МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Разработка приложений для мобильных платформ»

Тема: «Беговой GPS трекер»

Студент гр. 0303 Скиба В. В.

Преподаватель Заславский М.М.

Санкт-Петербург

2024

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Идея создания данного приложения появилась после использования существующий решений: приложений для записи статистики и результатов беговой тренировки на Android. Среди рассмотренных приложений были те, которые дополнительно составляли статистику рекордов на фиксированные дистанции: рассматривались все записи тренировок и сравнивалось время первого километра, 500м, 250м, 2км, и т.д. Таким образом, эта статистика выступала в роли глобальной статистики физической подготовки пользователя, с мотивацией побить рекорды.

Идея рекордов на фиксированные дистанции показалась мне очень мотивирующей, но существующие реализации брали в расчет только самое начало дистанции общей тренировки, игнорируя потенциально более низкое время в остальной части тренировки. Помимо данного недостатка, алгоритмы фильтрации данных казались реализованы недостаточно качественно: иногда сопоставление записанного маршрута и подсчет дистанции отличался от фактической, что приводит к некорректным статистическим результатам.

Я в роли целевого пользователя приложения хотел бы иметь более высокий фактор доверия к рекордам, которые я устанавливаю, поэтому было принято решение создания своего приложения, в котором точности статистических данных было бы уделено больше внимания.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Запуск тренировки. Для начала тренировки пользователь нажимает кнопку старт, после чего происходит проверка разрешений на местоположение. Если местоположение не определено, выводится сообщение об ошибке с дальнейшими действиями. Если разрешение на местоположение было предоставлено, отрывается новый экран с текущей тренировкой.

Текущая тренировка. На экране тренировки отображается информация о текущей тренировке: пройденная дистанция, время тренировки, текущая и средняя скорость. Экран интерактивно обновляет анимации в соответствии с текущей скоростью.

Приостановка тренировки. При нажатии кноки назад на экране тренировки, тренировка приостанавливается. У пользователя есть выбор: либо завершить тренировку, либо продолжить.

Просмотр записей прошлых тренировок. Записи прошлых тренировок можно посмотреть на отдельном экране.

Просмотр общей статистики за все время. Общую статистику можно посмотреть на отдельном экране.

3. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

3.1. Макет интерфейса с графом переходов

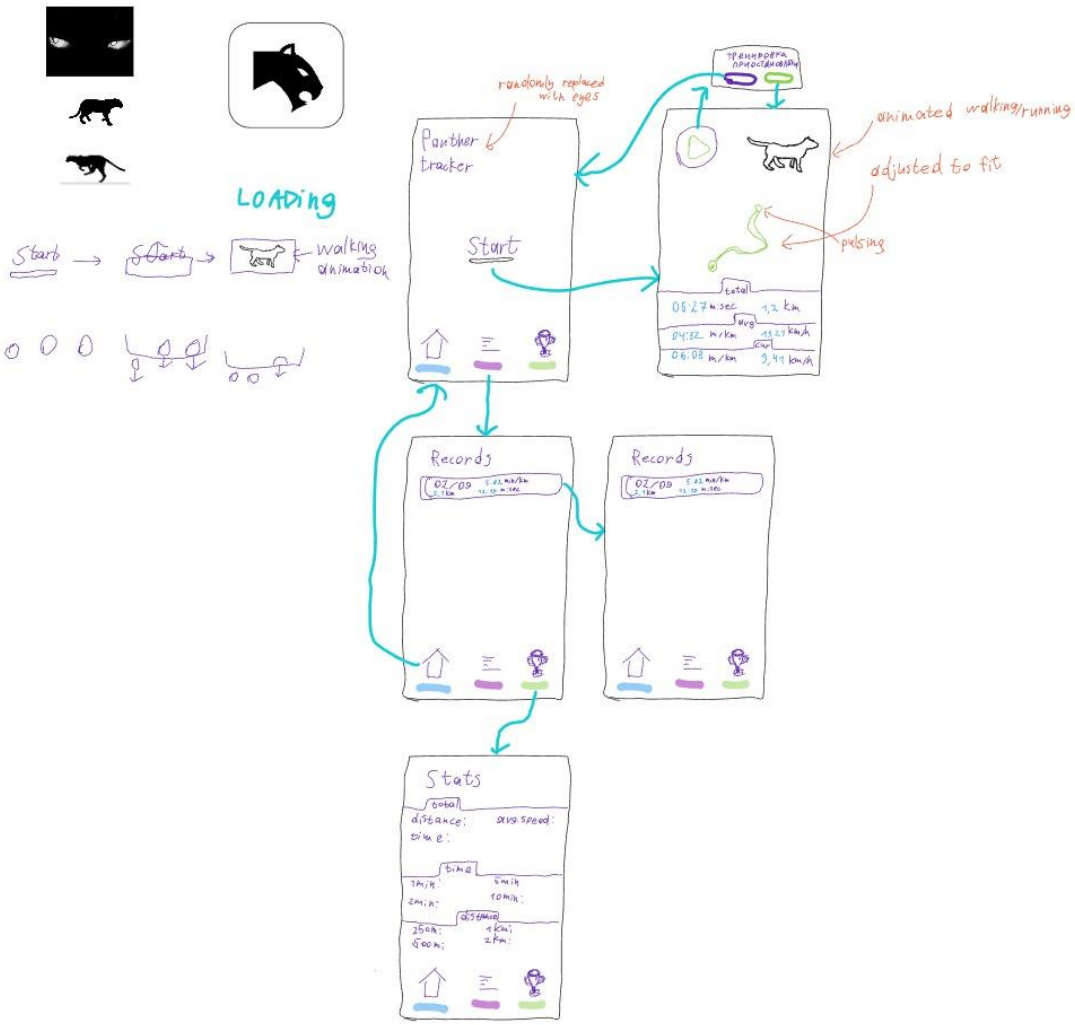

Рисунок 1 – Макет UI для пользователя приложения.

3.2. Целевые устройства, обоснование требований и максимально подробные характеристики

Тип устройств - смартфоны

Характеристики:

Экран

- hw.lcd.density 560

- hw.lcd.height 3120
- hw.lcd.width 1440

Прочее

- image.androidVersion.api 30
- avd.ini.displayname Pixel 6 Pro API 30
- avd.ini.encoding UTF-8
- AvdId Pixel_6_Pro_API_30
- 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:a8abfd3536f3d35e4ba2041a7b99f40e
- hw.device.manufacturer Google
- hw.device.name pixel_6_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 1536
- hw.sdCard yes
- hw.sensors.orientation yes
- hw.sensors.proximity yes
- hw.trackBall no
- image.sysdir.1 system-images/android-30/google_apis/x86/
- PlayStore.enabled false

- runtime.network.latency none
- runtime.network.speed full
- showDeviceFrame yes
- skin.dynamic yes
- tag.display Google APIs
- tag.id google_apis
- vm.heapSize 384

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Вместо классического использования Jetpack Compose и Java/Kotlin было принято решение разработки мини-фреймворка на нативном языке (Rust) для отображения элементов, обработки логики приложения (обработка и реакция на события), заполнения контента экранов, управления жизненным циклом экранов. Приложение состоит из единственного NativeActivity, который подключает нативную библиотеку и выполняет отрисовку и обработку всей логики приложения.

В некоторых ситуациях (предоставление разрешения на местоположение и запрос текущего местоположения) использования функциональности возможно только из кода Java. В этих случаях был создан класс LocationHelper, который связывает критически необходимые части кода на Java с остальным кодом на Rust.

