HARBIN INSTITUTE OF TECHNOLOGY

哈尔滨工业大学

Java 程序设计课程报告

题目：CubeCrash

学号	姓名	具体明确的小组分工说明 （小组负责人/PPT 制作人/报告撰写人/具体的程序开发分工等）
		报告撰写人，负责主要程序开发
		小组负责人，负责部分程序开发
		参与部分程序编写
		参与部分程序编写
		PPT 制作人，负责部分程序开发

目　录

Java 语言程序设计课程报告.....错误!未定义书签。
目　录.....- 1 -
报告撰写说明.....错误!未定义书签。
1.游戏概述.....- 2 -
2.游戏设计原理.....- 2 -
3.核心源代码说明.....- 4 -
4.游戏操作说明.....- 7 -
5.总结与感悟.....- 9 -

1.游戏概述

游戏的功能：

色彩明朗丰富的消除方块程序，简单愉快的小游戏，适合工作学习忙碌之余轻松一下。

玩法：

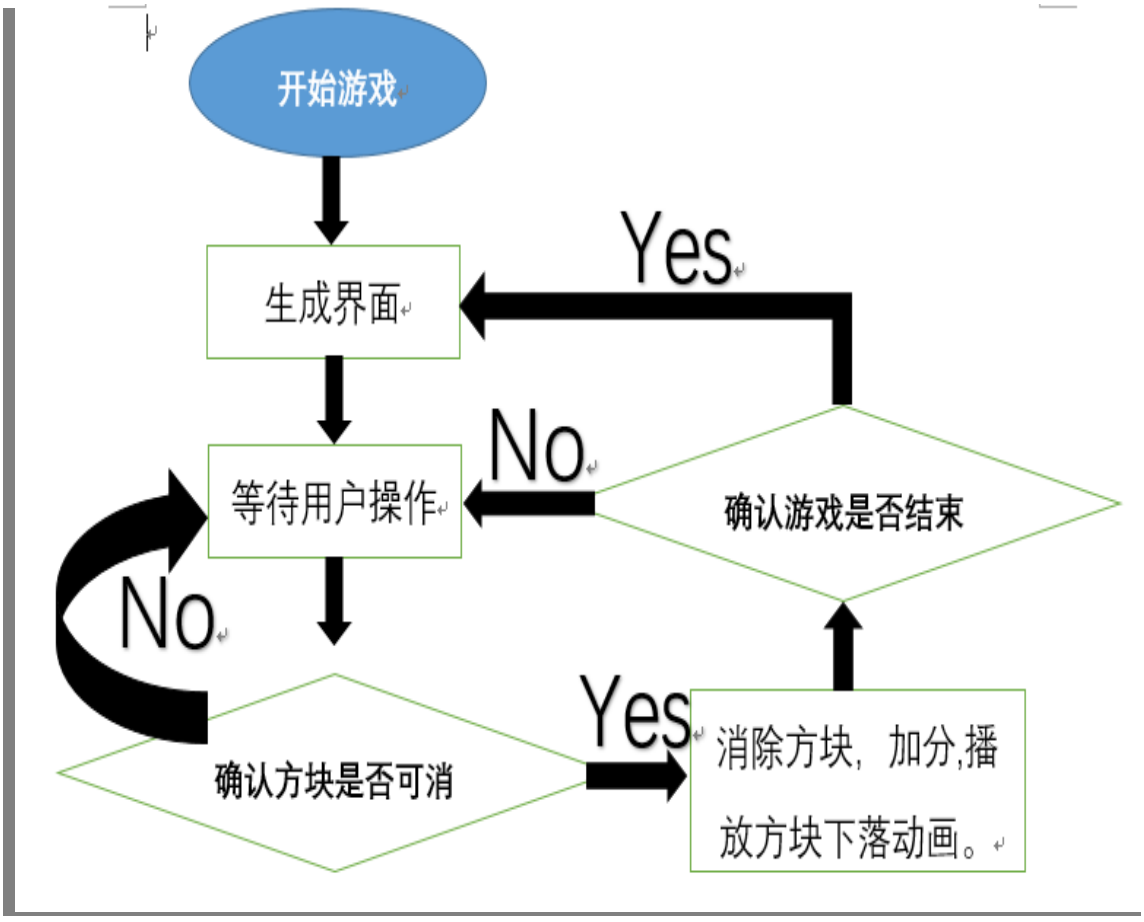
点击鼠标左键让同色的方块消除，消除越多得分越多。

游戏规则：

鼠标点击消除相连同色方块，直到剩下的方块都没有相邻同色方块，最终根据得分进行统计排名，每次游戏结束显示本次得分和历史最高得分。

2.游戏设计原理

1.游戏流程分析：



2.设计需求分析：

需求 1：GUI 界面和动画——如何使游戏更具美观性？

解决思路：使用**计时器 Timer 类**，控制每 0.1 秒形成一帧画面，利用视觉暂留形成动画效果。使用图形用户界面也增加了美观性。由此精美的游戏画面便可酝酿而成。

需求 2：交互——用户操作是否简单便捷？

解决思路：使用监听器 `MouseListener` 实现程序与用户更便捷的交互，直接获取用户鼠标操作，比键盘监听更具有实用性。

需求 3：可拓展性——多人合作编程时，能否互相看懂代码，方便他人修改升级程序？

解决思路：添加代码注释，帮助其他人看懂自己写的代码段，使用 line,col,red,blue…等通俗易懂的变量名而不是使用 magic number，并且使用宏定义，使代码中一处修改全局更正，而不用到处寻找数字，加快了 Debug 效率。这些措施极大方便了他人修改拓展自己的程序。

3.核心源代码说明

1.精简的数据结构

```
final int red=1,orange=2,green=3,blue=4,non=0;
final int line=10,col=10;//定义行列数，一次修正全局更改
 int[][][] map = new int[col][line][2];
```

用一个三维数组记录一个地图，前两维度描述坐标，最后一个维度描述属性：[][][0]内存储了地图的颜色属性，特别地，[][][1]用于本程序中的一个重要功能——如何确定一个方块在搜索时不会被重复计算，甚至产生死循环？由此引出这个用作 FLAG 的标记属性。

