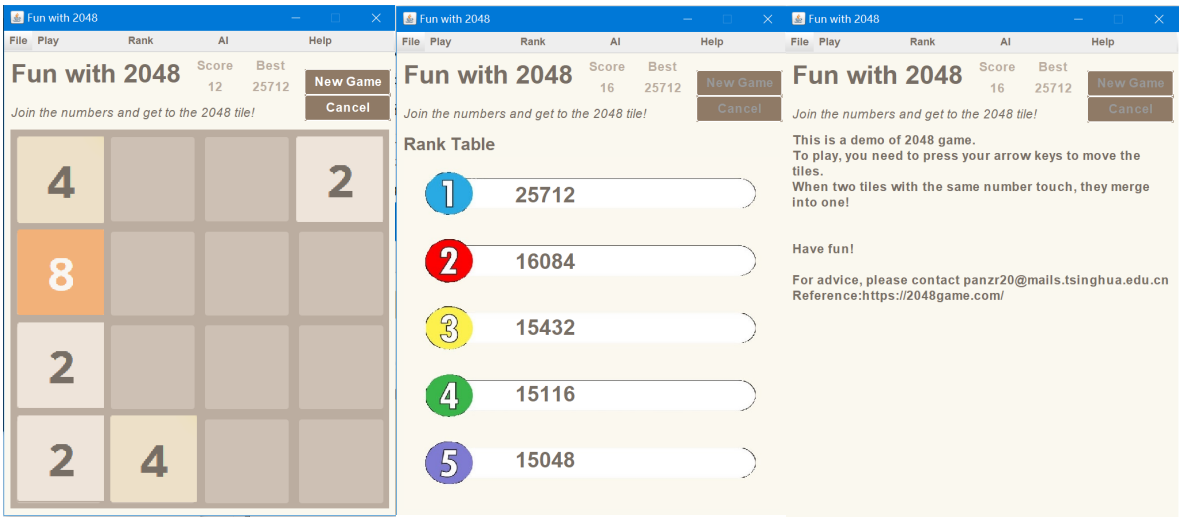
Java程序设计基础大作业报告

潘子睿 计05 2020010960

1.游戏操作方式

- 打开`.jar`文件即进入游戏界面，点击右上角×自动退出。在游戏过程中，可以随时点击`new game`按钮重新开始，或是点击`cancel`按钮退回上一步。
- 砖块（Tile）的移动依赖于键盘的上下左右按键。例如，按下↑，所有砖块都会尝试向上方移动以及合并。
- 砖块的移动规则见作业文档。当所有砖块均无法被移动或是合并时，游戏结束。此时根据弹出对话框的提示，可以选择重新开始或是直接退出。



2.游戏架构与设计

文件

```
├── executable
|   ├── 2048.jar
|   ├── config
|   |   └── config.txt
|   └── pic
|       ├── *.png
└── src
    ├── AITimerListener.java
    ├── Agent.java
    ├── Board.java
    ├── ChessPanel.java
    ├── HelpPanel.java
    ├── Play.java
    └── RankTable.java

4 directories, 29 files
```

本次大作业的设计主要分为两个部分，分别为**底层算法**以及UI的设计。

底层算法部分主要包括砖块的移动与合并，以及AI自动玩游戏。前者在`Board.java`中被封装为类`Board`，后者在`Agent.java`中被封装成类`Agent`。

UI的设计上，首先游戏的主页以及其上各个控件的逻辑在`Play.java`中实现。该类也是本项目的入口类。`Play.java`的`JFrame`上引用了三个`JPanel`，分别为`ChessPanel`（主游戏页面，玩家模式与AI模式共用），`HelpPanel`（帮助页面，提示基本的游戏规则以及关于本游戏的一些信息），`RankTable`（排行榜页面，显示本地保存的五个最高分）。`RankTable`类中其实也包含一些简单的算法，例如维护一个单调的、限制长度的队列。`ChessPanel`中根据`Board`类中传出的数字移动信息来对页面的动画进行渲染。

