开发文档

沈征宇 18302010009

**程序结构设计与分析**:

任务刚布置下来时,我还没有多少**面向对象**思维,仍然在使用**面向过程**思路,发现设计过程非常复杂，需要完备地考虑各种可能发生的情况，即开始之前就设计好整个过程，这**极大地挑战**了我的思考能力，最终以失败告终。咨询了**张星宇**学长之后，他为我提出了“先去自学面向对象吧”的建议。我接受了这个建议，开始了面向对象的自学。对开拓思路有了很大帮助。比如：将“方块”设计成**抽象类**，各种形状的方块设置为其**子类**，当前的方块，即将到来的方块都是一个**对象**。它的行列、小方块分布、旋转方法都存储在这个对象之中，成为这个对象的属性。将“游戏”设计成**类**，其中有地图，中间以“ ”先行填充。

在Java程序设计的过程中，我一边学习，一边改进自己的代码，进行了**多次重构**，取得了比较好的效果。比如，学习了“**数组**”，我把俄罗斯方块存为数组而非手绘数组，学习了“**方法**”，我将方块的各种动作抽象为方法，大大方便了程序的编写，使其看起来**精简**的多，学习了“**包**”、“**继承**”、“**多态**”，我将各种类型的方块存到一个包中，并编写一“抽象类”方块类，其繁复多样的子类（I方块，J方块，O方块等等）**继承于**抽象类“方块”，在方块子类中重写父类旋转方法以**适应**不同种类方块的旋转方式。

此时，我完成了除了**图形化界面以外**的工作。接下来我开始辛苦学习JAVA FX。我的思路就是，既然我已经有了PJ1为基础，那么我只要把PJ1中的命令行显示法改成图形界面显示法就稳了。如此思路仍然不变，我们有一个**game类，它存储了游戏的状态**，它在底下——用户看不见的地方悄悄地存在、运行者，而gamepane类构造了一个大舞台，核心当然是游戏界面，所以我们先来谈谈游戏界面。它是一个**Gridpane**，就像命运赠送的礼物，早已在暗中标好了价格一样，我的界面早已被设定为一格一格划分好的面板，当game中有方块存在，就按它的状态涂色，原来的15\*10的Map，此处就有对应的界面显示。方块有这些：**正下落的、已装载的（落下定死的）、和用来提示的**，我将它们分别涂色显示。而它的移动又是怎么实现的呢？我有一个**定时器**，每隔一定时间，Game类中的当前方块的位置就下落一格，因此显示也自然的被牵动了。

核心界面的左右护法，是左界面**指导界面**，和右界面信息界面。左指导界面由于要显示下一个方块，所以做了GridPane，加上各种**按钮**。右**信息界面**显示了时间、分数、和道具数目。

