软件设计实践报告

单 位：计算机·网络空间安全学院

班 级： 计算机科学与技术三班

学 号： 201705550335

姓 名： 殷玉洁

任课教师： 谢勇

湘 潭 大 学

2020年12月

1. 项目概述
2. 概述

本软件面向所有年龄段的人，为玩家提供图形可视化界面的俄罗斯方块游戏软件。软件提供俄罗斯方块游戏的基本功能：对方块的移动、旋转、堆积、消除，游戏得分和游戏等级等一系列基本操作。玩家可以自己选择界面颜色，可以自己调节游戏难度。游戏开始后，玩家通过键盘方向键控制方块向下、向左、向右移动以及变换方向，待堆积的方块填满一行的时候即可消除该行，同时积分增加一百分，累计满六百分难度自动增加一个等级。初始默认游戏难度为2级，最高10级，可自动增减难度等级。

1. 目的和用途

本软件的创建目的在于重现经典小游戏。运用自己所学的编程知识，自己重新制作一个俄罗斯方块小游戏，一方面锻炼了自己的编程能力，另一方面也希望能向游戏的创造者致敬，感谢其为我们的童年带来了美好的记忆。由于时间和能力均有限，所以制作较为粗糙。编写本软件的目的仅在于通过自己的双手来重现经典的小游戏。

本软件是适用于休闲娱乐玩家回顾经典的一款小游戏。虽然制作较为粗糙，外观并不怎么美观，但游戏应该有的功能基本都有且操作简单便捷，所以不仅仅适合老玩家回顾、怀旧，同时也适合新玩家入门学习、练手。此外，本软件有一些勉强能称之为闪光点的地方，对于有意开发俄罗斯方块小游戏的人具有一定的参考价值。

1. 项目小组成员

小组成员：殷玉洁、周星玥

各自负责的工作：

殷玉洁：

周星玥：

自评：

通过本次软件设计实践，我学习到了如何设计图形界面，锻炼了我的java水平，同时也发现了把前后端连接起来需要多个步骤与测试，才能保证在使用时不会出错。也让我认识到，完成一个项目，靠个人的力量是不够的，要学会团结大家的力量，同时知识是不可数的，只有在不断的学习中，我们才会有进步。

