海南大学计算机科学与技术学院

《面向对象程序设计》课程设计报告

班　　级： 计科 2022-4 班

成　　员： 徐亮（学号 20223003485）

指导老师： 张春元

完成日期： 2024 年 6 月 26 日

俄罗斯方块游戏设计与实现

摘要：本设计主要使用 Java 语言来编写一个俄罗斯方块小游戏，运用了图形化界面编程来实现游戏的运行和一系列操作，使用了 Java Swing 来实现图形化界面，其中本设计的面板分为两个部分，游戏主体和菜单，游戏主体是使用了文本域并进行网格布局来实现方块的绘制；菜单栏有一些选项，比如开始游戏、暂停、分数统计、状态和游戏说明文本等内容。

关键词：面向对象技术；Java；课程设计；Java Swing；

1、引言

本设计主要实现一个俄罗斯小游戏的开发，俄罗斯方块（Tetris）作为一款经典的益智游戏，自 1984 年诞生以来，便凭借其简单却极具挑战性的玩法风靡全球。通过设计和实现俄罗斯方块游戏，我们不仅能够深入理解基本的游戏开发流程，还能提高我们的编程技能、算法设计能力和界面交互设计能力。这对我们未来从事软件开发和计算机科学相关工作具有重要意义。

在本次课程设计中，我们选择了开发一款俄罗斯方块游戏。之所以选择这一题目，主要基于以下两点考虑：

经典性与广泛性：俄罗斯方块作为一款经典游戏，几乎所有人都玩过或听说过，它的规则简单，却需要玩家具备良好的空间想象力和快速反应能力，非常适合作为学习和练习编程的案例。

技术挑战：开发俄罗斯方块游戏，需要我们综合运用多种编程技术，包括但不限于图形绘制、键盘事件处理、碰撞检测和游戏逻辑实现，这对于全面提升我们的编程能力非常有帮助。

2、Java 程序设计语言综述

面向对象技术（Object-Oriented Technology）是一种软件开发方法论，它将软件系统看作由多个相互作用的对象组成的集合。每个对象都有自己的状态（即属性）和行为（即方法），对象之间通过消息传递进行通信和协作。面向对象技术强调模块化、可重用性、灵活性和扩展性，是现代软件开发中广泛应用的一种范式。

Java 语言是一种广泛应用于各种平台的高级编程语言，具有以下特点和优势：

（1）面向对象：Java 是一种纯粹的面向对象编程语言，支持类和对象的概念，提供了封装、继承和多态等面向对象特性。

（2）跨平台性：Java 程序可以在不同的操作系统上运行，这得益于 Java 虚拟机（JVM）的存在，它负责将 Java 字节码翻译成特定平台的机器码执行。

（3）健壮性：Java 通过强类型检查、异常处理、垃圾回收等机制提高了程序的健壮性，减少了因内存管理和指针操作而引发的错误。

（4）安全性：Java 具有安全性的特点，例如提供了安全管理器（Security Manager）来控制资源访问，防止恶意代码的运行。

（5）多线程支持：Java 提供了多线程编程的良好支持，通过内置的线程操作和同步机制，使得开发并发程序变得更加容易。

Java 的运行机制如下：

（1）编写程序：使用 Java 语言编写源代码文件，文件扩展名为.java。

（2）编译：通过 Java 编译器（javac）将源代码编译成字节码文件（.class 文件），字节码是一种与平台无关的中间代码。

（3）解释与运行：Java 虚拟机（JVM）负责将字节码文件加载到内存中，并解释执行或者即时编译为特定平台的机器码执行。这一过程确保了 Java 程序的跨平台性。

Java 语言作为一种成熟和广泛应用的编程语言，在当前的软件开发市场上仍然占据重要地位，Java 在企业级应用开发中广泛应用，如大型系统、金融软件、电子商务平台等；在移动开发中尽管 Android 平台的主要开发语言是 Kotlin，但 Java 仍然是 Android 开发的重要基础，许多旧版应用仍在使用 Java；Java 在大数据处理、分布式系统和云计算平台中有广泛的应用，如 Hadoop、Spark 等；Java 语言和相关框架（如 Spring）在 Web 应用开发中依然非常流行，提供了稳定性和性能上的保障。

总体而言，Java 语言凭借其跨平台性、健壮性和广泛的应用领域，仍然是许多开发者和企业的首选之一。随着技术的不断演进和 JVM 的优化，Java 在未来仍有持续发展的潜力。

3、系统分析与模块化设计

本设计主要设计了两个模块，一个模块是游戏界面窗体的设计，本页面主要添加了两个面板，其中一个面板是游戏区域，用来显示游戏画面，采用了网格布局的文本域来实现，其中方块的绘制使用文本域的背景颜色来实现。此模块的文件名为：BlocksGameView.java。

另一个模块进行游戏逻辑的编写，在此模块中主要实现了方块生成、方块下落、碰撞检测、方块旋转、分数统计以及各种按钮的监听事件编写，此模块用于实现游戏的基本逻辑。此模块文件名为：GameProcess.java。

图 3.1 系统模块图

4、详细设计与实现

本部分主要介绍我们对俄罗斯方块游戏的详细设计与实现。其中，4.1 节主要介绍了游戏界面模块详细设计与实现；4.2 节主要介绍了游戏逻辑模块详细设计与实现。

4.1、游戏界面模块详细设计与实现

使用 Java Swing 来实现图形化界面，创建了一个 BlocksGameView 类，继承了 JFrame，添加了 KeyListener 接口，在类中声明的函数如下：

初始化窗体的方法： public void initWindow()

初始化游戏界面的方法：public void initGamePanel()

给游戏添加焦点的方法：public void setFocus()

添加窗口组件方法：public void addComponent()

默认构造方法： public BlocksGameView()

