Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

з напряму підготовки 121 «Інженерія програмного забезпечення»

На тему: *Розробка мобільної гри з орієнтування на місцевості*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-22ск
_____ *Манджгаладзе А. Р.*

Керівник кваліфікаційної роботи _____ */ Стрюк А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2025

Криворізький національний університет
Факультет: *Інформаційних технологій*
Кафедра: *Моделювання та програмного забезпечення*
Освітньо-кваліфікаційний рівень: *бакалавр*
Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2025__ р.

ЗАВДАННЯ
на кваліфікаційну роботу
студента групи ІПЗ-22ск Манджгаладзе Антону Раульовичу

1. Тема: *Розробка мобільної гри з орієнтування на місцевості* затверджено наказом по КНУ №119 від «10» квітня 2025 р.
2. Термін подання студентом закінченого проекту «20» червня 2025 р.
3. Вихідні дані по роботі: *Пояснювальна записка: 64 сторінки, 14 рисунків, 1 додаток, 15 використаних у роботі джерел*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Визначення вимог до мобільної гри з орієнтування на місцевості. Проєктування програмного забезпечення. Реалізація програмного забезпечення мобільної гри.
5. Перелік графічного демонстраційного матеріалу: Приклади існуючих програм. Функціональна схема. Блок-схема основного алгоритму. Структура бази даних. Приклади роботи розробленної програми.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*13.02.2025-20.02.2025*
2	*Аналіз інформаційних джерел*	*21.02.2025-09.03.2025*
3	*Визначення вимог до програмного забезпечення*	*10.03.2025-18.03.2025*
4	*Розробка функціональної схеми*	*19.03.2025-23.03.2025*
5	*Розробка алгоритмів*	*24.03.2025-31.03.2025*
6	*Розробка бази даних*	*01.04.2025-14.04.2025*
7	*Розробка програмного забезпечення*	*15.04.2025-13.05.2025*
8	*Тестування програмного забезпечення*	*14.05.2025-20.05.2025*
9	*Оформлення пояснювальної записки*	*21.05.2025-12.06.2025*
10	*Розробка демонстраційних матеріалів*	*13.06.2025-18.06.2025*

Дата видачі завдання: «_____»_____ 2025 р.
Студент _____ Манджгаладзе А. Р.
Керівник роботи _____ Стрюк А. М.

РЕФЕРАТ

ОРІЄНТУВАННЯ, МОБІЛЬНА ГРА, ANDROID, ПРОГРАМУВАННЯ, GPS, НАВІГАЦІЯ, МАПА.

Кваліфікаційна робота: 64 с., 14 рис., 15 джерел.

Метою роботу є проєктування та реалізація мобільної гри, базовою механікою якої є орієнтування на місцевості.

В роботі досліджено базові особливості ігор з орієнтування, їх еволюція та розмаїття. Окрема увага приділена сучасним іграм з орієнтуванням, які базуються на використанні мобільних пристроїв та GPS-датчиків. Спроєктовано вимоги до мобільної гри з орієнтуванням на місцевості.

На етапі проектування програмного забезпечення побудовано функціональну схема мобільної гри, спроєктовано основні алгоритми та структура бази даних мобільної гри з орієнтуванням на місцевості.

Для реалізації програмного забезпечення мобільної гри було обрано IDE Android Studio та мову програмування Kotlin.

Створене програмне забезпечення може застосовуватись як індивідуальний засіб організації активного дозвілля, а також для цілеспрямованого розвитку навичок орієнтування на місцевості в різних навчальних ситуаціях.

ABSTRACT

ORIENTEERING, MOBILE GAME, ANDROID, PROGRAMMING, GPS, NAVIGATION, MAP.

Qualification work: 64 pages, 14 figures, 15 references.

The goal of the work is to design and implement a mobile game whose core mechanic is orienteering.

The study examines the fundamental features of orienteering games, their evolution, and diversity. Special attention is given to modern orienteering games that utilize mobile devices and GPS sensors. Requirements for a mobile orienteering game have been designed.