1. 项目工作进度

2020年12月26日 确定项目内容

2020年12月27日 确定项目实现方法

2020年12月31日 项目基本设计完成

2021年 1 月14日 代码基本完成

2021年 1 月17日 测试基本完成

2021年 1 月23日 文档撰写完成

1. 项目代码托管地址

/\*

\*

\*/

1. 项目文档、代码提交记录

/\*

\*

\*/

1. 软件演示视频连接

/\*

\*

\*/

1. 需求分析
2. 用例模型

|  |  |
| --- | --- |
| **用例编号与名称** | UC1 处理开始请求 |
| **范围** | 游戏控制 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望点击“开始”后即可立刻进行游戏 |
| **前置条件** | 无 |
| **成功保证** | “开始”按钮处于可点击状态，swing中有提供相应接口 |
| **主成功场景** | 用户点击“开始”后，方块开始下落，此按钮禁用 |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标与屏幕设备 |
| **技术与数据变元素** | 开始指令由鼠标输入 |
| **发生频率** | 可能会间隔一段时间再次发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC2 处理终止请求 |
| **范围** | 游戏控制 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望点击“结束游戏”后即可清空当前游戏界面，同时一切游戏设置初始化 |
| **前置条件** | 游戏已经是运行状态 |
| **成功保证** | “结束游戏”按钮处于可点击状态，swing中有提供相应接口 |
| **主成功场景** | 用户点击“结束游戏”后，初始化游戏界面，清空用户等级分数等信息 |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标与屏幕设备 |
| **技术与数据变元素** | 结束指令由鼠标输入 |
| **发生频率** | 可能会间隔一段时间再次发生 |
| **杂项** | 未决问题：在结束前是否经过用户的二次确认 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC3 处理暂停请求 |
| **范围** | 游戏控制 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望点击“暂停”后，游戏画面静止 |
| **前置条件** | 暂停前游戏应处于运行状态 |
| **成功保证** | “暂停”按钮处于可点击状态，swing中有提供相应接口 |
| **主成功场景** | 用户点击“暂停”后，界面静止，“暂停”变为“继续” |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标与屏幕设备 |
| **技术与数据变元素** | 暂停继续指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 未决问题：是否考虑使用快捷键（空格）控制 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC4 处理继续请求 |
| **范围** | 游戏控制 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望恢复时，能从中断处继续开始 |
| **前置条件** | 继续前游戏应处于暂停状态 |
| **成功保证** | “继续”按钮处于可点击状态，swing中有提供相应接口 |
| **主成功场景** | 用户点击“继续”后，游戏继续，“暂停”变为“继续” |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标与屏幕设备 |
| **技术与数据变元素** | 暂停继续指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 未决问题：是否考虑使用快捷键（空格）控制 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC5 处理方块变形请求 |
| **范围** | 游戏操作 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望按一次“↑”按键方块就顺时针旋转90度 |
| **前置条件** | 游戏处于运行状态且周围无阻拦物体（边界、其他方格） |
| **成功保证** | Swing中有提供接口 |
| **主成功场景** | 用户按压“↑”键后，方块保持下降的同时顺时针旋转90度 |
| **扩展** | 方块变形中如果遇到阻拦则保持原样式不变 |
| **特殊需求** | 需要键盘与屏幕设备 |
| **技术与数据变元素** | 变形指令由键盘输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC6 处理方块左移请求 |
| **范围** | 游戏操作 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望按一次“←”按键方块就可以执行左移操作 |
| **前置条件** | 游戏处于运行状态且周围无阻拦物体（边界、其他方格） |
| **成功保证** | swing中有提供相应接口 |
| **主成功场景** | 用户按压“←”键后，方块下降的同时向左平移1格 |
| **扩展** | 方块左移中如果遇到阻拦则不再左移 |
| **特殊需求** | 需要键盘与屏幕设备 |
| **技术与数据变元素** | 指令由键盘输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC7 处理方块右移请求 |
| **范围** | 游戏操作 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望按一次“→”按键方块就可以执行右移操作 |
| **前置条件** | 游戏处于运行状态且周围无阻拦物体（边界、其他方格） |
| **成功保证** | swing中有提供相应接口 |
| **主成功场景** | 用户按压“→”键后，方块下落的同时向右平移1格 |
| **扩展** | 方块右移中如果遇到阻拦则不再右移 |
| **特殊需求** | 需要键盘与屏幕设备 |
| **技术与数据变元素** | 指令由键盘输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC8 处理方块下落请求 |
| **范围** | 游戏操作 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望按一次“↓”按键方块就可以执行下落操作 |
| **前置条件** | 游戏处于运行状态且周围无阻拦物体（边界、其他方格） |
| **成功保证** | swing中有提供相应接口 |
| **主成功场景** | 用户按压“↓”键后，方块下落的同时下移1格 |
| **扩展** | 方块下移中如果遇到阻拦则不再右下移 |
| **特殊需求** | 需要键盘与屏幕设备 |
| **技术与数据变元素** | 指令由键盘输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC9 消除方块并积分 |
| **范围** | 游戏功能 |
| **级别** | 子功能 |
| **主要参入者** | 游戏主系统 |
| **涉众及其关注点** | 用户希望每填满一行，该行就可以被消除，同时获得积分 |
| **前置条件** | 游戏处于运行状态 |
| **成功保证** | 系统记录积分信息，系统消除相应行图案 |
| **主成功场景** | 方块填满一行后，该行不再显示在屏幕上，积分栏增加100积分 |
| **扩展** | 如果有多行同时被填满，则从上到下，依次把上一行方块分布复制到填满行中，增加积分数为100\*行数 |
| **特殊需求** | 需要屏幕设备 |
| **技术与数据变元素** | 无 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 未决问题：一次消除三行以上是否增加特效 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC10 游戏结束 |
| **范围** | 游戏功能 |
| **级别** | 子功能 |
| **主要参入者** | 游戏主系统 |
| **涉众及其关注点** | 用户希望获知本轮游戏最终积分 |
| **前置条件** | 游戏处于运行状态 |
| **成功保证** | 系统记录积分信息 |
| **主成功场景** | 方格到顶后，游戏画面静止，在屏幕上显示用户积分信息， |
| **扩展** | 无 |
| **特殊需求** | 需要屏幕设备 |
| **技术与数据变元素** | 无 |
| **发生频率** | 可能会间断发生 |
| **杂项** | 未决问题：考虑是否新设一个窗口以供用户选择“结束”/“下一轮”；考虑结算积分时是否加上额外积分 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC11 方块生成与显示 |
| **范围** | 游戏功能 |
| **级别** | 子功能 |
| **主要参入者** | 游戏主系统 |
| **涉众及其关注点** | 游戏系统希望能生成随机样式的方块，从随机位置下落；  用户希望能提前看到下一个方块样式 |
| **前置条件** | 游戏处于运行状态 |
| **成功保证** | 随机函数保证随机性，设置预显窗口 |
| **主成功场景** | 系统将随机生成的方块样式显示在预显窗口 |
| **扩展** | 无 |
| **特殊需求** | 需要屏幕设备 |
| **技术与数据变元素** | 无 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC12 方块移动 |
| **范围** | 游戏功能 |
| **级别** | 子功能 |
| **主要参入者** | 游戏主系统 |
| **涉众及其关注点** | 用户希望在没有任何操作的情况下，方块是垂直且匀速下落的 |
| **前置条件** | 游戏处于运行状态 |
| **成功保证** | 系统定时刷新 |
| **主成功场景** | 系统将随机生成的方块样式显示在预显窗口 |
| **扩展** | 无 |
| **特殊需求** | 需要屏幕设备 |
| **技术与数据变元素** | 无 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC13 处理增加难度请求 |
| **范围** | 游戏设置 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望点击“增加难度”后，游戏难度系数增加 |
| **前置条件** | 游戏未开始前 |
| **成功保证** | Swing提供相应接口响应事件 |
| **主成功场景** | 用户每点击一次“增加难度”按钮，初始游戏难度系数加一，最高难度系数为10，难度系数越高，下落速度越快 |
| **扩展** | 无 |
| **特殊需求** | 需要鼠标和屏幕设备 |
| **技术与数据变元素** | “增加难度”指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC14 处理降低难度请求 |
| **范围** | 游戏设置 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望点击“降低难度”后，游戏难度系数减小 |
| **前置条件** | 游戏未开始前 |
| **成功保证** | Swing提供相应接口响应事件 |
| **主成功场景** | 用户每点击一次“降低难度”按钮，初始游戏难度系数减一，最低难度系数为10，难度系数越低，下落速度越慢 |
| **扩展** | 无 |
| **特殊需求** | 需要鼠标和屏幕设备 |
| **技术与数据变元素** | “降低难度”指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC15 游戏升级 |
| **范围** | 游戏功能 |
| **级别** | 子功能 |
| **主要参入者** | 游戏主系统 |
| **涉众及其关注点** | 用户希望难度系数是随时间增长而自动增长的 |
| **前置条件** | 游戏已在运行状态 |
| **成功保证** | 系统记录积分信息 |
| **主成功场景** | 用户每积累一定分数，游戏难度系数自动升高一级，最高难度系数为10 |
| **扩展** | 无 |
| **特殊需求** | 需要屏幕设备 |
| **技术与数据变元素** | 无 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC16 新游戏 |
| **范围** | 游戏设置 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望放弃当前游戏，新开一局 |
| **前置条件** | 游戏已在运行状态，“新游戏”按钮可点击 |
| **成功保证** | 初始化游戏面板函数正确 |
| **主成功场景** | 用户点击“新游戏”按钮，所有信息全部初始化 |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标屏幕设备 |
| **技术与数据变元素** | “新游戏”指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC17 退出游戏 |
| **范围** | 游戏设置 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望退出游戏 |
| **前置条件** | “退出”按钮可点击 |
| **成功保证** | 无数组溢出等错误 |
| **主成功场景** | 用户点击“退出”按钮，结束当前程序的运行 |
| **扩展** | 无 |
| **特殊需求** | 需要鼠标屏幕设备 |
| **技术与数据变元素** | “退出”指令由鼠标输入 |
| **发生频率** | 一次 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC18 设置方块颜色 |
| **范围** | 游戏设置 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望更改方块颜色 |
| **前置条件** | 按钮可点击 |
| **成功保证** | swing中有接口 |
| **主成功场景** | 用户点击按钮，选择颜色，确认后方块颜色也随之改变 |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标屏幕设备 |
| **技术与数据变元素** | 指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