2.封装重复代码

```
public void swap(int x1,int y1,int x2,int y2){
   int [][][]temp=new int[1][1][2];
   temp[0][0]=map[x1][y1];
   map[x1][y1]=map[x2][y2];
   map[x2][y2]=temp[0][0];
   }
```

以交换数据的方法为例，本程序十分注重将常用的交换数据的代码封装到同一个函数中，极大提高代码的简洁性，也使得短短 167 行的代码实现了如此复杂而精巧的功能。

3.递归搜索方块

```
private int findsameCube(int  x,int y){//递归搜索相同色块，主要算法
    int count=1;
    map[x][y][1]=1;
```

```
        if(x>0&&map[x][y][0]==map[x-1][y][0]&&map[x-1][y][1]==0)
            count+=findsameCube(x-1, y);
        if(y>0&&map[x][y][0]==map[x][y-1][0]&&map[x][y-1][1]==0)
            count+=findsameCube(x, y-1);
        if(x<col-1&&map[x][y][0]==map[x+1][y][0]&&map[x+1][y][1]==0)
            count+=findsameCube(x+1, y);
        if(y<line-1&&map[x][y][0]==map[x][y+1][0]&&map[x][y+1][1]==0)
            count+=findsameCube(x, y+1);

        return count;
    }
```

递归搜索相同色块，巧妙使用标记避免重复计数，并且注意到了越界处理，递归的代码比迭代更加精简巧妙，是本课程需要掌握的重要思维。

4.主要变量说明

```
    final Timer timer = new Timer(100, new TimerListener());
    //计时器，每100ms运行一次模块class TimerListener implements
ActionListener模块


    private int score = 0;//分数

    final int red=1,orange=2,green=3,blue=4,non=0;//用数字指代颜
色


    final int line=10,col=10;//定义行列数

     int[][][] map = new int[col][line][2];//地图存储在一个
col*line*2 大小的三维数组中。
```