BlocksGameView 的 UML 图如下：

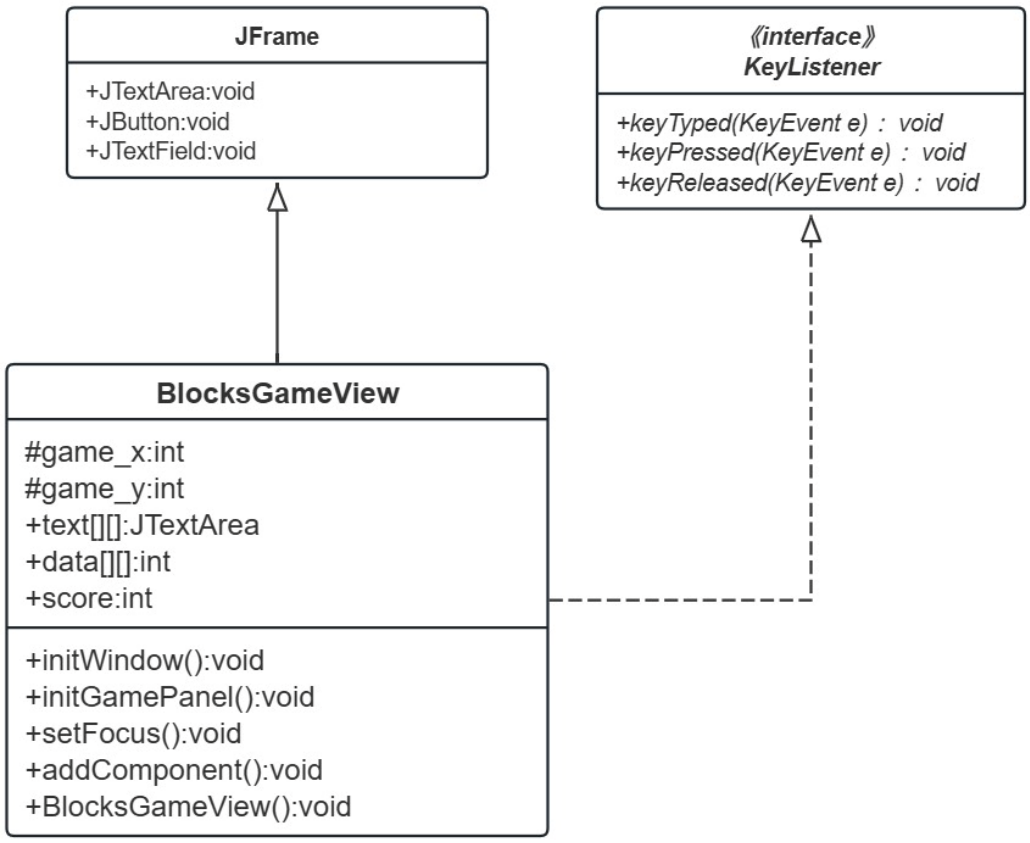

图 4-1 BlocksGameView 的 UML 图

实现的主界面运行结果如下图所示：

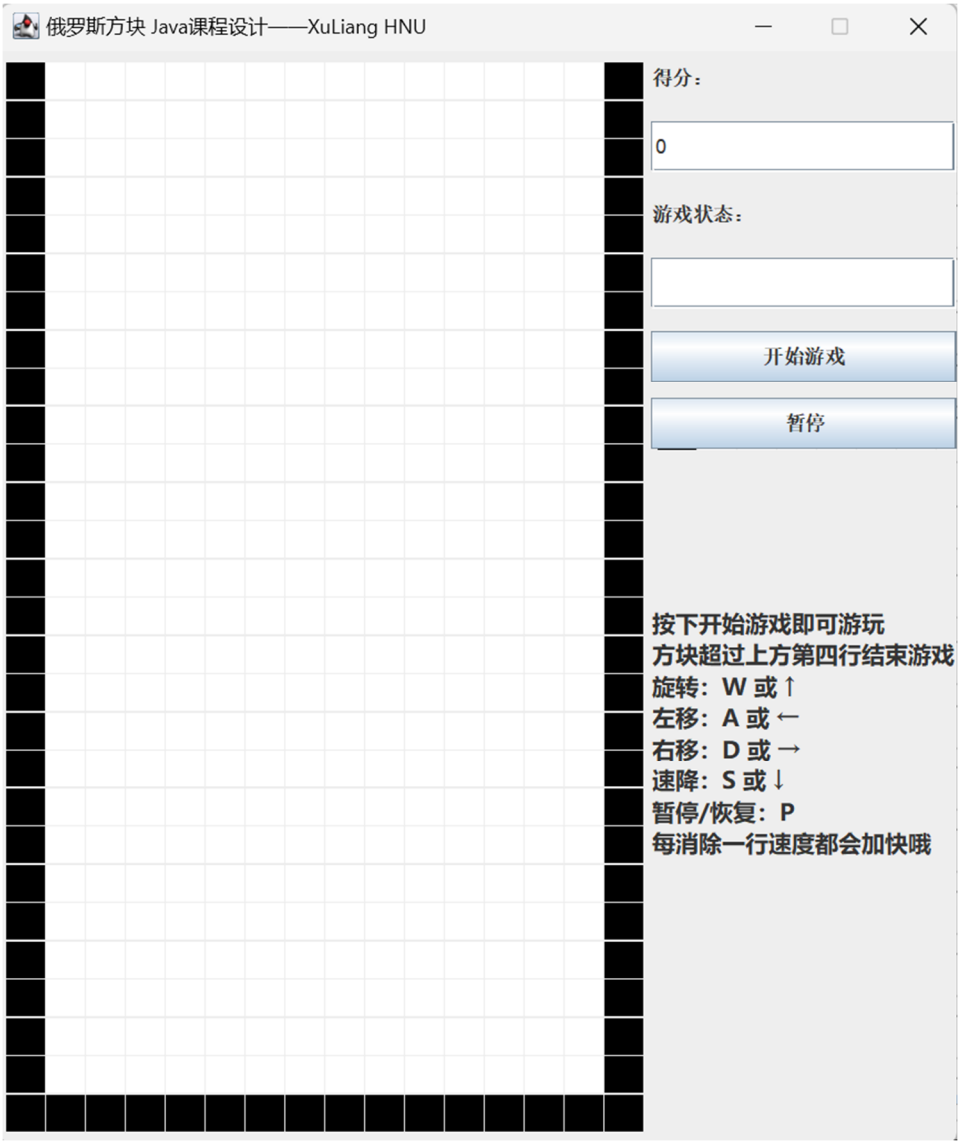

图 4-2 主界面图

其中，左边为游戏面板，右边为菜单面板，整个窗口采用 BorderLayout 布局，游戏面板和菜单面板分别位于“WEST”和“EAST”。

源代码如下：

```
import java.awt.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.*;
```

```
public class BlocksGameView extends JFrame implements
KeyListener {
    protected static final int game_x=28;
    protected static final int game_y=16;//游戏界面的行、列数
    public JTextArea text[][];//设置文本域
    int[][] data;//数据数组 1 代表有方块
    static JButton btn[] = new JButton[2];//按钮
    static String[] btnLabel= new String[2];//按钮标签
    int score = 0;//得分
    JTextField tex1 = new
JTextField(String.valueOf(score),20);//游戏得分
    JTextField tex2 = new JTextField(20);//游戏状态


    //初始化窗体的方法
    public void initWindow(){
        this.setSize(590,695);//宽 590 高 695 的窗口
        this.setVisible(true);//窗口可见
        this.setLocationRelativeTo(null);//窗口居中
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);/
/释放窗体
        this.setResizable(false);//窗口不可变
        this.setTitle("俄罗斯方块 Java 课程设计——XuLiang HNU");
    }

    //初始化游戏界面的方法
    public void initGamePanel(){
        JPanel game_main = new JPanel();
        game_main.setLayout(new
GridLayout(game_x,game_y,1,1));
        for(int i = 0 ; i<text.length; i++){
            for(int j =0 ; j<text[i].length;j++){
                //设置文本域行列
                text[i][j] = new JTextArea(game_x,game_y);
                //设置背景颜色
                text[i][j].setBackground(Color.WHITE);
                //添加键盘监听事件
                text[i][j].addKeyListener(this);
                //初始化游戏边界
                if(j==0 || j==text[i].length-1 ||
i==text.length-1){
```

```
                    text[i][j].setBackground(Color.BLACK);
                    data[i][j]=1;
                }
                //设置文本域不可编辑
                text[i][j].setEditable(false);
                //文本域添加到主面板
                game_main.add(text[i][j]);
            }
        }

        this.setLayout(new BorderLayout());
        this.add(game_main,BorderLayout.CENTER);
    }

    //给游戏添加焦点，解决点击按钮导致无法监听游戏的问题
    public void setFocus(){
        for(int i = 0 ; i<text.length; i++){
            for(int j =0 ; j<text[i].length;j++){
                text[i][j].requestFocus();
            }
        }
    }

    //添加窗口组件