|  |  |
| --- | --- |
| **用例编号与名称** | UC19 设置背景颜色 |
| **范围** | 游戏设置 |
| **级别** | 子功能 |
| **主要参入者** | 用户 |
| **涉众及其关注点** | 用户希望更改背景颜色 |
| **前置条件** | 按钮可点击 |
| **成功保证** | Swing中有接口 |
| **主成功场景** | 用户点击按钮，选择颜色，确认后画布颜色也随之改变 |
| **扩展** | 点击无反应的情况下进行错误提示 |
| **特殊需求** | 需要鼠标屏幕设备 |
| **技术与数据变元素** | 指令由鼠标输入 |
| **发生频率** | 可能会不断发生 |
| **杂项** | 无 |

1. 补充性规格说明

可用性：

1. 为了让用户更好地体验本游戏，增加使用说明书到系统中；
2. 采用图形化界面，除了游戏操作需要使用键盘，其他操作仅需轻点鼠标即可实现。

可靠性：需要体现游戏的稳定性与反应速度。

1. 词汇表

画布：俄罗斯方块游戏主界面的长方形背景板，由许多同样大小的方格组成

样式：由小方格组成的基本图形，如T字形、Z字形等。

方块：为了便于程序的描述，我们把样式放置在一个4\*4的方块内考虑，对这些样式的变形移动操作均视为对方块的整体操作

游戏主系统：游戏的主题模块，包含游戏设置、游戏操作、游戏功能、游戏控制四个子模块

1. 设想

本项目预计完成一个面向所有年龄段的，为玩家提供图形可视化界面的俄罗斯方块游戏软件。他应该具备一个经典的俄罗斯方块应该有的基本功能，包括对方块的移动、旋转、堆积、消除、游戏积分等；此外，如果时间允许，我们可以增加用户登录，联机对战等功能，打造一个怀旧小游戏社交平台。

