k; m++)
                                    {
                                        color[j + m][i] = 1;
                                    }
                                    xiao = 1;
                                }
                            }
                        }
                        //确定列数数需要上色的数字

                        if (xiao == 0)
                        {
                                    dig[Y1][X1] = dig[Y2][X2];
                                    dig[Y2][X2] = m;
                                    piccolor[Y1][X1] = piccolor[Y2][X2];
                                    piccolor[Y2][X2] = n;
                                    //换回去
                                    cct_gotoxy(0, 2 * row + 2);
                                    output = 'A' + Y2 - 1;
                                    cout << "不能交换" << char('A' + Y1 - 1) << "行" << X1 << "列 <=>
" << output << "行" << X2 << "列";
                                    Sleep(100);

                                    X1 = 0;
                                    Y1 = 0;
                                    X2 = 0;
                                    Y2 = 0;
                                    m = 0;
                                    n = 0;
                                    //重置X2,Y2条件

                                    for (int i = 1; i <= row; i++)
                                    {
                                        for (int j = 1; j <= col; j++)
                                        {
                                            color[i][j] = 0;
                                        }
                                    }
```