`AITimerListener.java`中实现的是针对AI模式的一个计时器。更具体地，是让AI每隔500ms尝试根据当前局面做出一个最佳决策。

3.游戏功能说明

基本功能全部实现，这里不再赘述，但在基本功能的基础上还实现了一些小的附加功能：

- `cancel`按键可以回退到上一步；但需注意，游戏只存储了上一步的状态，因此只能回退一步。（其实在此基础上回退多步也很容易实现）
- 页面上会显示实时的分数以及历史最高分。需要注意的是，当实时分数超过历史最高分时，历史最高分也会被实时更新。但这不意味着该得分已经被保存了。本次大作业中实现的逻辑为当选择重新开始、将当前局面保存到本地、存本地读取某个进度文件、在玩家模式与AI模式间切换、游戏结束（指达到了游戏结束的条件，无法进行移动与合并）这些情况下会将本次的得分存入历史文件。
- 增加了帮助页面，提示玩家基本的游戏操作。

下面主要叙述提高功能的实现。

- **动画功能。**

  本次大作业中实现了各砖块的平滑移动，以及当砖块合并时，其大小会有相应的变化。另外，对于新出现的砖块，其会有一个从小到大的动画，以提示玩家这是新产生的砖块。

  具体实现上，`ChessPanel`类中会有一个计时器，其每10ms会触发一个事件。在这个事件中，会另开一个新的线程对页面进行渲染。需要注意的是，只有当游戏中的某个事件（向上、下、左、右移动砖块）后才会触发页面的更新。并且，每次更新会在13个周期内完成，理论上也就是130ms，但实际可能会有少许延迟。13个周期中，前8个周期用于渲染砖块的移动，通过**定比分点**，在给定源和目的的情况下可以得到某个周期时砖块的位置。后5个周期用于渲染砖块大小的变化，针对的是合并的以及新增的砖块，砖块的大小会经历一个先增大后减小直至恢复的过程。（至于为什么选13，实际测试中，这个数字不至于让动画太慢以至于被判为卡顿，也不至于让动画太快了而不易观察）

  `ChessPanel`类中通过两个局面的对比来判断哪些砖块被合并了、被移动了或是新增加的。具体的逻辑比较复杂，实现细节请见`ChessPanel.move_and_merge()`这个函数。实现动画的核心代码请见`ChessPanel.MyTimerListener.actionPerformed()`。

- **实现保存游戏及排行榜**

  本次大作业实现了可以在任何时刻将本局面信息保存至本地某个`.txt`文件，也支持从任何**合法的**`.txt`文件中读取局面信息。如果读取失败或是`.txt`文件不合法，会回到操作之前的局面。

  保存以及读取详见`Board.load()`和`Board.save()`，实现起来其实非常简单，只需要统一格式，保证读取和保存合法即可。

  本次大作业实现了可以在本地保存至多5个历史最高分。当然想保存更多也很简单，修改`RankTable`中的上限即可。具体地，在`RankTable`中维护一个单调的数组，新的数据被插入时依次与数组中的各个元素进行比对即可。由于数据较小，不妨在插入时直接让后续的数字全部后移一位。

- **AI模式**

本次大作业中实现了一个AI，可以根据当前局面给出对最佳的下一步操作的判断。

AI进行判断的算法逻辑具体如下：首先，基本思想是**枚举**。设置一个最大深度，程序会遍历这个最大深度之前的所有判断组合，考察实行什么样的操作对结果最有利。需要注意的是，由于2048游戏带有随机性，因为每次操作后新生成的数字可能是2也可能是4，另外生成的数字位置也是随机的。（本大作业文档中提到生成2或是4是等可能的，生成位置也是等可能的）。因此，每层的操作后，还需要对新生成的数字的位置进行枚举，对2和4的情况分类讨论，将二者的得分加权相加。

