МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Разработка приложений для мобильных платформ»

Тема: Простая игра про раскраску карт N цветами.

Студент гр. 1303 _____ Бутыло Е.А.

Студент гр. 1303 _____ Депрейс А.С.

Студентка гр. 1303 _____ Андреева Е.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2024

ЗАДАНИЕ

Студенты

Бутыло Е.А.

Депрейс А.С.

Андреева Е.А.

Группа 1303

Тема работы: Простая игра про раскраску карт N цветами

Исходные данные:

1. Портирование игры:
   https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/map.html
2. Требования:
   Разные уровни сложности
   Статистика
   Пауза

Содержание пояснительной записки:

Введение

Сценарий использования

Пользовательский интерфейс

Разработанное приложение

Выводы

Приложение

Литература

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 03.02.2025

Дата сдачи реферата: 16.03.2025

Дата защиты реферата: 16.03.2025

Студент гр. 1303		Бутыло Е.А.
Студент гр. 1303		Депрейс А.С.
Студентка гр. 1303		Андреева Е.А.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В ходе выполнения курсовой работы разработано мобильное приложение для раскраски карт N цветами, созданное в рамках дисциплины «Разработка приложений для мобильных платформ». Приложение предоставляет пользователю возможность решать задачи раскраски графов, соблюдая установленные ограничения. Для разработки использовались Android Studio, язык программирования Kotlin и toolkit Jetpack Compose. Реализован удобный пользовательский интерфейс с интерактивной визуализацией карт и возможностью выбора цветов. В результате создана интуитивно понятная и увлекательная игра, позволяющая пользователям развивать логическое мышление и знакомиться с основами теории графов.

SUMMARY

As part of the project, a mobile application for coloring maps with N colors was developed within the discipline "Mobile Application Development". The application allows users to solve graph coloring problems while adhering to predefined constraints. The development utilized technologies, including Android Studio, the Kotlin programming language, and the Jetpack Compose toolkit. A user-friendly interface was implemented with interactive map visualization and color selection capabilities. As a result, an intuitive and engaging game was created, enabling users to develop logical thinking and become familiar with the basics of graph theory.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность решаемой проблемы

Задача раскраски карт является одной из фундаментальных проблем дискретной математики и теории графов. Разработка мобильного приложения для решения этой задачи позволяет не только создавать увлекательный игровой процесс, но и повышать интерес пользователей к логическому мышлению и алгоритмическим задачам.

Постановка задачи

Целью проекта является разработка мобильного приложения, позволяющего пользователям решать задачи раскраски карт N цветами в соответствии с правилами теории графов.

Предлагаемое решение

Создание мобильного приложения на основе:

- Kotlin для разработки логики приложения
- Jetpack Compose для создания современного, удобного и интерактивного пользовательского интерфейса.
- Android Studio для управления процессом разработки и отладки приложения

Качественные требования к решению:

- Разные уровни сложности
- Статистика
- Пауза

1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

1. **Выбор уровня**

   Действующие лица:

   - Пользователь.

   Предусловия:

   - Пользователь открыл приложение и находится на главном экране (экран 1).

   Основной сценарий:

   - Пользователь выбирает уровень сложности игры.

   Результат основного сценария:

   - Система перенаправляет пользователя на экран с игрой (экран 2) выбранного уровня сложности и запускает игру (секундомер).

2. **Пауза**

   Действующие лица:

   - Пользователь.

   Предусловия:

   - Пользователь находится на экране с действующей игрой (экран 2).

   Основной сценарий:

   - Пользователь нажимает кнопку паузы.

   Результат основного сценария:

   - Система перенаправляет пользователя на экран с паузой (экран 3) и останавливает секундомер.

3. **Возобновление игры**

   Действующие лица:

   - Пользователь.

   Предусловия:

   - Пользователь находится на экране с паузой (экран 3).

   Основной сценарий:

   - Пользователь нажимает кнопку "Resume".

   Результат основного сценария:

- Система перенаправляет пользователя на экран с действующей игрой (экран 2) и запускает секундомер.