综上，本程序所使用的变量及其少而精妙。

5.流程控制函数

```
    private void deletCube(int  x,int y){//实际上这个函数是游戏的主流程
        int count=findsameCube(x,y);//寻找同色方块
        if(count>1){//是否可消除
            for(int i=0;i<col;i++)
                for(int j=0;j<line;j++)
                    if(map[i][j][1]==1)map[i][j][0]=non;
            score+=Math.pow(2,count);
        }else{
            map[x][y][1]=0;
```

```
        }

        repaint();
        timer.start();//开始动画
        checkover();//检查是否结束游戏
    }
```

当监听器检测到用户点击了有效区域时，调用该函数，该函数转接调用判断有无同色方块、是否游戏结、束播放动画的几个其他函数。

6.监听鼠标点击

```
• public void mouseClicked(MouseEvent  e) {
        if(e.getButton()==MouseEvent.BUTTON1){//点击鼠标左键
            int x=e.getX();
            int y=e.getY();

    if(x>20&&x<20*col+20&&y>40&&y<line*20+40)deletCube((x-20)/20,(y-40)/20)
;
        }
    }
```

监听到用户的鼠标操作时，调用 deletCube（）函数进入游戏主流程

7.每一帧画面形成

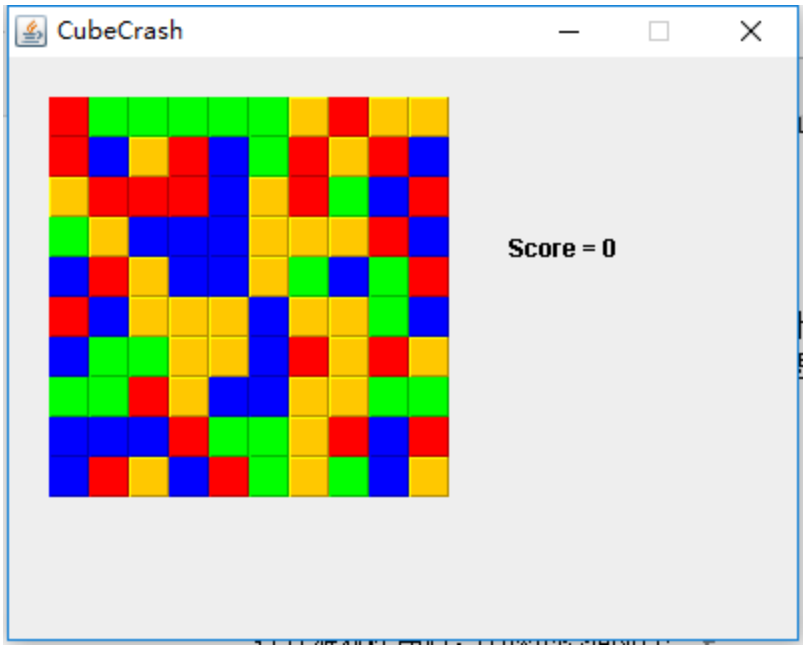
```
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        for(int i=0;i<col;i++)
            for(int j=0;j<line;j++){
                boolean flag=true;
                switch(map[i][j][0]){
                case green:g.setColor(Color.GREEN);break;
                case orange:g.setColor(Color.ORANGE);break;
                case red:g.setColor(Color.RED);break;
                case blue:g.setColor(Color.BLUE);break;
                default:flag=false;break;
                }
                if(flag)g.fill3DRect(i * 20+20, j * 20+20, 20, 20, true);
            }
```

```
        g.setColor(Color.BLACK);
        g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 12));
        g.drawString("Score = " + score, 250, 100);
    }
```