游戏开始时的界面导向多个不同的界面，各个界面都是由**方法**做出。

在我的项目中打包了一些常重复出现的代码，使代码更简洁。

主要函数的功能：

Main类：

**Main函数**：游戏运行、游戏进入界面展开（各个按钮**导向不同的功能**）、初始化各参数、开始播放音乐。

Game类：

**getRandomBlock函数**：获得一个七种方块之中任意类型的方块对象。

**nextBlock函数**：为对象“nextBlock”赋**当前位置，方块类型**。

**Initial函数：**初始化游戏界面属性（皆为**空墙**）。

**Isover函数**：判断游戏是否结束（已无法再进行）了。

**Collide函数**：返回一个**布尔类型值**，它表现了在进行了某个操作之后，是否现在的方块会和墙、旧方块发生碰撞。

**reachTheBottom函数**：判断是否方块已到底，如果到底，就可以把它**固定**到地图里了。

**LoadBlock函数**：当方块已无法再降，将他装到地图里，成为地图的“元素”。

**Eliminate函数**：消除已满的行，为玩家加分。

**goLeft,goRight,goDown,goBottom函数**：方块的移动操作。

**Weapon函数：**道具的使用。

**EliminateGroup函数**：打包了**装载方块、消除、更新方块**和**获得下一个方块**操作。

**nextRotatedCells函数**：返回一个当前方块旋转后的方块，**但对当前方块不进行任何改动**。

**Rotate函数**：对当前方块进行**真正的旋转**。

**getCells函数**：获得当前的方块，这是为了更好的封装性。

**PrintMap函数**：画一个初始化好的方块。

Gamepane类：

**带参构造方法**：传入用户名、存档位置、底部游戏、音量。

其中：首先根据Main中的**bgm参数**选择bgm播放

然后舞台建立、**定时器**启动。

准备工作有划好格子、搭建三个子界面等一切后续要用到的东西。

对gamepane进行**键盘监听**，游戏操作基本只需要键盘就能完成。

**特别地，**按到R（重玩键）会跳出新窗口，来确认你的操作。

**Paint方法**：用图形化的方法来显示整个游戏。

首先根据Main中的参数来决定背景，这个背景是在开始界面设定好的。然后判断阴影，再移除之前游戏界面里的所有**子节点**，相当于重绘界面。遍历游戏地图，遇到正在下落的涂一色，已经落地的涂一色，阴影涂一色，它有些透明。如果选择了**盲打模式**，会造一个黑长方形，其中挖去了一个**动圆**来透出方块，最后盖在整个游戏界面上。信息界面则用标签来显示信息，帮助界面则显示下一个方块和用以辅佐游戏的按钮。

Operation类：

**setBackground方法**：传入要设置背景的pane和图片url。

**outGame方法：**以.dat格式输出当前的游戏。

**printRanking方法：**得到这局的分数，然后做一次**排序，插入，显示。**

**Button的种种方法：**返回一个按钮，有的自定文字，有的确定了。

**beautifulText方法：**返回一个美化过的字符串。

**playLowerVolumeMusic：**传入**降音量倍数**和**音乐url**，减轻音量并播放。

**Showstage的种种方法**：呈现**重启界面、选择关卡界面，帮助界面，重载游戏界面，设置界面**（包括设置音量、背景图片、bgm）、登陆界面。

编程中遇到的问题和解决⽅法：

1. 在判断旋转后是否会碰撞的时候，不能真去旋转，较好的解决办法是造一个旋转后的方块而**不去改变现有的方块**，利用这个被造出来的方块来判断是否碰撞，如果不碰撞，就可以放心地去真正地旋转方块了。
2. 拓展部分要求实现排行榜，我自学了文件I/O的知识，以txt文件保存了排行榜的信息，但在“读取路径”这个问题上，我犯了难，当我把项目发送给助教之后，该文件的绝对路径是不一样的，所以，我学习了相对路径的知识，解决了问题，我也了解到，**相对路径**其实也是绝对路径，只是**编译器**帮我们在底层添加了上几层的目录。
3. 在装载方块进入地图数组的时候，因为我采用了4\*4的容器来装方块（3\*3容器更简单，**但长条只能实现三格**，故不采用），极有可能容器底部一行没有方块，因此触底的一行实际上是中间的两行之一，那么装载时，触底那一行被装到了地图数组底部，触底行的下一行就被装载到地图下面，这会造成数组越界。我的办法是：**先判断方块最底行在哪一行**，然后从上向下装载方块，最底行还底的行不进行装载，完美解决。

一：排行榜：我的排行榜在游戏结束后显示，自动显示前10名分数。

二：存档&读档：我开发了支持**多玩家，多存档位置**的存读档方式，多名玩家可以使用存档功能，每个玩家可以有多个存档位置**自行选择**，且这些存档一直存在，不会被销毁。

三：创意：道具系统（**combo使用，连击奖励**）

1. 在实现“盲打模式”的时候，遇到了很大的困难，没有现成的已习得的方法使用，网上搜索也没有得到较好的解决办法。去**浏览API**，忽得**Shape.subtract**方法，是从一图形中挖出另一图形，完美解决。这让我知道没有现成的研究结论可以沿用的时候，就要自己**努力去探索**。
2. 事件处理与主代码低耦合，所以**无法直接去读取现有对象的状态**，于是我把一些状态参数（**游戏音量、下落速度、游戏模式、照片选择、音乐选择**等）放在主类里，成为**静态变量**，以待操作，取得成功。
3. 一开始我将各种显示stage的方法**分别写成类**，但这样不好，类太多了，于是我把他放在operation类里，成为各个方法，搭出Stage并直接show。
4. 有时按钮被按动了以后，主界面就不再接受键盘指令，后来了解到这是焦点问题。一定要按完以后就把焦点重新放到界面里。
5. 对象的序列化读取。一定要读取game对象，我在trace里找到game对象的序列号，然后定义好。