    public void addComponent(){
        JPanel menus = new JPanel();//新建菜单页面，在主窗口的
右边包含文本框 按钮 说明等板块
        menus.setLayout(new GridLayout(4,1,1,10));//网格型 4
行 一列

        //以下为按钮界面
        JPanel buttonsPanel = new JPanel();
        btnLabel = new String[]{"开始游戏","暂停"};
        buttonsPanel.setLayout(new GridLayout(4,1,1,10));
        for(int i=0;i<2;i++){
            btn[i] = new JButton(btnLabel[i]);
            buttonsPanel.add(btn[i]);
        }
        //以下为文本框界面
        JPanel textField = new JPanel();
        textField.setLayout(new GridLayout(4,2,1,10));
        JLabel la1 = new JLabel("得分：");
```

```
        JLabel la2 = new JLabel("游戏状态：");
        textField.add(la1);
        textField.add(tex1);
        textField.add(la2);
        textField.add(tex2);
        //以下为说明
        JTextArea introduces = new JTextArea("按下开始游戏即可游玩\n 方块超过上方第四行结束游戏\n 旋转：W 或 ↑\n 左移：A 或 ←\n 右移：D 或 →\n 速降：S 或 ↓  \n 暂停/恢复：P\n 每消除一行速度都会加快哦",9,10);
        introduces.setBackground(null);
        introduces.setEditable(false);
        introduces.setFont(new Font("微软雅黑", Font.BOLD, 14));


        menus.add(textField);
        menus.add(buttonsPanel);
        menus.add(introduces);
        this.add(menus,BorderLayout.EAST);

    }

    public BlocksGameView(){
        text = new JTextArea[game_x][game_y];
        data = new int[game_x][game_y];
        initGamePanel();
        addComponent();
        initWindow();
    }


    @Override
    public void keyTyped(KeyEvent e) {

    }
    @Override
    public void keyPressed(KeyEvent e) {

    }

    @Override
```

```
    public void keyReleased(KeyEvent e) {

    }
}
```

4.2、 游戏逻辑模块详细设计与实现

一、方块的表达

创建了一个 GameProcess 类继承上面的 BlocksGameView 类，并在其中编写了一些涉及到游戏运行的函数，其中最为核心是方块的设计和旋转功能的实现。

首先，本设计的方块是使用网格布局的文本域通过填充背景颜色来实现的，每一个小文本域可以有数据，0 代表没有方块，1 代表有方块；在此基础上，我们可以把所有方块都放在一个 4*4 的区域内并且使用数据来标注。这样，在这个 4*4 的网格里面我们就能得到一个数表，整个数表可以表示成一个四位的十六进制数，比如一个 2*2 的方块，可以画成下面这样：

0	0	0	0
0	0	0	0
1	1	0	0
1	1	0	0

那么，其按行读取的数据就可以表示为 00CC，在 Java 中写作 0x00CC，类似的我们可以定义所有的方块，其中也画出同一形状的不同方向的方块，在实现方块旋转的时候就可以直接切换；这样可以实现将所有的方块都表达成一个十六进制数，然后存到整型数组即可，这样的好处是节省了很大的内存空间，每一个方块只需占用一个 int 型的空间。

本设计给出的所有方块共 22 种，其中旋转也只需在其中进行切换即可实现。定义的方块数组如下：

```
allBlocks = new int[]{0x00cc, 0x8888, 0x000f, 0x888f, 0xf888,
0xf111,0x111f, 0x0eee, 0xffff, 0x0008, 0x0888, 0x000e,
0x0088,0x000c, 0x08c8, 0x00e4, 0x04c4, 0x004e, 0x08c4,
0x006c, 0x04c8, 0x00c6};
```

