Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

«Технологии программирования (Java)»

Отчет по курсовой работе
Игра 2048 на Android

Работу выполнил:
Симоновский Д. Л.
Группа:
3530901/10001
Преподаватель:
Алексюк А. О.

Санкт-Петербург
2022

ЗАДАНИЕ
НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

Студенту группы 3530901/10001 (Номер группы) Симоновскому Даниилу Леонидовичу (Фамилия, Имя, Отчество)

1. **Тема проекта:** «2048»
2. **Срок сдачи законченного проекта:** 21.05.2022
3. **Исходные данные к проекту:** IDE: Android Studio Chipmunk 2021.2.1
4. **Содержание пояснительной записки** (перечень подлежащих разработке вопросов): введение, основная часть (текст программы, описание программы, испытания программы), заключение, список использованных источников.

Дата получения задания: «9» апреля 2022 г.

Руководитель — Алексюк А.О.
Задание принял к исполнению — Симоновский Д.Л.
«9» апреля 2022 г.

Содержание

1. Цель работы

Создать и протестировать игру 2048 на Android

2. Правила игры

2048 - игра для одного человка. В ней путём свайпов (в оригинале нажатий на стрелочки) игрок может скинуть все плитки игрового поля в одну из 4 сторон. Если при сбрасывании две плитки одного номинала «налетают» одна на другую, то они превращаются в одну, номинал которой равен сумме соединившихся плиток. После каждого хода на свободной секции поля появляется новая плитка номиналом «2» (90%) или «4» (10%). Если при нажатии кнопки местоположение плиток или их номинал не изменится, то ход не совершается.

Если в одной строчке или в одном столбце находится более двух плиток одного номинала, то при сбрасывании они начинают соединяться с той стороны, в которую были направлены. Например, находящиеся в одной строке плитки (4, 4, 4) после хода влево превратятся в (8, 4), а после хода вправо — в (4, 8). Данная обработка неоднозначности позволяет более точно формировать стратегию игры.

За каждое соединение игровые очки увеличиваются на номинал получившейся плитки.

Игра заканчивается поражением, если после очередного хода невозможно совершить действие.

Игра заканчивается победой, если была достигнута клетка наминалом 2048

3. Описания решения

Приложение разделено на 2 пакета: *«core»* и *«ui»*. Изначально планировалось использовать модель «Model-View-Controller», однако было принятно решение от неё отказаться в пользу упрощения чтения кода т.к. часть «Controller» добавляла слишком много лишней смысловой нагрузки.

3.1. core

Пакет «core» описывает саму логику работы игры, все перемещения, объединения цифр обрабатываются именно там. Пакет «core» содержит:

- Публичный enum **«Direction»**. Используется для передачи в основную часть информацию о направлении передвижения плиток.
- Package-private class **«Vector»**. Используется для преобразования из не очень очевидного «Direction» в конкретные цифры перемещения по x и y
- Публичный class **«Coordinate»**. Представляет из себя координаты на поле определенной клетки. Имеет в себе функции для взаимодействий с классом «Vector» (в частности сложение)
- Публичный вложенный в «Coordinate» static class **«Move»**. Используется для того, чтоб передать перемещение фигуры из точки *from* в точку *to*, необходим для дальнейшей анимации.

- Публичный class **«Game»**. В этом классе происходит вся логика предложения. Он содержит следующие публичные методы:

  doMove(Direction) - совершает само перемещение фигур в направлении, переданном в Direction. Возвращает List из Move т.е. перемещения клеток

  spawnSquare() - создает на поле новую фигуру (90% - 2, 10% - 4) и возвращает Pair из Coordinate с местом, где появилась и фигура, и цифры на фигре. Или null, в случае, если места на поле нет.

  gameIsLost() - возвращает информацию о том, проиграна ли игра.

  getScore() - возвращает текущий счет игры.

3.2. ui

Пакет «ui» является смесью из View и Controller, в нем происходит отображение всего происходящего и обработка нажатий кнопок и свайпов. Этот пакет содержит:

- Публичный class **«OnSwipeTouchListener»**. Этот класс был взят из открытых источников и используется для получения направления свайпов.

- Публичный class **«Board»**. Этот класс наследуется от View и представляет собой поле, отображаемое на экане телефона. В нем присутсвует метод для преобразования из координаты, полученной из core в координату на экране телефона, для дальнейшей отрисовки.

- Публичный class **«Square»**. Этот класс наследуется от View и представляет собой плитку, отображаемую на экране телефона. Меняет цветплитки и цвет шрифта в зависимоти от цифры на плитке (цвета взяты из открытого репозитория создателя игры).