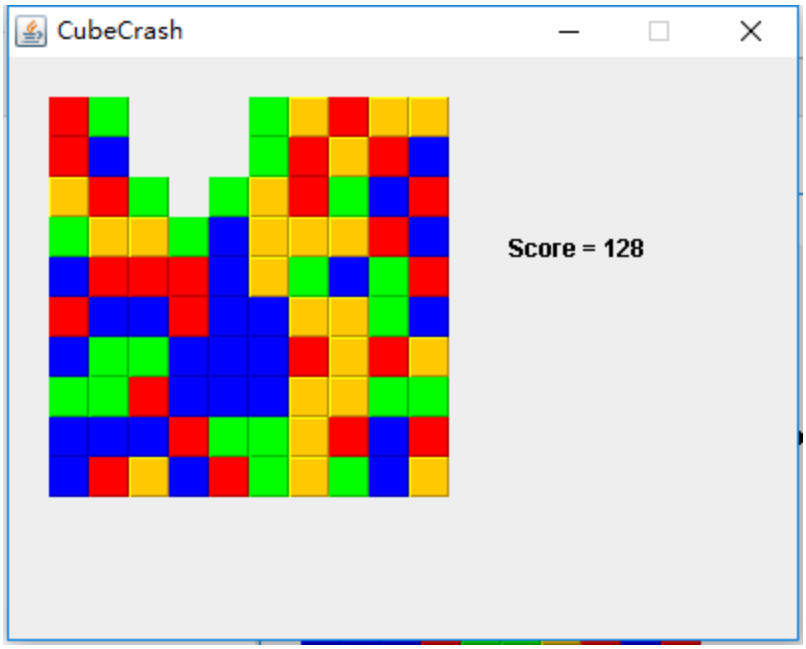
依靠该函数形成一帧一帧的画面，并且每 0.1 秒调用一次 repaint()更新画面，依靠视觉暂留效应，形成了用户可见的动画