方块可以分为以下几类（**其中数据前面的编号对应着数组的下标**）

方形	4*1 形	3*1 形	2*1 形	L 形	T 形	阶梯形
0. 0x00cc: 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 7. 0x0eee 0 0 0 0 1 1 1 0 1 1 1 0 1 1 1 0 8. 0xffff 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 9. 0x0008 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	1. 0x8888 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 2. 0x000f 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1	10. 0x0888 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 11. 0x000e 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0	12. 0x0088 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 13. 0x000c 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0	3. 0x888f 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 4. 0xf888 1 1 1 1 1 0 0 0 1 0 0 0 1 0 0 0 5. 0xf111 1 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 6. 0x111f 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1	14. 0x08c8 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 15. 0x00e4 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 16. 0x04c4 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 17. 0x004e 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0	18. 0x08c4 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 19. 0x006c 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 20. 0x04c8 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 21. 0x00c6 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0

二、方块的旋转

在上面的表格中，方形无需旋转，其余六列的旋转可以在同一个类型之间进行切换。旋转就可以通过简单的判断语句来实现，例如其中一个类型的旋转代码可以写成下面这样：

```
if (old >= 14 && old <= 17) {
            next = allBlocks[old + 1 > 17 ? 14 : old + 1];

            if (canTurn(next, x, y)) {
                block = next;
            }
        }

//*说明：old 表示旋转前的方块对应的下标，此处是为了实现 T 形方块的旋转，next 表示旋转后的方块对应的下标，代码给出的语句可以实现在 14~17 之间的循环切换从而实现旋转功能，canTurn()是用来判断能否旋转的方法。*/
```

三、方块的检测算法

在本设计中，方块实际是由数据来表达的，那么如何实现方块的检测就是一个重要的问题，这个问题涉及到方块的绘制、下落、碰撞、消除等一系列功能的实现。本设计给出了一种通用算法，形式如下：

```
int temp = 0x8000;
        for(int i = 0;i<4;i++){
            for(int j = 0;j<4;j++){
                if((temp&block)!=0){
                    if(data[m+1][n]==1){
                        return false;
                    }
                }
                n++;
                temp>>=1;
            }
            m++;
            n=n-4;
        }
        return true;
//*说明：temp 也是一个十六进制数 0X8000，其代表一个仅在第一格数据为 1 的 4*4 数表，使用一个二重循环，可以实现对一个 4*4 区域的检测，其过程大概如下：首先从第一行第一列开始，temp&block 表示按位与运算，若有方块结果则是 1&1=1 表示有方块，此时需要返回为假，说明此 4*4 区域内有方块。若没有方块则将 temp 右移一位，此时再进行按位与运算就表示对第二列的数据进行检测，遍历完一行后，m++进入下一行，n=n-4 再从第一列开始遍历，若全部循环结束都没有方块出现，返回为真，说明此 4*4 区域内没有方块
*/
```

GameProcess 类继承上面的 BlocksGameView 类，其中编写了一些涉及到游戏运行的函数。

游戏开始方法：public void gameBegin()

随机生成方块方法：public void ranBlocks()

游戏运行方法：public void gameRun()

判断能否下落方法：public Boolean canFall(int,int)

改变不可下落方块的值的方法：public void changeData(int,int)

消除一行方法：public void clearRow(int)

刷新移除后游戏界面的方法：public void reflash(int)

方块下落方法：public void fall(int,int)

重新绘制方块方法：public void reBuild(int,int)

判断能否旋转方法：public void canTurn(int,int,int)

默认构造方法：public GameProcess()