4. **Выход из игры в меню**

   Действующие лица:

   - Пользователь.

   Предусловия:

   - Пользователь находится на экране с паузой (экран 3).

   Основной сценарий:

   - Пользователь нажимает кнопку "Menu".

   Результат основного сценария:

   - Система перенаправляет пользователя на главный экран (экран 1) и сбрасывает прогресс текущей игры.

5. **Просмотр статистики**

   Действующие лица:

   - Пользователь.

   Предусловия:

   - Пользователь находится на главном экране (экран 1).

   Основной сценарий:

   - Пользователь нажимает кнопку "Statistics".

   Результат основного сценария:

   - Система перенаправляет пользователя на экран со статистикой.

6. **Выход из статистики в меню**

   Действующие лица:

   - Пользователь.

   Предусловия:

   - Пользователь находится на экране со статистикой (экран 4).

   Основной сценарий:

   - Пользователь нажимает кнопку "Menu".

   Результат основного сценария:

   - Система перенаправляет пользователя на главный экран (экран 1).

7. **Кастомизация уровней**

Действующие лица:

- Пользователь.

Предусловия:

- Пользователь находится на главном экране (экран 1).

Основной сценарий:

- Пользователь нажимает на кнопку кастомизации.

- Пользователь выбирает количество полигонов.

Результат основного сценария:

- Система перенаправляет пользователя на экран кастомизации (экран 5).

- Система устанавливает выбранное количество полигонов для всех уровней.

8. **Выход из кастомизации в меню**

Действующие лица:

- Пользователь.

Предусловия:

- Пользователь находится на экране с кастомизацией (экран 5).

Основной сценарий:

- Пользователь нажимает кнопку возврата в меню.

Результат основного сценария:

- Система перенаправляет пользователя на главный экран (экран 1).

9. **Выбор цвета для раскраски**

Действующие лица:

- Пользователь.

Предусловия:

- Пользователь находится на экране с действующей игрой (экран 2).

Основной сценарий:

- Пользователь нажимает круглую кнопку с пипеткой.

- Пользователь нажимает на уже закрашенную область и выбирает этот цвет.

Результат основного сценария:

- Система отображает выбранный цвет как текущий (кнопка окрашивается в этот цвет).

10. Окрашивание области

Действующие лица:

- Пользователь.

Предусловия:

- Пользователь находится на экране с действующей игрой (экран 2).

Основной сценарий:

- Пользователь нажимает на любую область на игровом поле для закраски.

Результат основного сценария:

- Система окрашивает область в текущий выбранный цвет.

Альтернативный сценарий:

- Пользователь нажимает на стартовую окрашенную область.

Результат альтернативного сценария:

- Ничего не меняется.

11. Подсказка

Действующие лица:

- Пользователь.

Предусловия:

- Пользователь находится на экране с действующей игрой (экран 2).

Основной сценарий:

- Пользователь нажимает на кнопку подсказки.

Результат основного сценария:

- Система окрашивает область в правильный цвет.
- Система добавляет к таймеру 10 секунд.

2. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

Макет UI (см. Рисунок 1)