During the software design phase, a functional diagram of the mobile game was created, along with the design of key algorithms and the structure of the database for the mobile orienteering game.

Android Studio IDE and the Kotlin programming language were chosen for the implementation of the game's software.

The developed software can be used both as an individual tool for organizing active leisure and for the targeted development of orienteering skills in various educational scenarios.

ЗМІСТ

ВСТУП

Орієнтування на місцевості — це важлива навичка, яка допомагає людині зрозуміти, де вона знаходиться, і вибрати правильний напрямок руху. Вона включає вміння визначати сторони світу й своє місце відносно оточуючих об’єктів. На основі цієї навички створені різні ігри й спортивні види діяльності, які називають іграми з орієнтуванням або геоіграми. У таких іграх важливу роль відіграє реальне місцезнаходження гравця, яке зазвичай відстежується через GPS. Ці ігри можуть бути як активними, наприклад, геокешинг, де потрібно знаходити заховані предмети, так і цифровими, які працюють через мобільні додатки. Одним із популярних видів орієнтування є спортивне орієнтування, де учасники за допомогою карти та компаса намагаються знайти контрольні точки на незнайомій території.

Орієнтування було важливою навичкою для людей протягом всієї історії. Раніше це не було просто розвагою, а необхідністю для тих, хто полював, ловив рибу, ходив по морях, досліджував природу чи працював у лісах і горах. За допомогою цієї навички вони могли знайти дорогу та виконати свої завдання. У військовій справі орієнтування також мало велике значення — вміння читати й створювати карти часто визначало успіх у боях. Крім практичної користі, орієнтування допомагає краще розуміти природу, навчитися вирішувати складні задачі, працювати разом у команді та краще спілкуватися з іншими.

Спочатку орієнтування було потрібним для виживання і роботи, наприклад, у військових справах чи професіях. З часом ця корисна навичка стала основою для спортивних змагань, а із появою сучасних технологій — і для цифрових ігор. Це показує, як люди перетворюють необхідні вміння на спосіб розваги і конкуренції, допомагаючи їм ставати популярними і розвиватися. Спортивне орієнтування не тільки цікаве, але й допомагає тренувати важливі навички: вміння розбиратися в просторі і логічно мислити.

Ці здібності корисні навіть зараз, у час, коли технології все більше підтримують наше життя.

Зазвичай для ігор з орієнтування є організатори, які розробляють маршрути, контролюють їх проходження, забезпечують всі аспекти проведення змагань і опікуються учасниками. Це може включати створення карт, розміщення контрольних точок, підготовку обладнання та забезпечення безпеки учасників. Однак такий підхід часто вимагає значних ресурсів, часу і коштів, що обмежує доступність цього виду дозвілля для широкої аудиторії.

Нашою метою є зробити орієнтування простішим і доступнішим, перетворивши цей захопливий вид активності на інтерактивну цифрову гру. Для цього ми розробляємо мобільний застосунок, який переведе класичні принципи орієнтування у цифровий формат, поєднуючи їх із сучасними можливостями смартфонів. Застосунок буде включати інтерактивні маршрути, GPS-навігацію, автоматичне визначення місцезнаходження контрольних точок та елементи гри, такі як завдання або різні рівні складності. Завдяки простому і зрозумілому інтерфейсу, застосунок буде доступним для людей будь-якого віку і рівня підготовки, а його використання не вимагатиме спеціальних знань чи обладнання. Крім того, гравці зможуть самостійно створювати та налаштовувати власні маршрути, змагатися з друзями, а також відкривати нові локації в усіх куточках світу.

Таким чином, ми прагнемо не лише популяризувати орієнтування як вид активного дозвілля, але й зробити його більш доступним та захопливим для всіх, незалежно від місця проживання чи рівня підготовки. Наш продукт поєднує спорт, технології та можливість насолоджуватися природою, відкриваючи нові горизонти для самостійних подорожей і змагань.