GameProcess 类的 UML 图如下：

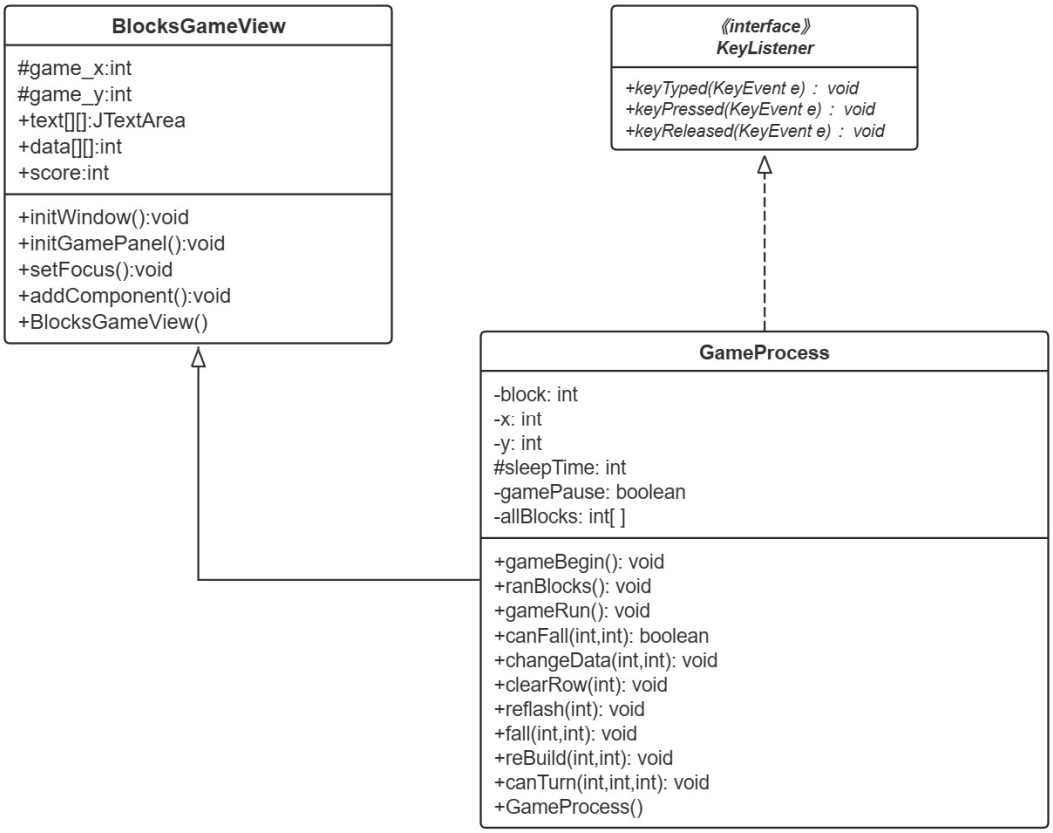

图 4-2 GameProcess 的 UML 图

源代码如下：

```
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Random;
import javax.swing.*;
public class GameProcess extends BlocksGameView implements
KeyListener{
    static int startSymbol;//游戏状态标识---1 代表开始
```

```
    private int block;//当前方块（二进制数组）
    private int x,y;//当前方块的坐标
    protected int sleepTime = 1000;//线程休眠时间
    boolean gamePause= false;//是否暂停标识符
    static int pauseClickCount = 0;
    private int[] allBlocks = new int[22];
    //游戏开始方法
    public void gameBegin(){
        while(true){
            if(startSymbol==0){
                break;
            }
            gameRun();
        }
        tex2.setText("游戏结束！");
        JOptionPane.showMessageDialog(null, "GameOver");//弹
出提示框
    }
    //生成随机方块方法
    public void ranBlocks(){
        Random random = new Random();
        block=allBlocks[random.nextInt(22)];
    }
    //游戏运行方法
    public void gameRun(){
        ranBlocks();//生成随机方块
        setFocus();
        x=0;
        y=6;
        for(int i=0;i<game_x;i++){
            try {
                Thread.sleep(sleepTime);
                if(gamePause){
                    i--;
                }else{
                if(!canFall(x,y)){
                    changeData(x,y);//data 置为 1
                    //判断是否可以消除
                    for(int j =x; j<x+4;j++){
                        int sum=0;
                        for(int k =1 ; k<=(game_y-2);k++){
                            if(data[j][k]==1){
```

```
                                sum++;
                            }
                        }
                        if(sum==(game_y-2)){
                            clearRow(j);
                        }
                    }
                    //判断游戏是否失败
                    for(int j =1; j<=(game_y-2);j++){
                        if(data[3][j]==1){
                            startSymbol=0;
                            break;
                        }
                    }
                    break;
                }else{
                    x++;
                    fall(x,y);
                }
              }
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    //判断能不能下降
    public boolean canFall(int m,int n){
        int temp = 0x8000;
        for(int i = 0;i<4;i++){
            for(int j = 0;j<4;j++){
                if((temp&block)!=0){
                    if(data[m+1][n]==1){
                        return false;
                    }
                }
                n++;
                temp>>=1;
            }
            m++;
            n=n-4;
```

```
        }
        return true;
    }

    //改变不可下降方块对应的值的方法
    public void changeData(int m,int n){
        int temp = 0x8000;
        for(int i=0;i<4;i++){
            for(int j=0;j<4;j++){
                if((temp&block)!=0){
                    data[m][n] = 1;
                }
                n++;
                temp>>=1;
            }
            m++;
            n=n-4;
        }
    }
    public void clearRow(int row){
        int temp =200;
        for(int i = row; i>=1; i--){
            for(int j=1;j<=(game_y-2);j++){
                data[i][j]=data[i-1][j];
            }
        }
        reflash(row);

        if(sleepTime > temp){
            sleepTime -= 100;
        }//每消除一行提高下落速度

        score += 10;
        tex1.setText(String.valueOf(score));
    }

    //刷新移除某一行后游戏界面的方法
     public void reflash(int row){
        for(int i =row;i>=1;i--){
            for(int j=1;j<=(game_y-2);j++){
                if(data[i][j]==1){
                    text[i][j].setBackground(Color.ORANGE);
```

```
                }else{
                    text[i][j].setBackground(Color.WHITE);
                }
            }
        }
     }

    //方块下落方法
    public void fall(int m,int n){
        if(m>0){
            //清除上一层的方块颜色
            clearColor(m-1,n);
        }
        reBuild(m,n);//重新绘制方块
    }

    //方块下落后清除上一层的颜色的方法
    public void clearColor(int m,int n){
        int temp = 0x8000;
        for(int i = 0;i<4;i++){
            for(int j=0;j<4;j++){
                if((temp&block)!=0){
                    text[m][n].setBackground(Color.WHITE);
                }
                n++;
                temp>>=1;
            }
            m++;
            n=n-4;
        }
    }

    //重新绘制方块的方法
    public void reBuild(int m,int n){
        int temp = 0x8000;
        for(int i=0;i<4;i++){
            for(int j=0;j<4;j++){
                if((temp&block)!=0){
                    text[m][n].setBackground(Color.ORANGE);
                }
                n++;
                temp>>=1;
```

```
            }
            m++;
            n=n-4;
        }
    }

     //判断方块此时是否可以变形的方法
     public boolean canTurn(int a, int m, int n) {
        //创建变量
        int temp = 0x8000;
        //遍历整个方块
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if ((a & temp) != 0) {
                    if (data[m][n] == 1) {
                        return false;
                    }
                }
                n++;
                temp >>= 1;
            }
            m++;
            n = n - 4;
        }
        //可以变形
        return true;
    }

    public GameProcess(){
        //初始化方块
        allBlocks = new int[]{0x00cc, 0x8888, 0x000f, 0x888f,
0xf888, 0xf111,0x111f, 0x0eee, 0xffff, 0x0008, 0x0888,
0x000e, 0x0088,0x000c, 0x08c8, 0x00e4, 0x04c4, 0x004e,
0x08c4,0x006c, 0x04c8, 0x00c6};
        //给按钮添加监听器
        //btn[0]控制开始游戏
        btn[0].addActionListener(new ActionListener(){
           public void actionPerformed(ActionEvent e) {
              startSymbol=1;
              btn[0].setEnabled(false);//开始后不可触发
              tex2.setText("游戏中...");
              tex2.setForeground(Color.BLUE);
```

```
                tex2.setFont(new Font("微软雅黑", Font.BOLD,
14));
                /*gameBegin() 方法执行的任务量太大，可能会阻塞界面
的事件分发线程（Event Dispatch Thread）从而导致卡死。
                  所以单独分配一个线程**/
                Thread gameThread = new Thread(new Runnable() {
                  public void run() {
                      gameBegin();
                  }