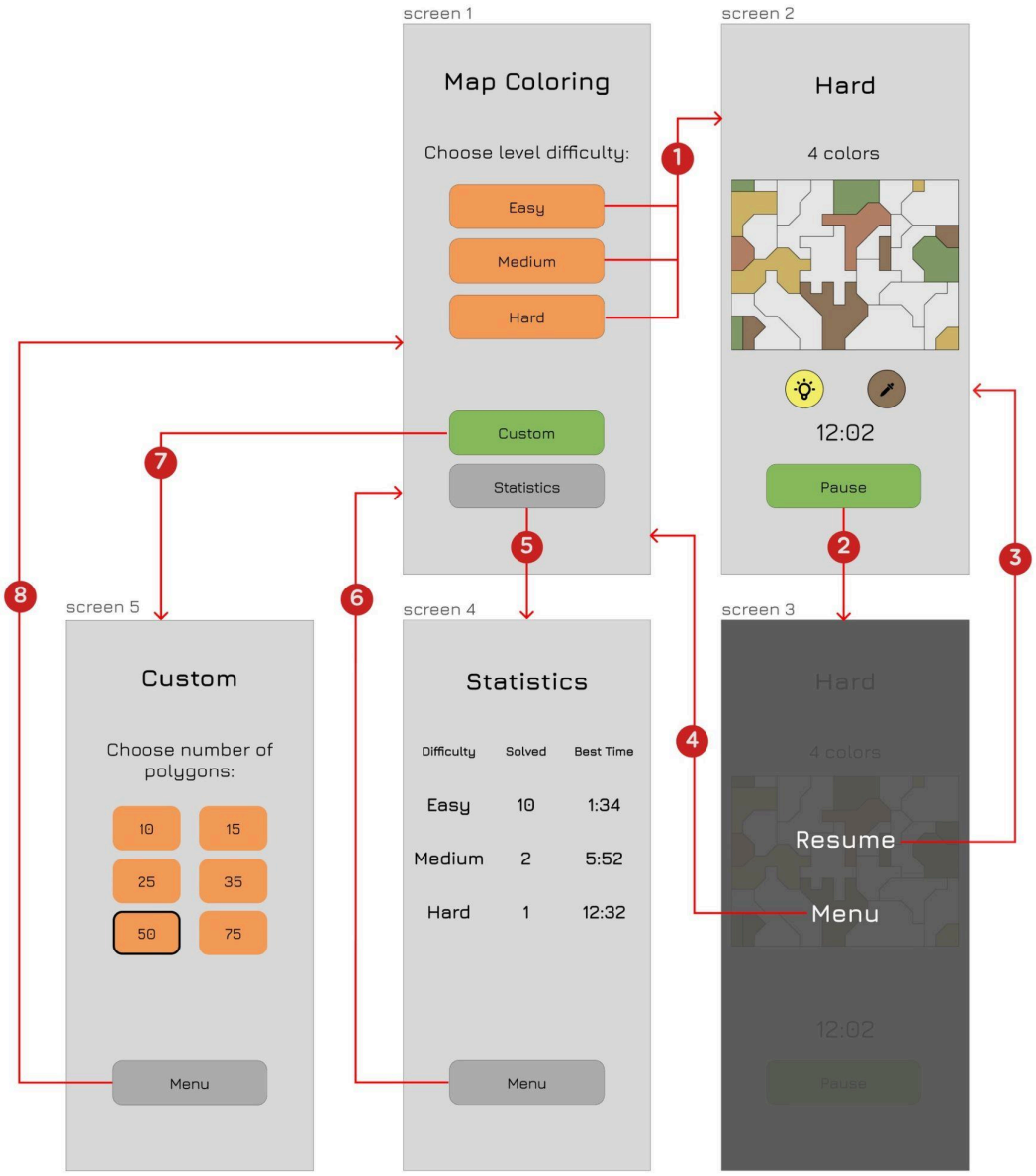

Рисунок 1 - Макет UI

Целевые устройства

Телефоны, модель: Google Pixel 9 Pro

Аппаратная составляющая

Версия API 35

Экран**:**

hw.lcd.density 480

hw.lcd.height 2856

hw.lcd.width 1280

Остальное**:**

avd.ini.displayname Pixel 9 Pro API 35

avd.ini.encoding UTF-8

AvdId Pixel_9_Pro_API_35

disk.dataPartition.size 2G

fastboot.chosenSnapshotFile

fastboot.forceChosenSnapshotBoot no

fastboot.forceColdBoot no

fastboot.forceFastBoot yes

hw.accelerometer yes

hw.arc false

hw.audioInput yes

hw.battery yes

hw.camera.back virtualscene

hw.camera.front emulated

hw.cpu.ncore 2

hw.device.hash2 MD5:73e7b35d09e3a8055043aca4688e0dad

hw.device.manufacturer Google

hw.device.name pixel_9_pro

hw.dPad no

hw.gps yes

hw.gpu.enabled yes

hw.gpu.mode auto
hw.initialOrientation portrait
hw.keyboard yes
hw.mainKeys no
hw.ramSize 11548
hw.sdCard yes
hw.sensors.orientation yes
hw.sensors.proximity yes
hw.trackBall no
image.androidVersion.api 35
image.sysdir.1
system-images/android-35/google_apis_playstore/x86_64/
PlayStore.enabled true
runtime.network.latency none
runtime.network.speed full
showDeviceFrame yes
skin.dynamic yes
tag.display Google Play
tag.displaynames Google Play
tag.id google_apis_playstore
tag.ids google_apis_playstore
vm.heapSize 256