4.2. Снимки экрана приложения

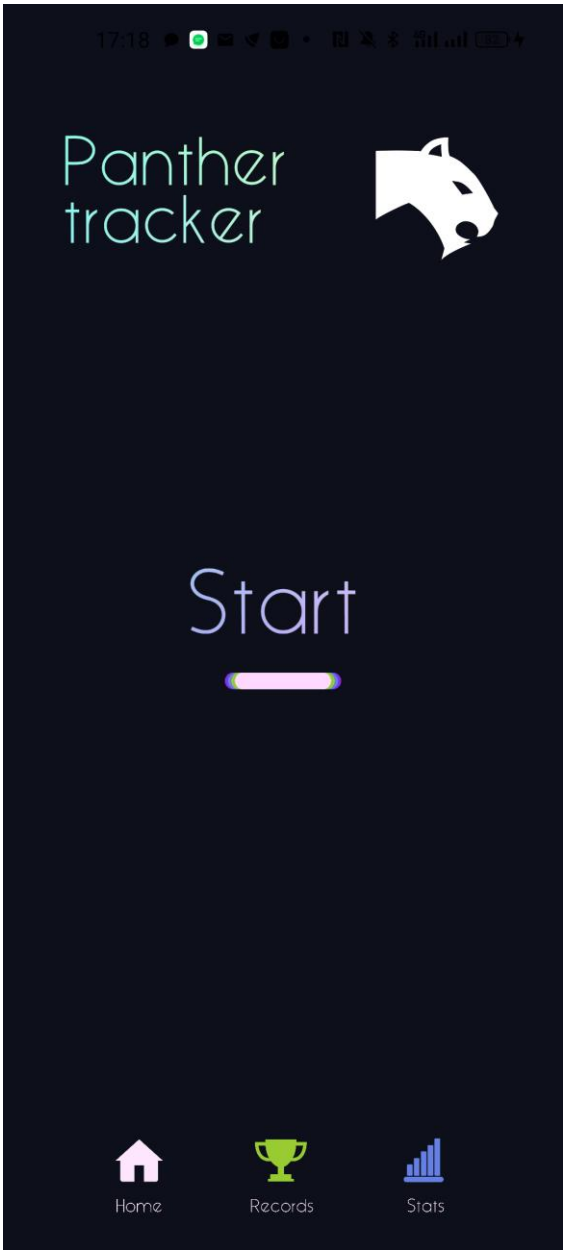

Рисунок 2. Главный экран приложения

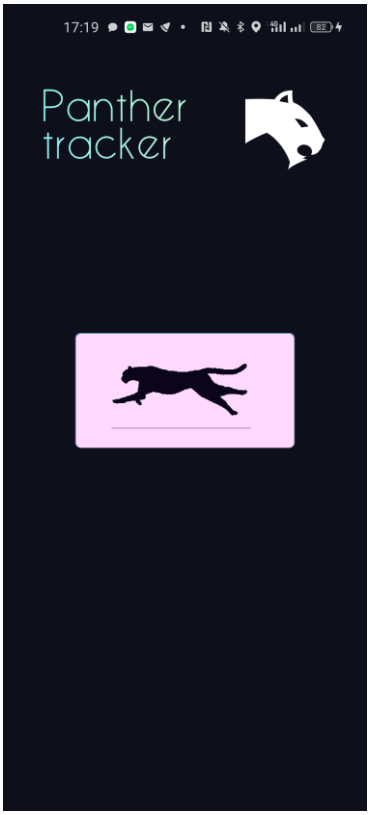

Рисунок 3. Начальная загрузка тренировки

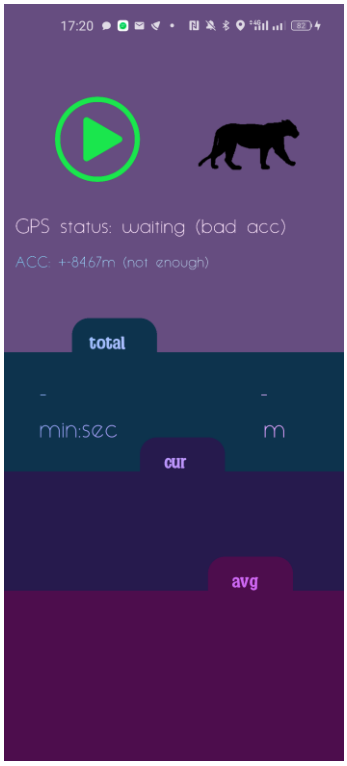

Рисунок 4. Экран тренировки при условии недостаточной точности данных GPS.