              });
              // 启动线程
              gameThread.start();
            }

          });
         //btn[1]控制暂停
         btn[1].addActionListener(new ActionListener() {
             public void actionPerformed(ActionEvent e){
                 if(startSymbol==0){
                     return;
                 }
                 pauseClickCount++;
                 //按下一次，暂停
                 if(pauseClickCount==1){
                     gamePause=true;
                     tex2.setText("已暂停");
                     tex2.setForeground(Color.PINK);
                     tex2.setFont(new Font("微软雅黑",
Font.BOLD, 14));
                 }
                 //按下两次，恢复暂停
                 if(pauseClickCount==2){
                     gamePause=false;
                     pauseClickCount=0;//重新开始计数
                     tex2.setText("游戏中...");
                     tex2.setForeground(Color.BLUE);
                     tex2.setFont(new Font("微软雅黑",
Font.BOLD, 14));
                     setFocus();
                 }
             }
         });
```

```
    }

    public static void main(String[] args) {
        GameProcess gameExec = new GameProcess();
    }

    @Override
    public void keyPressed(KeyEvent e) {
        //方块进行左移
    if (KeyEvent.VK_LEFT == e.getKeyCode() || KeyEvent.VK_A
== e.getKeyCode()) {
        //判断游戏是否结束
        if (startSymbol==0) {
            return;
        }

        //判断游戏是否暂停
        if(gamePause){return;}

        //方块是否碰到左墙壁
        if (y <= 1) {
            return;
        }

        //定义一个变量
        int temp = 0x8000;

        for (int i = x; i < x + 4; i++) {
            for (int j = y; j < y + 4; j++) {
                if ((temp & block) != 0) {
                    if (data[i][j - 1] == 1) {
                        return;
                    }
                }
                temp >>= 1;
            }
        }

        //首先清除目前方块
        clearColor(x, y);
        y--;
        reBuild(x, y);
```

```
    }

        //方块进行右移
        if (KeyEvent.VK_RIGHT == e.getKeyCode() ||
KeyEvent.VK_D == e.getKeyCode()) {
          //判断游戏是否结束
          if (startSymbol==0) {
            return;
          }

        //判断游戏是否暂停
        if(gamePause){return;}

        //定义变量
        int temp = 0x8000;
        int m = x;
        int n = y;

        //存储最右边的坐标值
        int num = 1;

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if ((temp & block) != 0) {
                    if (n > num) {
                        num = n;
                    }
                }
                n++;
                temp >>= 1;
            }
            m++;
            n = n - 4;
        }

        //判断是否碰到右墙壁
        if (num >= (game_y - 2)) {
            return;
        }

        //方块右移途中是否碰到别的方块
        temp = 0x8000;
```

```
        for (int i = x; i < x + 4; i++) {
            for (int j = y; j < y + 4; j++) {
                if ((temp & block) != 0) {
                    if (data[i][j + 1] == 1) {
                        return;
                    }
                }
                temp >>= 1;
            }
        }

        //清除当前方块
        clearColor(x, y);
        y++;
        reBuild(x, y);
    }

    //方块进行下落
    if (KeyEvent.VK_DOWN == e.getKeyCode() || KeyEvent.VK_S
== e.getKeyCode()) {
        //判断游戏是否结束
        if (startSymbol==0) {
            return;
        }

        //判断游戏是否暂停
        if(gamePause){return;}

        //判断方块是否可以下落
        if (!canFall(x, y)) {
            return;
        }

        clearColor(x, y);

        //改变方块的坐标
        x++;

        reBuild(x, y);
    }
    //键盘控制游戏暂停
    if(KeyEvent.VK_P == e.getKeyCode()){
```

```
        if(startSymbol==0){
            return;
        }
        pauseClickCount++;
        //按下一次，暂停
        if(pauseClickCount==1){
            gamePause=true;
            tex2.setText("已暂停");
            tex2.setForeground(Color.PINK);
            tex2.setFont(new Font("微软雅黑", Font.BOLD, 14));
        }
        //按下两次，恢复暂停
        if(pauseClickCount==2){
            gamePause=false;
            pauseClickCount=0;//重新开始计数
            tex2.setText("游戏中...");
            tex2.setForeground(Color.RED);
            tex2.setFont(new Font("微软雅黑", Font.BOLD, 14));
        }

    }



    //控制方块进行变形
    if (KeyEvent.VK_UP == e.getKeyCode() || KeyEvent.VK_W ==
e.getKeyCode()) {
        //判断游戏是否结束
        if (startSymbol==0) {
            return;
        }

        //判断游戏是否暂停
        if(gamePause){return;}


        //定义变量,存储目前方块的索引
        int old;
        for (old = 0; old < allBlocks.length; old++) {
            //判断是否是当前方块
            if (block == allBlocks[old]) {
                break;
```

```
        }
    }

    //定义变量,存储变形后方块
    int next;

    //判断是方块
    if (old == 0 || old == 7 || old == 8 || old == 9) {
        return;
    }

    //清除当前方块
    clearColor(x, y);

    if (old == 1 || old == 2) {
        next = allBlocks[old == 1 ? 2 : 1];

        if (canTurn(next, x, y)) {
            block = next;
        }
    }

    if (old >= 3 && old <= 6) {
        next = allBlocks[old + 1 > 6 ? 3 : old + 1];

        if (canTurn(next, x, y)) {
            block = next;
        }
    }

    if (old == 10 || old == 11) {
        next = allBlocks[old == 10 ? 11 : 10];

        if (canTurn(next, x, y)) {
            block = next;
        }
    }

    if (old == 12 || old == 13) {
        next = allBlocks[old == 12 ? 13 : 12];

        if (canTurn(next, x, y)) {
```

```
                block = next;
            }
        }

        if (old >= 14 && old <= 17) {
            next = allBlocks[old + 1 > 17 ? 14 : old + 1];

            if (canTurn(next, x, y)) {
                block = next;
            }
        }

        if (old == 18 || old == 19) {
            next = allBlocks[old == 18 ? 19 : 18];

            if (canTurn(next, x, y)) {
                block = next;
            }
        }

        if (old == 20 || old == 21) {
            next = allBlocks[old == 20 ? 21 : 20];

            if (canTurn(next, x, y)) {
                block = next;
            }
        }
        //重新绘制变形后方块
        reBuild(x, y);

    }

    }
}
```