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Краткое описание

Разработанное мобильное приложение представляет собой интерактивную игру, в которой пользователю необходимо раскрасить карту N цветами, соблюдая правило: соседние области не должны иметь одинаковый цвет. Игра основана на принципах теории графов и предлагает различные уровни сложности.

Приложение разработано на языке Kotlin с использованием Jetpack Compose для создания современного и удобного интерфейса. В нем предусмотрены механизмы проверки корректности раскраски, система подсказок, а также возможность генерации случайных карт.

Ознакомиться с работой реализованного приложения можно на рисунках 2-8 (см. раздел Приложения, Рисунки 2-8).

Схема архитектуры

Архитектура приложения построена на локальном хранении данных и использовании современных инструментов разработки для Android. Все вычисления и обработка данных происходят на устройстве пользователя без необходимости подключения к серверу.

Основные компоненты архитектуры:

1. Логика приложения: для реализации основной логики раскраски был создан класс “Полигон”, который отвечает за окрашиваемую фигуру. Он хранит текущий цвет, соседей и клетки, на которых расположен полигон. Взаимодействие с картой полигонов изменяется через класс “Карта полигонов”, можно попытаться изменить цвет полигона по относительным координатам, или взять цвет полигона по текущим координатам, также проверяется завершенность раскраски.
2. Хранилище данных: используется DataStore (Preferences Storage) для сохранения локальной информации, такой как: выбранное количество полигонов для генерации карты и игровая статистика (количество

успешно завершенных уровней, рекордное время прохождения). DataStore предпочтителен для небольших объемов данных, так как он эффективен, безопасен и интегрируется с современными Android API.

3. Генерация карт: карты создаются с использованием алгоритма, который случайно объединяет полигоны, пока на карте не останется заданное их количество. Изначально полигон имеет одну клетку карты. Алгоритм обеспечивает корректное распределение полигонов, чтобы они соответствовали правилам раскраски.

Использованные технологии (внешние)

В разработке приложения не используются сторонние технологии, библиотеки или внешние API. Вся логика и функциональность реализованы средствами Kotlin и Jetpack Compose, а для хранения данных применяется встроенный механизм DataStore (Preferences Storage).

Использованные модули/системные библиотеки вашей платформы

При разработке приложения использованы стандартные системные библиотеки и модули платформы Android, обеспечивающие удобную работу с UI, навигацией, управлением состоянием и хранением данных.

Основные используемые модули:

1. Jetpack Compose:

- androidx.activity.compose – интеграция Compose с управлением жизненным циклом Activity.
- androidx.foundation – базовые UI-компоненты и жесты.
- androidx.ui.v150 – UI-компоненты и инструменты для работы с Compose.
- androidx.material3 – современный Material Design 3 для стилизации приложения.
- androidx.runtime – управление состоянием в Compose.
- androidx.ui.tooling.preview.android – инструменты для предпросмотра UI в Android Studio.

2. Основные библиотеки AndroidX:

- androidx.core.ktx – расширения для упрощенной работы с Kotlin.
- androidx.appcompat – поддержка совместимости со старыми версиями Android.
- androidx.activity – управление жизненным циклом Activity.
- androidx.constraintlayout – работа с ConstraintLayout.

3. Жизненный цикл и навигация:

- androidx.lifecycle.runtime.ktx – управление жизненным циклом компонентов.
- androidx.navigation.compose – навигация в приложении с Jetpack Compose.