1 ВИЗНАЧЕННЯ ВИМОГ ДО МОБІЛЬНОЇ ГРИ З ОРІЄНТУВАННЯ НА МІСЦЕВОСТІ

1.1 Базові особливості ігор з орієнтування

Орієнтуватися на місцевості означає вміти зрозуміти, де ти знаходишся, і яка дорога тобі потрібна. Для цього люди можуть користуватися різними способами: дивитися на природні чи створені людиною орієнтири, як-от дерева, будинки або дороги, а також застосовувати спеціальні прилади, наприклад, компас чи карту.

Люди здавна користувалися Сонцем, щоб зрозуміти, де які сторони світу. Вдень, коли Сонце у найвищій точці, воно завжди на півдні, а тінь від предметів показує на північ. Вранці Сонце сходить на сході, а ввечері заходить на заході. Завдяки цьому легко визначити, де північ, південь, схід і захід.

Щоб визначити сторони світу, окрім Сонця, можна орієнтуватися за іншими ознаками місцевості. Наприклад, мох найчастіше росте на північній стороні дерев і каменів. Також можна звертати увагу на будівлі: у православних церков, каплиць і лютеранських кирок головний вхід зазвичай дивиться на захід, а вівтар знаходиться на сході. У католицьких костьолів усе навпаки. Хрести на куполах церков часто розташовані так, щоб їхня площина вказувала напрямок "схід-захід". Крім того, піднятий кінчик нижньої перекладини хреста спрямований на північ.

1.1.1 Використання компаса та карти

Компас – це важливий інструмент, який допомагає визначати сторони світу. Щоб він працював правильно, треба тримати його рівно. Синій кінець стрілки показує на північ, а червоний – на південь. На корпусі компаса є позначки "Сх." і "Зх.", які вказують на схід і захід. Сучасні компаси, особливо для спортивного орієнтування, мають обертовий корпус із позначками сторін світу (Пн, Сх, Пд, Зх) і градусами, прозору основу, стрілку для вказування напрямку та рівні краї, які допомагають працювати з картою. Інколи в них є лупа, щоб краще бачити дрібні деталі на карті.

Карта є візуальним представленням місцевості з висоти пташиного польоту. Для ефективного орієнтування за картою необхідно розвивати навички спостереження, щоб зіставляти зображені на ній об'єкти з реальними особливостями ландшафту. Перед початком змагань з орієнтування вкрай важливо досконало розібратися, як користуватися як компасом, так і картою.

Орієнтування — це не лише визначення сторін горизонту та використання інструментів. Це також навички, як визначати відстань і напрямок (азимут). Люди тренуються рахувати, скільки кроків вони роблять, щоб розраховувати відстань в "парах кроків", і вчаться рухатися у потрібну сторону за азимутом. Якщо потрібно виміряти відстань між точками на карті по кривій лінії, для цього використовують спеціальний прилад — кувіметр.

Хоч сучасні технології, наприклад GPS, швидко розвиваються, старі способи орієнтування, як користування Сонцем, компасом, картою та місцевими ознаками, все одно дуже важливі. Експерти кажуть, що ці знання не тільки корисні, а інколи й необхідні для виживання, бо електронні пристрої можуть розрядитися або не працювати там, де немає сигналу. Тому традиційні способи залишаються надійним "запасним варіантом". Це показує, що технології доповнюють старі методи, але не замінюють їх повністю. Важливо поєднувати обидва підходи, щоб навчитися добре орієнтуватися у будь-якій ситуації. Справжнє вміння орієнтуватися полягає у тому, щоб використовувати все доступне і пристосовуватися до різних умов.

1.1.2 Історичні корені та ранні форми ігор з орієнтуванням

Ігри з орієнтуванням мають давню історію, яка пов’язана як із життєво важливими потребами людей, так і з традиціями стародавніх культур. Спочатку орієнтування було необхідністю, а з часом перетворилося на організовані змагання, вид спорту та спосіб відпочинку.

Вміння розуміти, де ти знаходишся і як дістатися потрібного місця, завжди було дуже важливим для людей, щоб вижити та успішно робити свою роботу. У військовій справі це особливо важливо, адже вміння використовувати місцевість правильно багато разів допомагало перемагати у

битвах. Це було потрібно і в давніх сутичках, і в сучасних війнах. Окрім військових, орієнтування на місцевості було необхідним для багатьох інших людей, які часто переміщувалися або працювали у невідомих місцях. Наприклад, рибалкам, морякам, мандрівникам, шукачам корисних копалин, працівникам лісу та геологам.