1. 业务规则

（1）在二维的平面里用各种随即产生的方块堆积木，每填满一行消去一行，当到达顶部时，游戏结束。

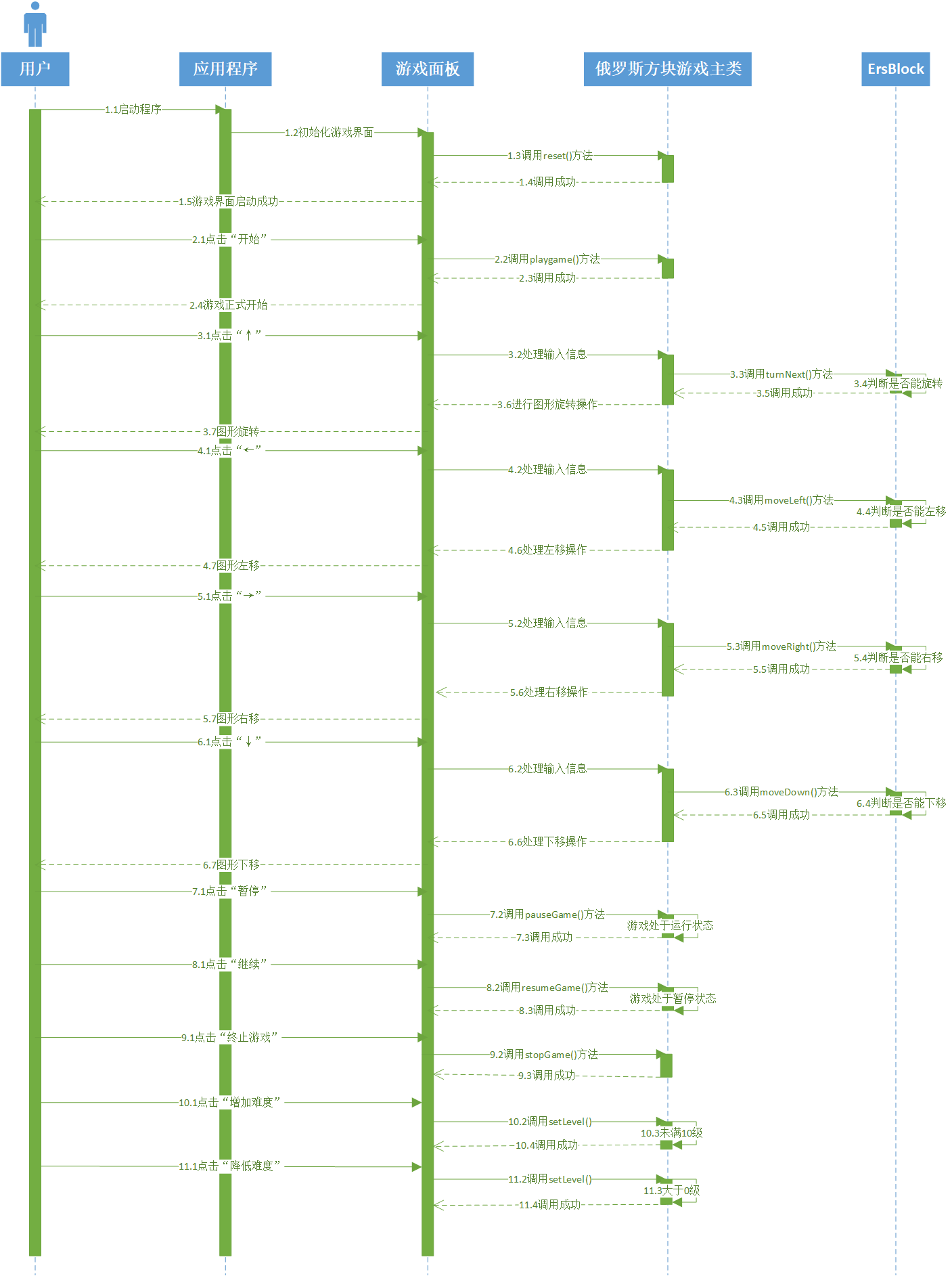
（2）玩家能通过方向键来控制方块的转动，左移，右移和直落。

（3）每种类型的方块都有颜色。

（4）游戏能够在玩的过程中，给出玩家的分数，分数是由随即方块的类型决定的，每堆一个方块，就将分数累加到总分中。

（5）游戏有暂停、开始和结束、游戏规则、游戏说明等。

1. 系统设计

**系统时序图**

**操作契约**

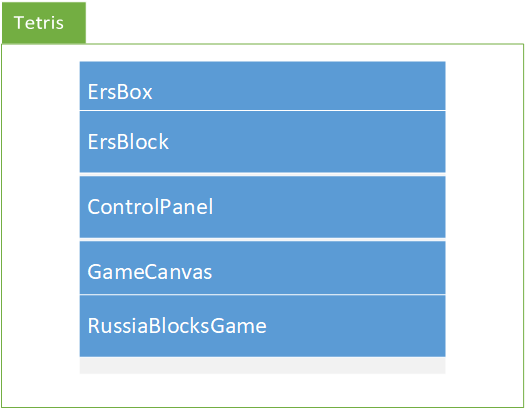
|  |  |
| --- | --- |
| 操作 | reset（） |
| 交叉引用 | UC16:新游戏 |
| 前置条件 | “新游戏”按钮可用 |
| 后置条件 | 游戏画布清空，控制面板复位，新一轮游戏可以开始 |

|  |  |
| --- | --- |
| 操作 | pauseGame（） |
| 交叉引用 | UC3：处理暂停请求 |
| 前置条件 | 暂停前游戏应处于运行状态 |
| 后置条件 | 点击“暂停”后，游戏画面静止 |

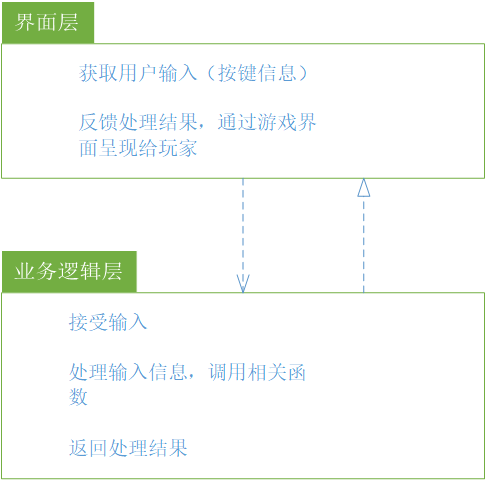
|  |  |
| --- | --- |
| 操作 | resumeGame（） |
| 交叉引用 | UC4：处理继续请求 |
| 前置条件 | 继续前游戏应处于暂停状态 |
| 后置条件 | 恢复时，能从中断处继续开始 |

|  |  |
| --- | --- |
| 操作 | setLevel（） |
| 交叉引用 | UC13处理增加难度请求、UC14处理降低难度请求 |
| 前置条件 | 游戏未开始 |
| 后置条件 | 使用后，方块下落速度会改变 |