- Публичный class **«GameActivity»**. В этом классе происходит всё, что отвечает за визуальную часть игры:

  onCreate() - метод, вызываемый при открытие приложения. В началае устанавливается xml файл (activity_game.xml), который хранит расположение всех элементов на экране. После этого я по id получаю доступ к элементам с экрана, которые меня интересуют. Далее присходит настройка кнопок: я добавляю каждой кнопке слушателя, чтоб при её нажатии что-то происходило (кнопка quit - выход, кнопка restart - перезапуск), добавляет слушателя на объект swipeDetector, чтоб вызывать функцию doIteration в направлении свайпа. После чего вызывается метод старт.

  start() - метод для начала игры. Устанавливает таймер на 0 и запускает его, устанавливает счет на 0, пересоздает доску, создает первую плику.

  doIteration() - метод, вызываемый при свайпе на экране. С анимацей перемещает все плитки в направлении свайпа, потом с анимацией объединяет необходимые. После всего этого спавнит новую плитку, вызывает обновление счёта и проверяет игру на победу/поражение (при необходимости вызывает метод endGame).

  endGame(String) - останавливает таймер, выводит на экран текст, переданный в функцию.

  restart() - метод, вызываемый при нажатии кнопки restart, делает приготовления, после чего вызывает метод start

4. Внешний вид приложения

На данных скриншотах видно пустое поле и поле с большим числом плиток разного цвета, а также меню проигрыша:

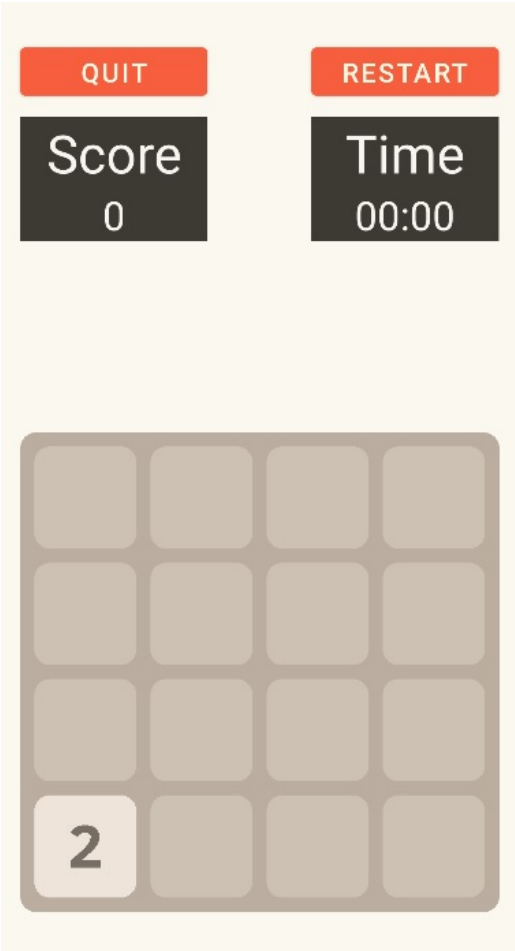

Рисунок 4.1. Пустое поле

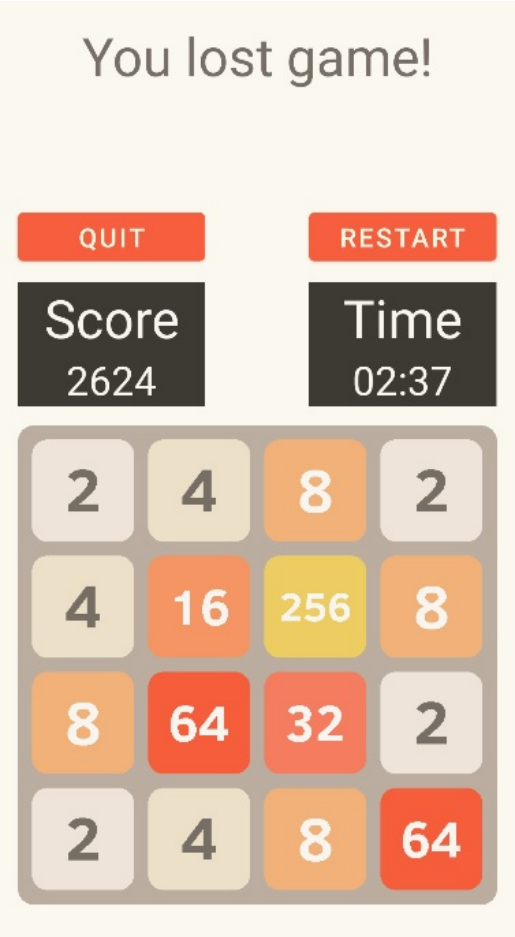

Рисунок 4.2. Поле с плитками

5. Тестирование

Были сделаны Unit тесты для core части в количестве 10 штук, что дает 90% покрытия. Присутсвуют как и обычные тесты, так и те, которые возникли в процессе дебагинга. Смысла отдельно рассматривать каждый тест нет, для подробного изучения см. репозиторий на гитхаб.

6. Выводы

В ходе проделанной работы была создана игра 2048 на Android, которая ничуть не уступает оригиналу. Были разработаны тесты, которые проверяют правильность работы кода.

Исходный код проекта можно найти по ссылке на GitHub

7. Источники

- github.com - репозиторий с исходным кодом оригинального 2048
- wikipedia.org - описание правил игры 2048