Ідея "пошуку скарбів" (scavenger hunts) з'явилася дуже давно і бере свій початок у стародавніх іграх. У різних культурах люди проводили подібні активності як частину свят або ритуалів. Тоді учасники шукали предмети або виконували завдання, що були частиною святкового дійства. Наприклад, у стародавній Греції під час свята Діонісія організовували такі забави, а в середньовічній Європі вони були популярними на свята врожаю чи інших заходах. Такі ігри проводили навіть у племенах корінних американців, щоб навчити молодь важливим навичкам, розповісти про рідну природу та культуру.

Навігація в місцевості — це важливий навик для військових. Навчання з орієнтування часто проходить у важких умовах, наприклад, уночі чи під час дощу, щоб підготувати солдатів до складних ситуацій. Саме з військової навігації наприкінці 19-го століття з'явився спорт під назвою "орієнтування". Це слово вперше використали у 1886 році для опису руху по невідомій місцевості з картою і компасом. Відомо, що змагання з орієнтування почали проводити ще у 1895 році в північних військових гарнізонах, навіть до того, як з'явилися перші відкриті змагання для всіх..

Орієнтування спочатку було важливим навиком для виживання та військових дій. З часом його елементи потрапили в народні ігри та традиції, що показує, як практичні знання поступово ставали частиною розваг. Особливо важливу роль зіграв військовий контекст, де навички орієнтування на місцевості почали використовувати для навчання солдатів, завдяки таким діячам, як Роберт Бейден-Поуелл. Це допомогло зробити орієнтування більш організованим і перетворити його на спортивну дисципліну. Тобто процес навчання важливих навичок через ігри природний для людей, а військова

практика сприяла тому, щоб орієнтування стало відомим спортивним заняттям, доступним для всіх бажаючих.

1.1.3 Спортивне орієнтування: становлення та розвиток

Спортивне орієнтування є одним з найяскравіших прикладів трансформації життєво важливої навички у повноцінний вид спорту. Його історія бере початок у Північній Європі наприкінці 19 століття.

Спортивне орієнтування з’явилося наприкінці 19 століття в північних країнах Європи, таких як Швеція та Норвегія. Вперше термін «орієнтування» почали використовувати у 1886 році. Він означав пошук дороги через незнайому місцевість за допомогою карти та компаса. У той же час було придумано бігові змагання через складну місцевість, щоб тренувати солдатів і вчити їх швидко мислити. А перші офіційні змагання зі спортивного орієнтування для всіх охочих відбулися в Норвегії в 1897 році.

У 1930-ті роки спортивне орієнтування почало стрімко розвиватися у Швейцарії, Данії, Норвегії, а до 1934 року поширилося також в Угорщині та СРСР. Перші міжнародні змагання відбулися у 1947 році між скандинавськими країнами.

Важливою подією для спортивного орієнтування стало створення Міжнародної федерації спортивного орієнтування (IOF) 21 травня 1961 року на зібранні в Копенгагені. До її складу спочатку увійшли 10 європейських країн: Болгарія, Угорщина, НДР, Данія, Норвегія, Фінляндія, ФРН, Чехословаччина, Швейцарія та Швеція. До 1977 року федерацію визнав Олімпійський комітет. Хоча спортивне орієнтування так і не стало олімпійським видом спорту, воно є частиною Всесвітніх ігор, які проходять раз на чотири роки. Сьогодні IOF об'єднує вже 80 країн, що показує, наскільки цей спорт став популярним у світі.