4. Хранение данных:

- androidx.datastore.preferences – локальное хранилище данных с использованием DataStore.

4. ВЫВОДЫ

Достигнутые результаты

- Разработано мобильное приложение для раскраски карт N цветами.
- Реализована интерактивная игровая механика с автоматической проверкой корректности раскраски.
- Создан удобный и адаптивный интерфейс с использованием Jetpack Compose.
- Обеспечено локальное хранение данных (предпочтений и статистики) с помощью DataStore Preferences Storage.

Недостатки и пути улучшения

- Полигоны для раскраски имеют примитивную форму в виде набора квадратов.
- Нет обучающего режима для пользователей.

Будущее развитие решения

- Добавление алгоритма генерации полигонов со сложной формой.
- Реализация обучающего режима.
- Добавление достижений.
- Оптимизация производительности для работы на более широком спектре устройств.
- Возможность сохранения и загрузки предыдущих игр.

Разработанное приложение является стабильным, автономным и удобным инструментом для решения задач раскраски карт, а его дальнейшее развитие позволит расширить функционал и повысить интерес пользователей.

5. СПИСОК ЛИТЕРАТУРЫ

1. Kotlin Docs. Official page – https://kotlinlang.org/docs/home.html
2. Jetpack Compose Documentation. Official page – https://developer.android.com/develop/ui/compose/documentation?hl=ru
3. Игра про раскраску карт. Official page - https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/map.html
4. Курс “Android basics with compose” для разработчиков Android – https://developer.android.com/courses/android-basics-compose/unit-1
5. Репозиторий с разработанным приложением – https://github.com/moevm/adfmp1h25-map

6. ПРИЛОЖЕНИЯ

Запуск приложения

Для запуска приложения на устройстве или эмуляторе необходимо выполнить следующие шаги:

1. Убедитесь, что установлена последняя версия Android Studio. Проверьте, что в проекте настроен Gradle и все зависимости загружены корректно.
2. Откройте проект в Android Studio. Подключите физическое устройство (с включенной отладкой по USB) или запустите эмулятор Android. Выберите целевое устройство и нажмите Run. Дождитесь завершения сборки и развертывания приложения.
3. После установки приложение откроется на главном экране. Пользователь может выбрать уровень сложности игры, нажав на соответствующую кнопку. После нажатия, автоматически откроется поле игры и запустится таймер.
4. Для паузы в игре необходимо нажать на кнопку “Pause”, таймер приостановиться и появится две кнопки “Menu” и “Resume” для перехода в меню и для возобновления игры соответственно.
5. Для закрашивания области необходимо нажать на кнопку с пипеткой, а затем на цвет, который уже присутствует на поле, а затем закрасить пустую область.
6. Для использования подсказки необходимо нажать кнопку с лампочкой, при этом закрасится пустой полигон, а также прибавится 10 секунд к таймеру.
7. Для изменения количества полигонов следует нажать на кнопку “Custom”, а затем выбрать нужное количество, возврат в меню осуществляется через кнопку “Menu”.
8. Для просмотра статистики пользователя следует нажать на кнопку “Statistics”, где отображается количество пройденных уровней и лучшее

время на каждую сложность, возврат в меню осуществляется через кнопку “Menu”.