**包图**

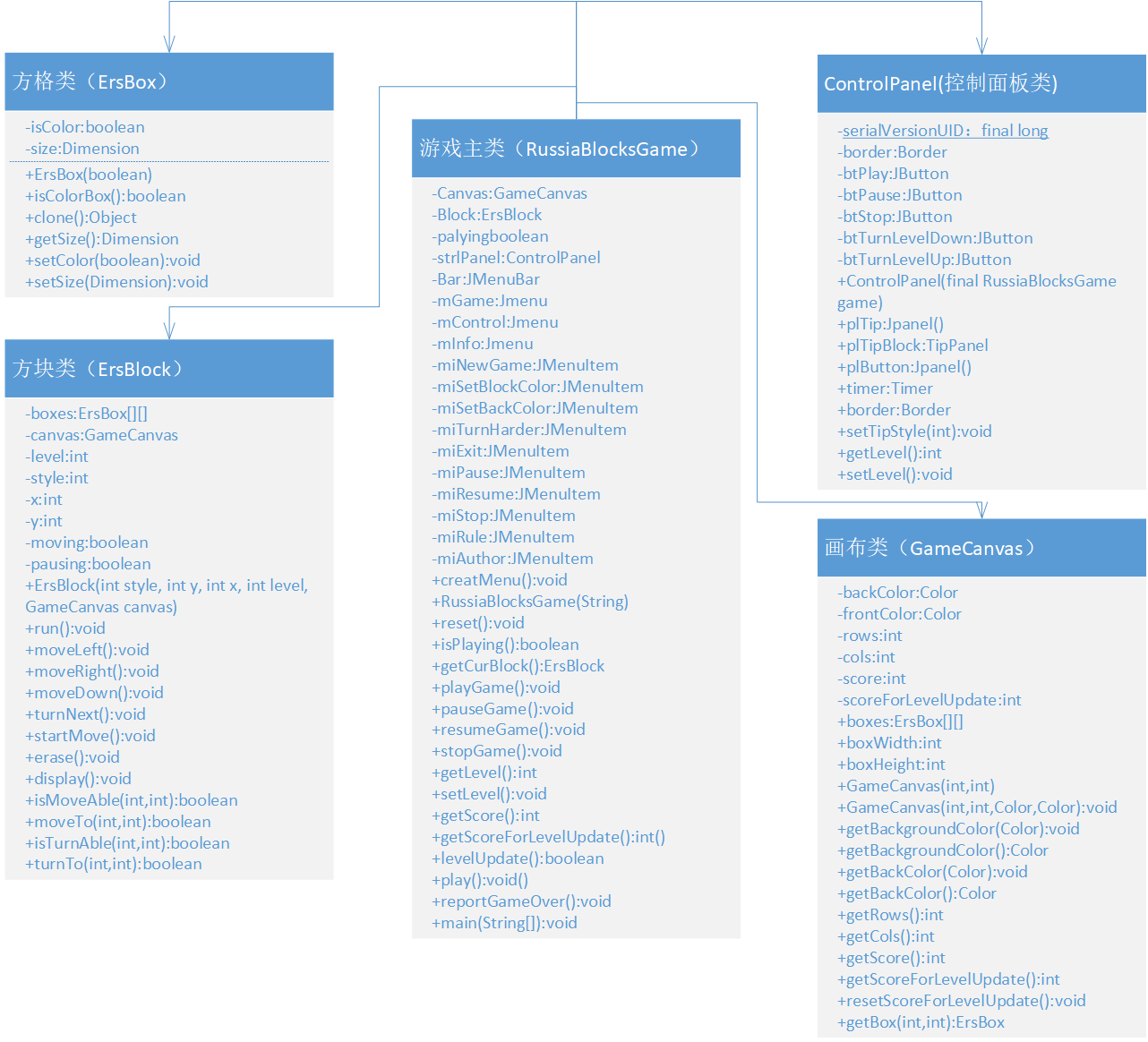


**逻辑分层**

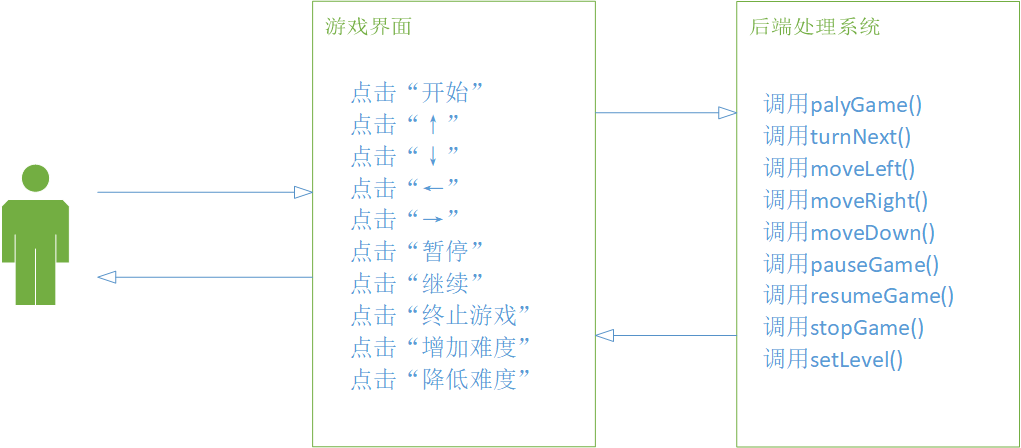


1. 详细设计

**类图**

****

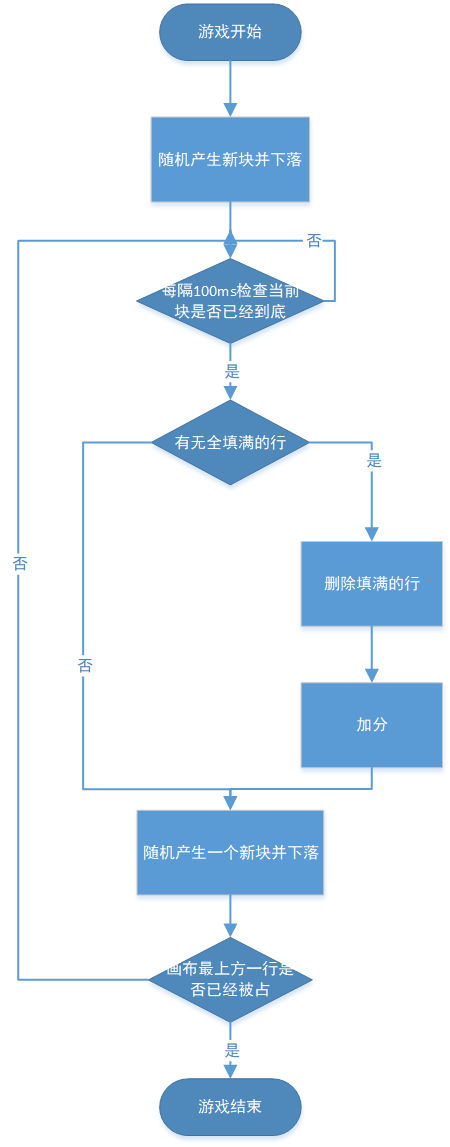
**交互图**

****

**UI设计**

1. 代码实现

流程图



核心代码解析

1. 一轮游戏：

（1）随机产生新的块和下落列号

int col = (int) (Math.random() \* (canvas.getCols() - 3));

int style = ErsBlock.STYLES[ (int) (Math.random() \* 7)][(int) (Math.random() \* 4)];

（2）检查是否有全填满的行

checkFullLine(); //

if (isGameOver()) {

reportGameOver();

miPlay.setEnabled(true);

miPause.setEnabled(false);

miResume.setEnabled(false);

miStop.setEnabled(false);

ctrlPanel.setPlayButtonEnable(true);

ctrlPanel.setPauseButtonLabel(false);

ctrlPanel.setStopButtonEnable(false);

return;