5、系统测试与运行

本设计主要设计了两个模块，一个模块是游戏界面窗体的设计，本页面主要添加了两个面板，其中一个面板是游戏区域，用来显示游戏画面，采用了网格布局的文本域来实现，其中方块的绘制使用文本域的背景颜色来实现。本设计中方块颜色为橙色。

游戏界面的右边是菜单面板，当中显示实时得分和游戏状态，在游戏未运行时游戏状态为空，开始游戏后游戏状态为“游戏中…”，字体为蓝色，暂停时游戏状态为“已暂停”，字体为粉色。游戏到达上方第四行时结束，结束时弹出窗口提示，游戏状态显示为“游戏结束!”，字体为蓝色。游戏结束后得分即为此局游戏的最终得分。

1.主界面效果图：

图 5-1 游戏主界面

2.点击开始游戏按钮即可开始游戏，游戏时会显示游戏状态并进行实时计分。

图 5-2 游戏状态

3.游戏过程中点击一次暂停即可实现游戏暂停，再点击一次即可恢复游戏。

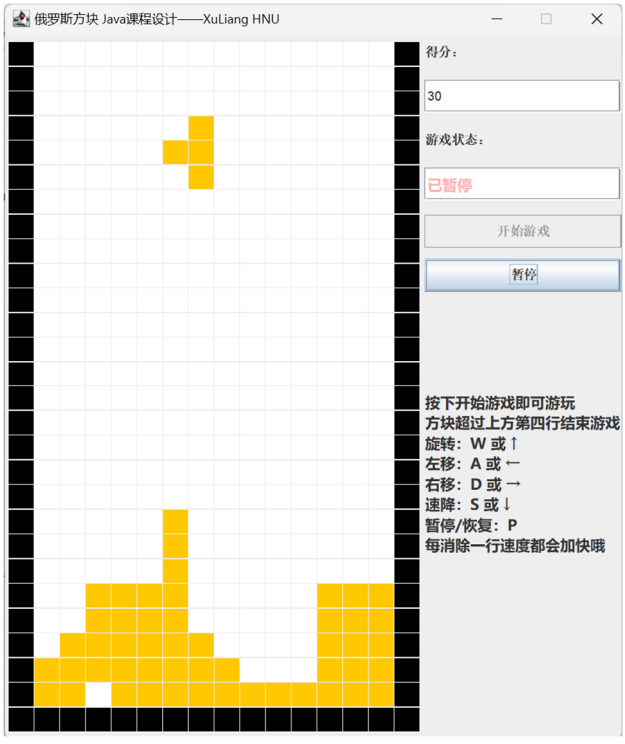

图 5-3 暂停状态

4.方块旋转

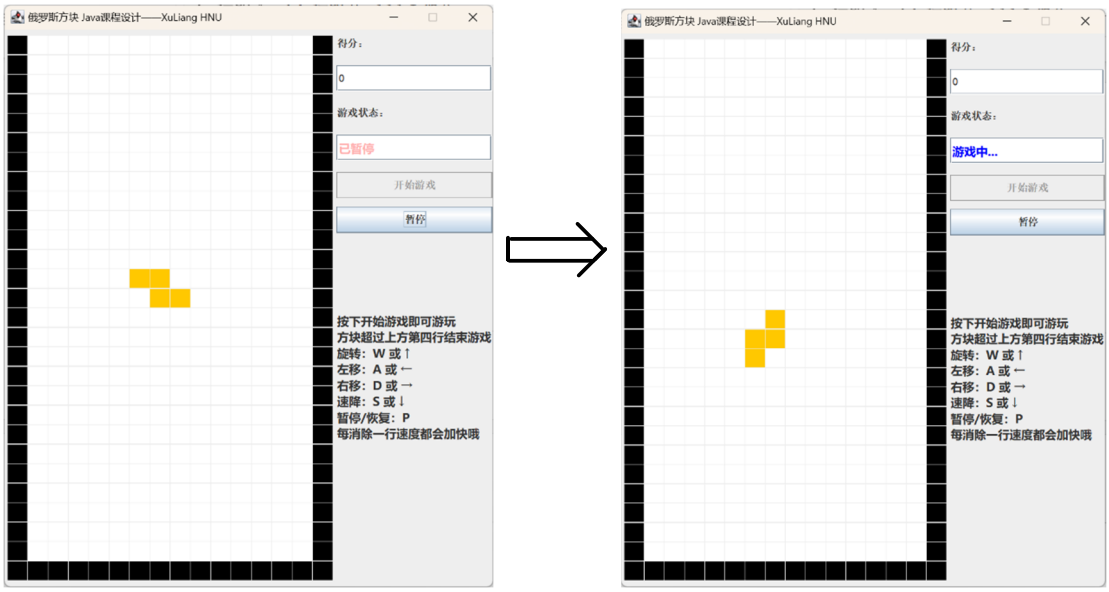

图 5-4 方块旋转

5.每消除一行加 10 分。

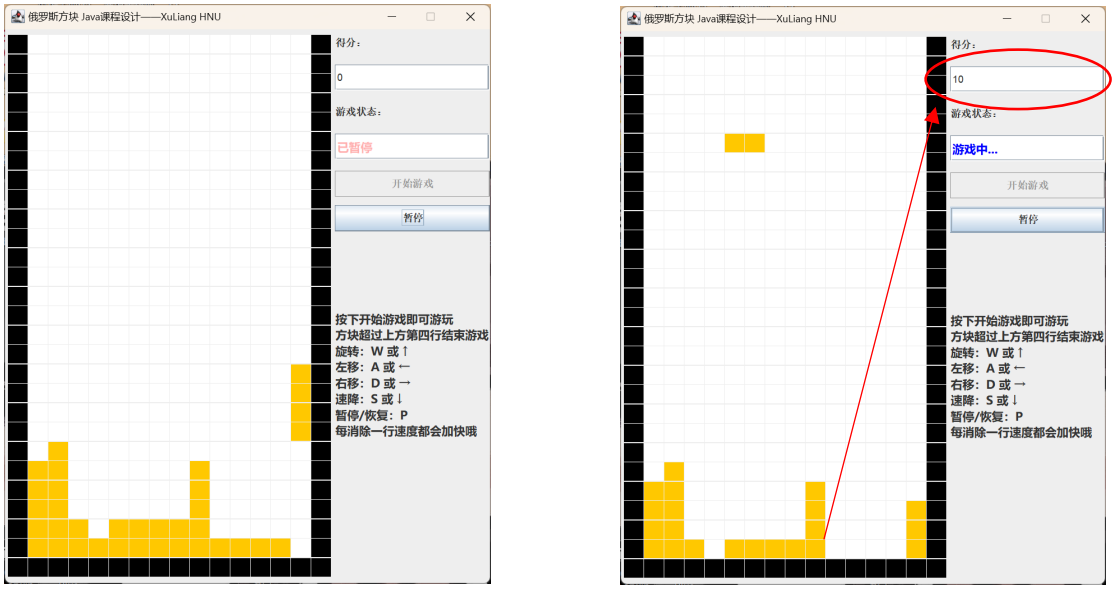

图 5-5 消除一行方块

6.方块到达最高线处游戏结束（本设计不能超过第四行）

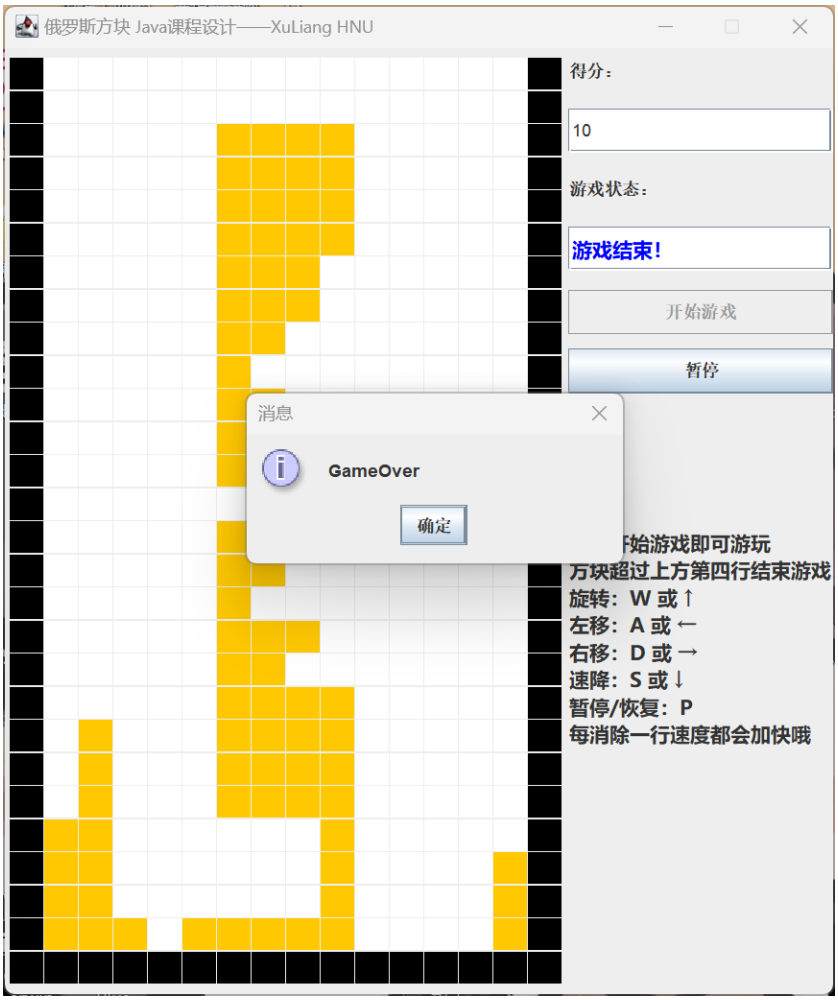

图 5-6 游戏结束界面

6、结束语

在本次课程设计中，我遇到了一些困难，但通过努力和坚持，成功地克服了它们。首先，我面临的一个挑战是对俄罗斯方块游戏的规则和算法不够了解。然而，我通过查阅相关资料和学习经典的游戏实现，逐渐掌握了游戏的核心要素，并成功地将其应用到我的设计中。