算法的流程如上，实现时采用递归。计算某个局面的得分的函数如下：

```
double finalScore() {
    double sum = 0;
    for (int row = 1; row <= board_length; row++) {
        sum = sum + EmptyScore * emptyCellsInRow(row)
            + MergeScore * mergeInRow(row)
            + MonoScore * monoInRow(row)
            + SumScore * sumRow(row);
    }
    for (int column = 1; column <= board_length; column++) {
        sum = sum + EmptyScore * emptyCellsInColumn(column)
            + MergeScore * mergeInColumn(column)
            + MonoScore * monoInColumn(column)
            + SumScore * sumColumn(column);
    }
    return sum;
}
```

也即，得分主要考察四个方面：分别是每行每列的空格数、潜在的合并个数、单调值数以及总和。这其中，单调指数实际上是很重要的一个方面，因为如果将某一行单调递增或是递减排列，就可能产生连续的合并，并且也容易避免两个可以合并的砖块被某个更大的砖块隔开的情况。另外，对于不可得到新局面的操作，其得分被置为零；因此，为了将其与有效操作做区分，有效操作的得分会被加上一个基础得分`FixedScore`。（这点其实很重要，否则后期操作有限的情况下你的AI可能会选择不停地做某个无效操作，使局面卡住）。

初始局面由于空格较多，因此枚举的情况较多，AI速度较慢。玩到后来就比较快了。

AI的测试效果还不错，~~甚至比本人玩得还好~~。少量测试了几次，每次AI都可以玩到1024往上，最高得分达到了25712分。一个AI玩到了2048的视频如下（上传到了云盘，可以在线看或是下载），该视频是从AI获得了1024开始，直到游戏结束。

AI视频

4.作业中遇到的问题以及收获

作业中遇到的问题主要出在对**线程**的使用。

1. 在更新游戏界面时，初始想要另开一个线程中，在这个线程中让其每隔一段时间更新一下页面。但是更新页面的`repaint`函数总是不工作。后来了解到`repaint`并不具有强制即时渲染的功能，只是**尽快**渲染。在网上搜索解决方法长时间没有效果，后来尝试使用计时器，每次计时结束就`repaint`页面，收到了奇效。并且，游戏界面几乎完全不卡顿，操作很快时仍然运行流畅。
2. AI模式中，由于AI需要持续的、每隔一段时间就做出一个决策，因此初始考虑也是另开一个线程。但是，实际运行过程中发现该线程的堵塞似乎会造成绘画的线程不工作。这样，切换至AI模式后页面就不会变化。受到1中的启发，也换成了计时器，效果依然很好。但是特别需要注意的是，由于AI做出决策和游戏界面重新渲染是两个不同的线程的任务，而二者共享一部分数据，因此要保证二者不会互相影响。考虑了一下，由于共享的数据较多，分散程度较大，进程锁不太方便实现。于是就简单地在AI决策线程开头判断一下当前绘画是否完成，如果没完成，那么AI的本次决策周期作废，等待计时器下一次归零。

本人来自计算机系，虽然在小学期时曾经用Qt写过一个军棋游戏，以及自己课余时间也实现过一个可以网络对战的五子棋游戏，但是在本次做动画以及写AI的过程中还是有一定的收获。（当初为什么要选这个课，主要是自己对Java一直有所耳闻但是没有接触过，加上计算机系取消了大二暑假的Java课程，我想着自己课余估计也很难抽出时间系统性地学习Java，而且cwj老师风评很不错，就决定报这门课好好学一下Java。~~其实我也是Java零基础来选课的~~）

5.参考资料

1. 游戏图片素材以及UI设计参考自网站https://2048game.com/。
2. AI的算法参考自https://www.baeldung.com/cs/2048-algorithm上的伪代码，自己设置了各种超参数进行实验，用Java进行了实现和一些改进。
3. https://stackoverflow.com/上的各类问题解答，比较多就不一一列举了。