}

block = new ErsBlock(style, -1, col, getLevel(), canvas);

block.start();

col = (int) (Math.random() \* (canvas.getCols() - 3));

style = ErsBlock.STYLES[ (int) (Math.random() \* 7)][(int) (Math.random() \* 4)];

ctrlPanel.setTipStyle(style);

}

}

（3）检查画布中是否有全填满的行，如果有就删之

public void checkFullLine() {

for (int i = 0; i < canvas.getRows(); i++) {

int row = -1;

boolean fullLineColorBox = true;

for (int j = 0; j < canvas.getCols(); j++) {

if (!canvas.getBox(i, j).isColorBox()) {

fullLineColorBox = false;

break;

}

}

if (fullLineColorBox) {

row = i--;

canvas.removeLine(row);

}

}

}

（4）根据最顶行是否被占，判断游戏是否已经结束了，true-游戏结束了，false-游戏未结束

private boolean isGameOver() {

for (int i = 0; i < canvas.getCols(); i++) {

ErsBox box = canvas.getBox(0, i);

if (box.isColorBox()) {

return true;

}

}

return false;

}

}

1. ErsBox类

**（1）**方格类的构造函数，true-前景色，false-背景色

public ErsBox(boolean isColor) {

this.isColor = isColor;

}

（2）此方格是不是用前景色表现

public boolean isColorBox() {

return isColor;

}

（3）设置方格的颜色

public void setColor(boolean isColor) {

this.isColor = isColor;

}

（4）得到此方格的尺寸

public Dimension getSize() {

return size;

}

（5）设置方格的尺寸

public void setSize(Dimension size) {

this.size = size;

}

（6）覆盖Object的Object clone()，实现克隆

@Override

public Object clone() {

Object cloned = null;

try {

cloned = super.clone();

} catch (Exception ex) {

ex.printStackTrace();

}

return cloned;

}

}

1. ErsBlock类

（1）7种模型的28种状态

public final static int[][] STYLES = {

{0x0f00, 0x4444, 0x0f00, 0x4444}, //长条型的四种状态

{0x04e0, 0x0464, 0x00e4, 0x04c4}, //T型的四种状态

{0x4620, 0x6c00, 0x4620, 0x6c00}, //反Z型的四种状态

{0x2640, 0xc600, 0x2640, 0xc600}, //Z型的四种状态

{0x6220, 0x1700, 0x2230, 0x0740}, //7型的四种状态

{0x6440, 0x0e20, 0x44c0, 0x8e00}, //反7型的四种状态

{0x0660, 0x0660, 0x0660, 0x0660}, //方块的四种状态

（2）构造函数，产生一个特定的块：

style 块的样式，对应STYLES的28个值中的一个

y 起始位置，左上角在canvas中的坐标行

x 起始位置，左上角在canvas中的坐标lie

level 游戏等级，控制块的下落速度

canvas 画板

public ErsBlock(int style, int y, int x, int level, GameCanvas canvas) {

this.style = style;

this.y = y;

this.x = x;

this.level = level;

this.canvas = canvas;

int key = 0x8000;

for (int i = 0; i < boxes.length; i++) {

for (int j = 0; j < boxes[i].length; j++) {

boolean isColor = ((style & key) != 0);

boxes[i][j] = new ErsBox(isColor);

key >>= 1;

}

}

display();}

（3）块向左移动一格

public void moveLeft() {

moveTo(y, x - 1);

}

（4）块向右移动一格

public void moveRight() {

moveTo(y, x + 1);

}

（5）块向下移动一格

public void moveDown() {

moveTo(y + 1, x);

}

（6）块变型

public void turnNext() {

for (int i = 0; i < BLOCK\_KIND\_NUMBER; i++) {

for (int j = 0; j < BLOCK\_STATUS\_NUMBER; j++) {

if (STYLES[i][j] == style) {

int newStyle = STYLES[i][(j + 1) % BLOCK\_STATUS\_NUMBER];

turnTo(newStyle);

return;

}

}

}

}

public void startMove() {

pausing = false;

moving = true;

}

1. ControlPanel类

（1）控制面板类，继承自JPanel。设置预显窗口，等级，得分，控制按钮，主要用来控制游戏进程。

class ControlPanel extends JPanel {

private static final long serialVersionUID = 3900659640646175724L;

private JTextField tfLevel = new JTextField("" + RussiaBlocksGame.DEFAULT\_LEVEL),tfScore = new JTextField(" 0"),tfTime = new JTextField(" ");

private JButton btPlay = new JButton(" 开始"),

btPause = new JButton(" 暂停"),

btStop = new JButton("终止游戏"),

btTurnLevelUp = new JButton(" 增加难度"),

btTurnLevelDown = new JButton(" 降低难度");

private JPanel plTip = new JPanel(new BorderLayout());

private TipPanel plTipBlock = new TipPanel();

private JPanel plInfo = new JPanel(new GridLayout(4, 1));

private JPanel plButton = new JPanel(new GridLayout(6, 1));

private Timer timer;

private Border border = new EtchedBorder(EtchedBorder.RAISED, Color.white, new Color(148, 145, 140));

1. GameCanvas类

（1）设置游戏背景色彩

public void setBackgroundColor(Color backColor) {

this.backColor = backColor;

}

（2）取得游戏背景色彩

public Color getBackgroundColor() {

return backColor;

}

（3）设置游戏方块颜色

public void setBlockColor(Color frontColor) {

this.frontColor = frontColor;

}

（4）取得游戏方块色彩

public Color getBlockColor() {

return frontColor;

}

（5）取得画布中方格的列数

public int getRows() {

return rows;

}

（6）取得画布中方格的行数

public int getCols() {

return cols;

}

（7）取得游戏成绩

public int getScore() {

return score;

}

（8）取得自上一次升级后的积分

public int getScoreForLevelUpdate() {

return scoreForLevelUpdate;