其中我认为最成功的应用是使用十六进制数来表达方块，这使得相关数据的存储变得简洁，函数的设计也仅需考虑简单的数值运算，避开了复杂的图像重绘等工作。

其次，我在设计过程中遇到了技术上的困难。由于我是初学者，对编程语言和游戏开发工具的使用并不熟悉。为了解决这个问题，我充分利用了网络资源和教程，积极向同学们请教，逐步提高了自己的技术水平。通过不断的尝试和调试，我成功地完成了游戏的基本功能，并加入了一些额外的特性，使游戏更加丰富和有趣。

在这个过程中，我获得了许多宝贵的收获。首先，我深刻理解了团队合作的重要性。虽然这次课程设计是个人完成的，但我在解决问题时积极与同学们交流，受益匪浅。其次，我提高了自己的问题解决能力和学习能力。面对困难，我能够主动寻求解决方案，并通过不断学习和实践不断改进。

然而，我也意识到了自己的一些不足之处。首先，时间管理能力还需要加强。在设计过程中，我有时会花费过多的时间在某个细节上，导致整体进度拖延。其次，对于游戏设计的美感和用户体验方面，我还有待提高。虽然我努力使游戏界面简洁明了、操作流畅，但仍然存在一些不足之处。

总的来说，通过这次课程设计，我不仅成功实现了俄罗斯方块游戏的基本功能，还提升了自己的技术能力和解决问题的能力。虽然还有一些不足之处，但我相信通过不断学习和实践，我将能够不断改进和完善自己的设计水平。这次经历将成为我学习和成长的重要里程碑，为我未来的学习和职业发展奠定了坚实的基础。

参考文献

[1]耿祥义、张跃平，《Java2 实用教程（第 6 版）》，清华大学出版社，2021.
[2]耿祥义、张跃平，《Java 2 实用教程（第 6 版）实验指导与习题解答》，清华大学出版社，2021.

《面向对象程序设计》课程设计团队成员自行评定总成绩

姓名	学号	职务	总成绩
徐亮	202230034485	队长	A