4.游戏操作说明

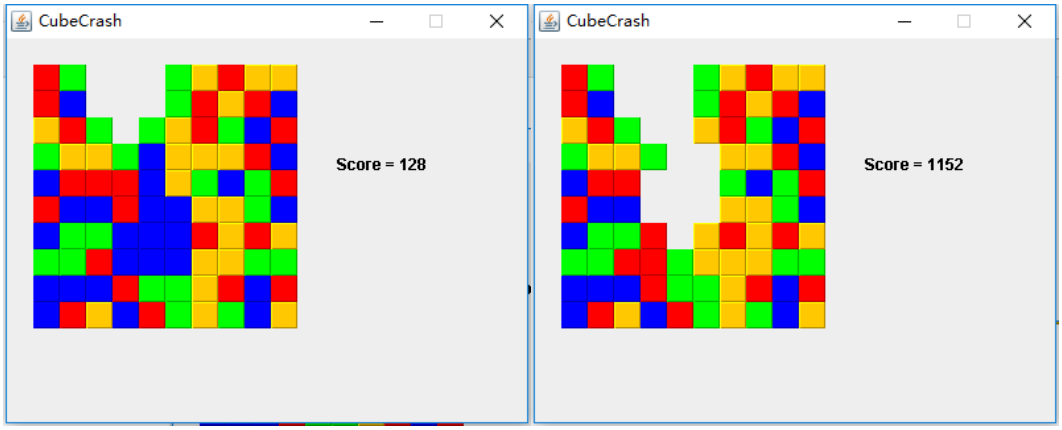
打开游戏程序后，出现的界面如下：

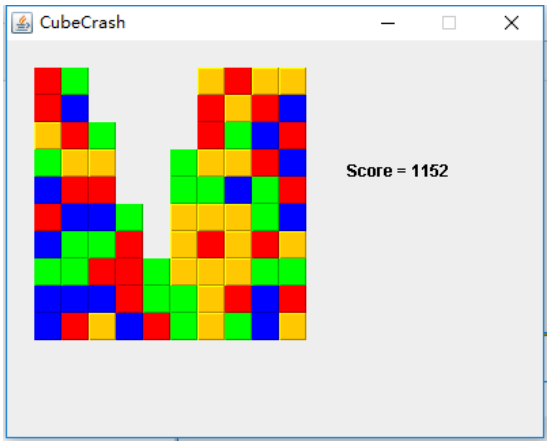


图中彩色部分即为游戏区域，玩家点击消除相连的同色方块，并且以为底数，一次消除同色方块数为指数，进行计分。

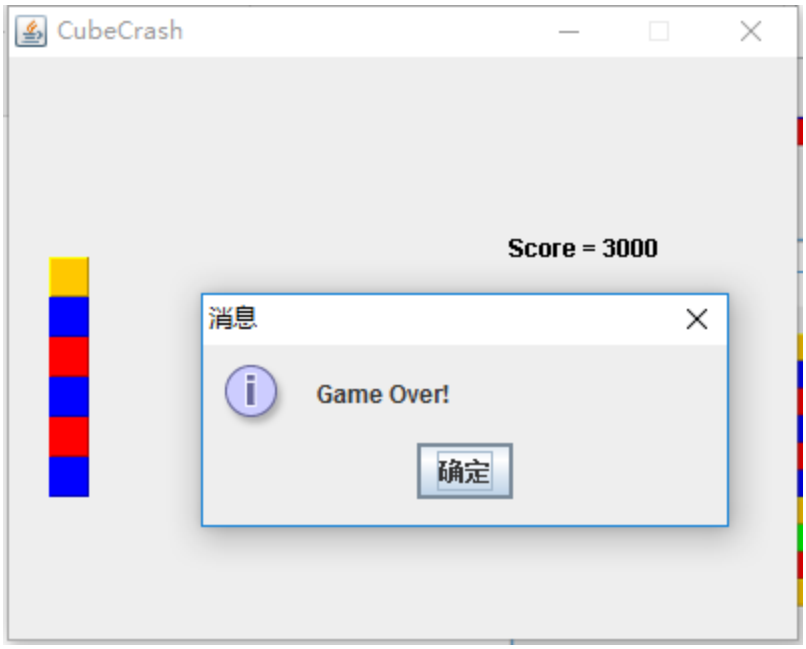


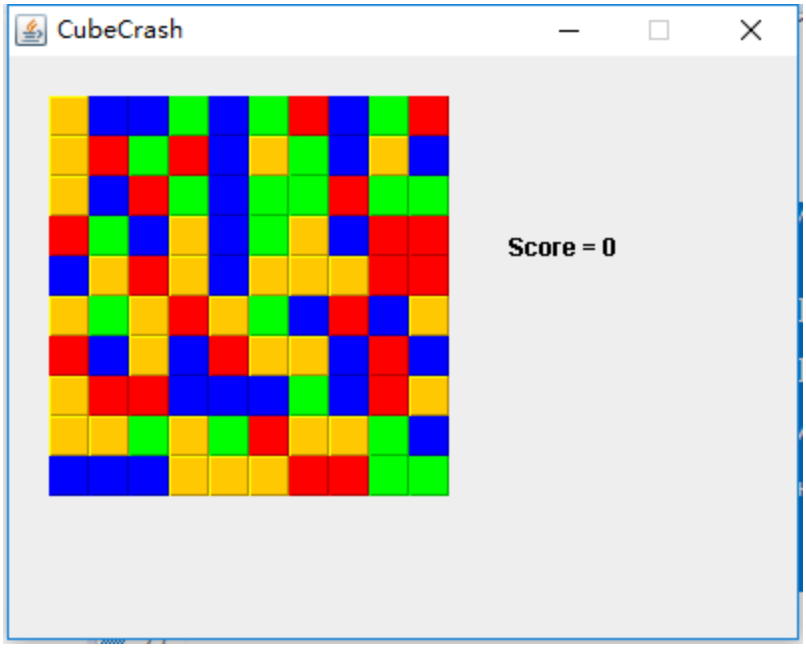
消除过程中，可见动画效果如下：





游戏结束时，弹出 GAME OVER 框，并重启游戏，如图所示：





5.总结与感悟

1. 学会了充分分析需求和流程再开始写代码

2. 学会了统筹大局巧妙使用宏定义

3. 更深一步理解了 GUI 对人机交互 的意义

4. 学会了自顶向下分部解决问题的思路和编程方法