}

（9）升级后，将上一次升级以来的积分清零

public void resetScoreForLevelUpdate() {

scoreForLevelUpdate -= RussiaBlocksGame.PER\_LEVEL\_SCORE;

}

（10）得到某一行某一列的方格引用

public ErsBox getBox(int row, int col) {

if (row < 0 || row > boxes.length - 1 || col < 0 || col > boxes[0].length - 1){

return null;

}

return (boxes[row][col]);

}

（11）覆盖JComponent类的函数，画组件

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(frontColor);

for (int i = 0; i < boxes.length; i++) {

for (int j = 0; j < boxes[i].length; j++) {

g.setColor(boxes[i][j].isColorBox() ? frontColor : backColor);

g.fill3DRect(j \* boxWidth, i \* boxHeight,

boxWidth, boxHeight, true);

} }}

（12）当一行被游戏者叠满后，将此行清除，并为游戏者加分

public synchronized void removeLine(int row) {

for (int i = row; i > 0; i--) {

for (int j = 0; j < cols; j++) {

boxes[i][j] = (ErsBox) boxes[i - 1][j].clone(); }

}

score += RussiaBlocksGame.PER\_LEVEL\_SCORE;

scoreForLevelUpdate += RussiaBlocksGame.PER\_LEVEL\_SCORE;

repaint();

}

1. 测试
2. 测试环境：

设备：PC机

操作系统：windows 10

CPU：Intel(R) Core(TM) i5-7200U CPU @2.50GHz 2.71GHz

内存：4.00GB

系统类型：64位操作系统，基于x64的处理器

测试软件：eclipse

1. 测试用例与结果

（1）游戏控制模块测试

|  |  |
| --- | --- |
| 输入等价类 | 鼠标点击 |
| 无效等价类 | 对非控制模块的点击 |
| 有效等价类 | 对控制模块中按钮的点击 |
| 测试结果 | 游戏可以被控制开始、暂停、继续、结束 |

（2）游戏设置模块测试

|  |  |
| --- | --- |
| 输入等价类 | 鼠标点击 |
| 无效等价类 | 对非设置模块的点击 |
| 有效等价类 | 对设置模块的点击 |
| 测试结果 | 点击“新游戏”可以直接复位游戏面板、更换方块颜色和背景颜色后面板颜色发生改变、设置难度后方块降落速度改变，点击“退出”后该程序结束 |

（3）游戏控制模块测试

|  |  |
| --- | --- |
| 输入等价类 | 键盘输入 |
| 无效等价类 | 除“↑，↓，←，→”外的按键 |
| 有效等价类 | “↑，↓，←，→” |
| 测试结果 | 敲击相应键盘按键，方块可以在下落的同时变形，下降，左移，右移 |

（4）游戏功能模块测试

|  |  |
| --- | --- |
| 测试功能 | 积分 |
| 测试结果 | 填满一行，消除一行，积分加100分，未填满则不加分 |

|  |  |
| --- | --- |
| 测试功能 | 下落 |
| 测试结果 | 未进行任何操作时，屏幕上的方块可垂直匀速下落 |

|  |  |
| --- | --- |
| 测试功能 | 预显 |
| 测试结果 | 预显窗口显示下一个方块 |

1. bug修改记录
2. 方格显示不能随着屏幕的调整而调整

解决方法：增加自适应函数，根据长宽自动调节方格大小

1. 界面排布不美观，不整齐

解决方法：修改布局方法，反复调整参数

1. 积分累计出错

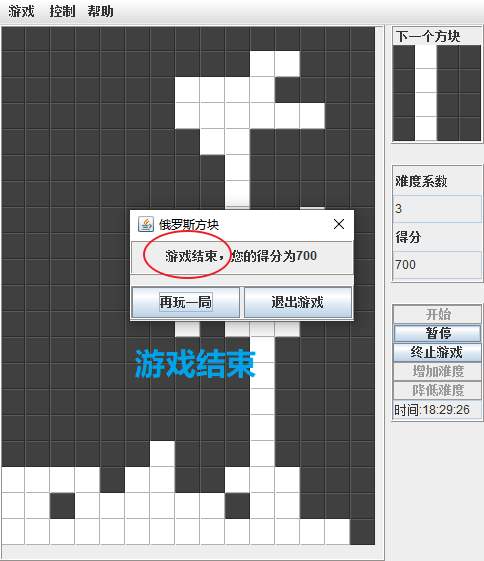
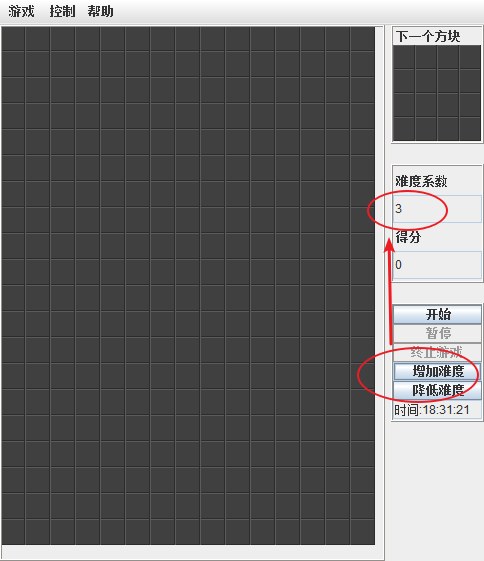
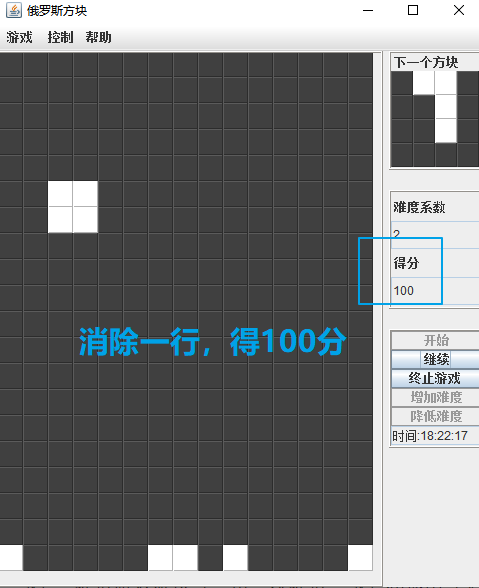
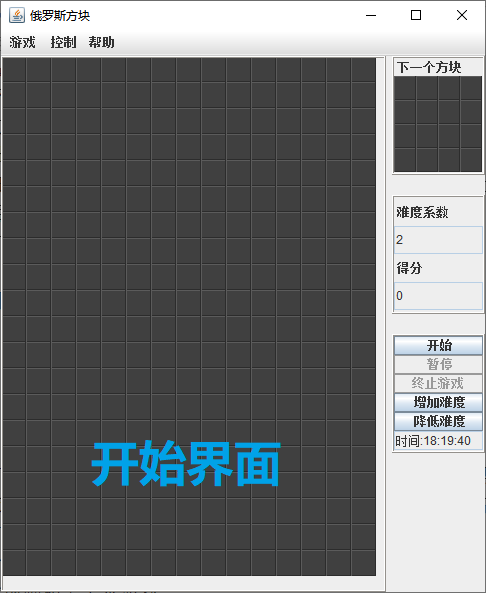
解决方法：仔细检查后，发现是计算积分把升级积分当成当前积分累计上去了