Снимки экрана приложения

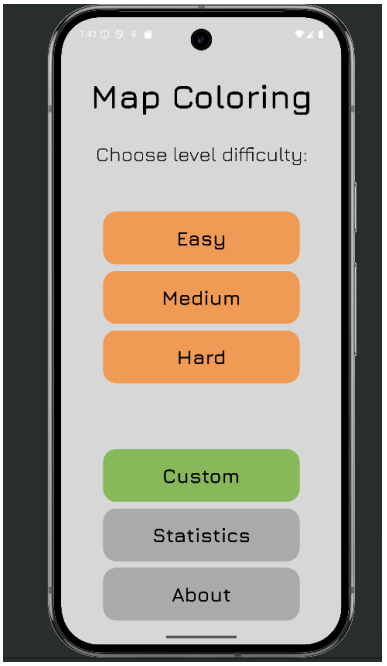

Рисунок 2 - Меню

Рисунок 3 - Выбор количества полигонов

Рисунок 4 - Просмотр статистики

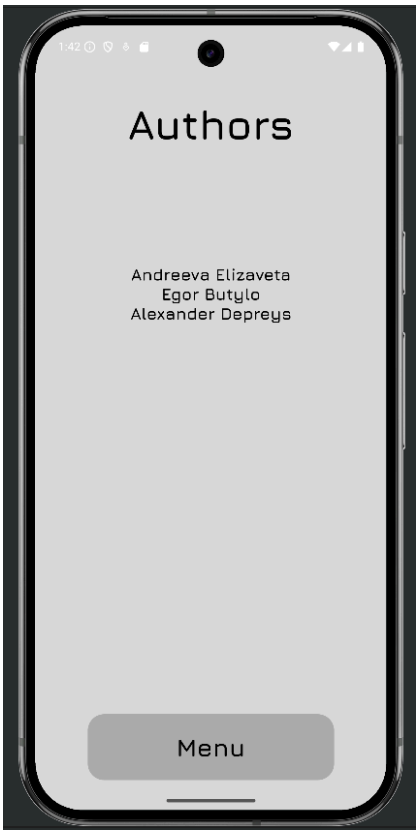

Рисунок 5 - Страница авторов

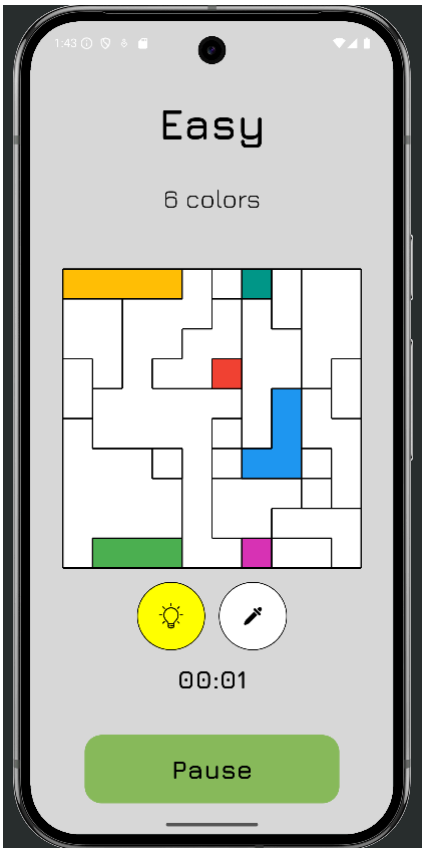

Рисунок 6 - Легкий режим с 6 цветами

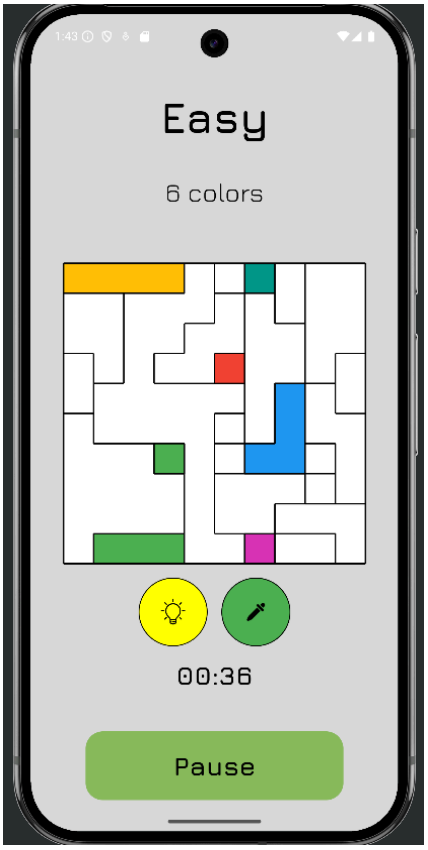

Рисунок 7 - Легкий режим с 6 цветами, выбран зеленый цвет

Рисунок 8 - Окно паузы