Рисунок 5. Приостановленная тренировка

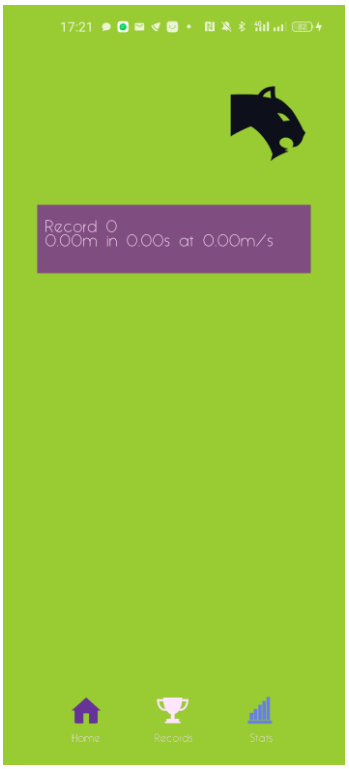

Рисунок 6. Список записей тренировок.

Рисунок 7. Общая статистика профиля.

5. ВЫВОД

5.1. Достигнутые результаты

В ходе работы был разработан графический фреймворк, предоставляющий базовую функциональность для создания приложений с интерфейсами, с особым акцентом на графические эффекты и анимации. Фреймворк позволяет создавать кастомные элементы с указанием собственного фрагментного шейдера, что дает возможности создания уникальных компонентов со своей логикой отображения.

Взаимодействие с приложением реализовано в фреймворке в виде ивентов касания (touch), перемещения (scroll), нажатия кнопки назад. Наличие этих фундаментальных событий позволило обойтись без переусложнения API для пользователей библиотеки, но при этом этих возможностей достаточно для создания довольно широкого класса различных приложений.

В качестве альтернативы механизму Activity в андроид были использованы экраны (Screens), которые функционируют похожим образом, но со встроенной возможностью управления анимациями переключения экранов.

После разработки фреймворка удалось относительно быстро создать главный экран и экран тренировки (по сравнению с вариантом без фреймворка), и заполнить их логикой работы приложения и анимациями, а также минимально реализовать остальные экраны приложения.

Для получения данных GPS были использованы функции, реализованные на java, с колбэками в нативном коде, который дальше обрабатывал состояние тренировки.

В разработанном приложении с использованием реализованного фреймворка можно записать полноценную тренировку и посмотреть

данные пройденной дистанции, времени и средней скорости. Так же общие статистические данные можно посмотреть на экране Stats.

5.2. Недостатки и пути для улучшения полученного решения

Из недостатков подхода к разработке приложения можно считать более слабую интеграцию с системой. Однако, по мере развития фреймворка, можно постепенно добавлять критически важную функциональность, в результате чего разработка мобильного приложения окажется на уровне удобства с разработкой на языке Kotlin, но гораздо производительнее, менее привязана к устройству и платформе, и с бесконечными потенциальными возможностями развития.

5.3. Будущее развитие решения

Планируется развитие фреймворка интерфейсов и отвязка от конкретной платформы. Потенциальный список платформ: Windows, Linux, MacOS, IOS, Android, VR/XR (OpenXR). Создание кроссплатфоменного решения не является проблемой в Rust, так как основная зависимость это графическая библиотека.

Также, планируется переход с OpenGL на гораздо более интересный в плане производительности и возможностей Vulkan.

6. ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

1. Описание библиотек языка Rust: https://docs.rs/

2. Документация OpenglES: https://registry.khronos.org/OpenGL-Refpages/es3.0/

3. Android API Reference: https://developer.android.com/reference

7. ПРИЛОЖЕНИЕ

Документация по сборке и развертыванию приложения (precompiled)

1. Склонировать репозиторий https://github.com/moevm/adfmp1h24-gps
2. Открыть директорию panther_tracker в Android Studio
3. Запустить его, нажав Menu > Run > Run ‘app’

Документация по сборке и развертыванию приложения (полная сборка) - Linux

1. Необходима установка rust toolchain на ваш компьютер в соответствии с https://rustup.rs/
2. Установка необходимых иснтрументов для Rust:

```
cargo install cargo-ndk
rustup target add \
   aarch64-linux-android \
   armv7-linux-androideabi \
   x86_64-linux-android \
   i686-linux-android
```

3. Склонировать репозиторий https://github.com/moevm/adfmp1h24-gps
4. Открыть директорию panther_tracker в Android Studio
5. Установить NDK в Android Studio: NDK: Settings > Languages and frameworks > Android SDK > SDK Tools > установить галочку на NDK (side by side) > OK.
6. Выполнить скрипт в консоли:

```
./run_gradle.sh
```

Предполагается использование Linux и стандартная установка Android Studio