1. 点击按钮后，没有出现预期反应

解决方法：没有设置按钮状态，调整为启用即可

1. 软件运行与使用说明

软件运行：



使用说明：

1. 在游戏界面上方的控制栏中，有三个选项，分别是：游戏、控制和帮助。

游戏：新游戏（重新开始一局）、设置方块颜色、设置背景颜色、升高游戏难度（点击一次升高一级）、降低游戏难度（点击一次降低一级）和退出（关闭游戏界面）。

控制：开始、暂停、恢复和终止游戏（让难度和积分回到初始状态）。

帮助：游戏规则和关于本游戏。

1. 在游戏界面右侧的控制栏中，最上方显示下一个方块，中间显示目前难度等级和目前总得分，最下方为开始、暂停、终止游戏、增加难度、减小难度按钮和当前时间。
2. 用户再打开游戏界面之后，点击“开始”可以开始玩一局新游戏，按“↑”键方块旋转（按一下旋转一次，长按持续旋转），按“←”、“→”方块向左、向右移动（按一下移动一次，长按持续旋转），

方块每填满一行自动消除，积分增加100分，每满600分难度自动上升一级。难度可以自主调节，点击

1. 设计体会

此次软件设计实践，老师并未明确规定要交出一份什么样的作品，因此我们可以选择的范围很广，包括但不限于小程序、网站还有应用程序等，但是因为之前没有相关开发经验，且团队只有两人，所以我们斟酌再三，还是选择做出一个应用程序来，参考老师给我们发的别的专业同学的作品，我们决定也做一个经典的小游戏——俄罗斯方块，这样的小游戏代码量在我们的控制范围内，业务逻辑也不至于太复杂，任务量虽然有些繁重但是也能接受。确定了目标后，我们就开始考虑开发语言，首选自然是我们最熟悉的语言——C++，但是后来我们改用了java。因为诸如俄罗斯方块这样的小游戏自然少不了图形界面的设计，GUI的图形界面也是一个我们没有怎么接触过的知识，而在了解的过程中，我们发现C++的GUI设计比较繁杂，一个简单的按钮可能就需要很大一部分的代码量，为了避免不必要的劳动量，我们选择了相较而言更加简单的java，希望把有限的精力花费在完善系统的功能上，而不是过多地关注如何实现图形化界面。

在确定了选题、完成了前期的调查后，我们就开始根据老师的文档流程，进行基本的设计，这些UML设计的知识，我们此前只在软件工程里简单了解过，但是具体的内容，比如用例图表、包图、时序图、类图等设计都没有接触过，因此我们也借鉴了很多资料，包括《构建之法》、CSDN上的各种博客、知乎和简书等，这些也会在参考资料中一一列出。为了赶在专业实习前，完成全部的设计，我们省略了一部分不必要的图，只画了我们所需要的，终于赶在了实习前完成任务。

设计阶段大致完成后，我们的实训任务也开始了，实训期间的“分层架构”设计这一观念对我们的影响很大，进而开始反思我们的设计：能够明显发现我们原先的设计把“前后端”都混在了一起，思路十分不清晰，于是又开始了新一轮的修改。当我们把这个系统的框架建造好了以后，剩下的代码似乎也不是那么难了，当时我们是这么想的，然而并没有，尽管java的GUI设计已经不算困难，但是从入门到实践，显然不是那么容易的一件事，再加上我们对java的应用不够得心应手，期间艰辛良多，修改也很多，最终在实训结束时才完成了大致的代码。

在放假回家前，我们完成了测试工作，一个俄罗斯方块终于算是完成了，也许功能不够齐全、在某些地方还存在小bug，但是他已经是个足够吸引人的小游戏了。回家以后，我们开始写文档，作品完成后再去回顾那些设计图，又能发现其中的某些不足，但是作品已经不好再更改了，我们决定还是继续之前的设计。写文档和写代码谁更痛苦，我们也说不出来。这些过程少不了与队友的协商合作，团队合作纵然这个团队只有两个人，也少不了摩擦，多人的小组摩擦只会更多，如何协调组员也是我们在团队合作中需要考虑的问题，一份完善的作品背后一定有一个默契的团队。

最后，感谢这个课程，他让我们完整的实现了一个项目，期间收获已流露在字里行间，不再一一表。

1. 附录

硬件要求：

笔记本电脑

推荐配置为

CPU方面：单核主频2.0GHz以上

内存方面：2GB及以上

硬盘方面：20G可用磁盘空间

系统要求：

推荐windows7，windows8或windows 10

工具要求：

Eclipse

环境要求：

遵循JAVA标准编译

1. 参考文献