Рисунок 1.1 – Спортивне орієнтування на місцевості

Спортивне орієнтування почалось із військової підготовки, де військові гарнізони організовували змагання з орієнтування задовго до того, як це стало популярним серед звичайних людей. Спортивне орієнтування швидко поширилось по країнах Північної Європи, а потім і по всьому світу. Це призвело до створення міжнародної організації IOF, яка зробила цей спорт офіційним і відомим. Визнання від Олімпійського комітету закріпило статус спортивного орієнтування як важливого виду спорту у світі. Загалом, цей приклад показує, як практичні військові вправи можуть перетворитися на популярний спорт. Чіткі правила, змагальний характер і організованість взяті з військової підготовки, допомогли зробити орієнтування цікавим для багатьох людей як видом відпочинку та спортивних змагань.

Офіційні формати Чемпіонатів світу зі спортивного орієнтування, за якими слідує більшість регіональних та національних чемпіонатів, включають різноманітні дисципліни:

Далека відстань: Найтриваліші та найскладніші індивідуальні випробування, що проходять у лісовій місцевості. Маршрути розробляються для значного фізичного виклику та різноманітного рівня технічної складності.

Середня дистанція: Відносно коротші змагання, що проводяться в лісі, на технічно складній місцевості.

Спринт: Швидкісні змагання, що проводяться в міських умовах, технічно нескладні, але складні з вибором маршруту.

Естафета: Змагання з масового старту, що складаються з команд з 3 учасників, де різні бігуни розділяються за допомогою розгалужень.

Спринтерська естафета: Проводиться командами з 4 осіб (першою та останньою повинні бути жінки) у міській місцевості з масовим стартом та розгалуженням. Це захоплююча та телевізійна подія.

Нокаут-спринт: Новий формат міського орієнтування, який починається з короткої кваліфікаційної гонки, що визначає стартові позиції для серії дуже коротких гонок на вибування з масовим стартом.

Дистанція у спортивному орієнтуванні визначається тим, як спортсмен добирається до контрольного пункту, надаючи можливість вибору покриття (пісок, ґрунт, трава).

Для участі у змаганнях, крім карти та компаса, спортсменам можуть знадобитися такі додаткові предмети: кувіметр (прилад для вимірювання відстані на звивистій лінії карти), налобний ліхтарик (особливо для нічних забігів), свисток (для виклику допомоги) та телефон.

Спортивне орієнтування може бути дуже різноманітним, залежно від місця та формату змагань. Це можуть бути довгі дистанції в лісі, короткі міські спринти чи особливі нокаут-гонки. Спорт адаптується до різних умов, наприклад, природи чи міста, і розвиває різні навички спортсмена: витривалість, вміння орієнтуватися, вибір найкращого маршруту та швидкість. Формати, як "Спринтерська естафета" або "Нокаут-спринт", були створені, щоб зробити орієнтування більш цікавим для глядачів і популярним, показувати його на телебаченні.

Завдяки новим форматам цей спорт постійно змінюється, щоб залишатися актуальним і цікавішим для людей. Змагання тепер охоплюють не тільки прогулянки незнайомою місцевістю, але й швидке прийняття рішень, вирішення завдань у міських умовах та роботу в команді. Це допомагає глибше зрозуміти, чим є сучасне спортивне орієнтування.

1.1.4 Орієнтування у скаутському русі

Скаутський рух, заснований на початку 20 століття, інтегрував навички орієнтування як ключовий елемент своєї виховної програми, спираючись на військовий досвід свого засновника.

Заснування скаутського руху та вплив Роберта Бейден-Поуелла

Скаутинг, або скаутський рух, — це добровільна організація для молоді, яка не пов’язана з політикою. Вона допомагає дітям і підліткам розвиватися. Початок скаутингу поклали в 1907 році, коли британський військовий Роберт Бейден-Поуелл організував перший скаутський табір на острові Браунсі в Англії. Після цього він написав книгу «Скаутинг для хлопчиків» (1908), у якій пояснив основні ідеї скаутингу. Ця книга досі вважається важливою для скаутського руху.

Військовий досвід Бейден-Поуелла сильно вплинув на створення скаутських навичок. Він служив у армії з 1876 до 1903 року, побував в Індії, на Балканах, у Західній Африці, Південній Африці та на Мальті. У молодості він займався створенням карт і збором важливої інформації, а також навчав цьому інших солдатів. Бейден-Поуелл придумав спосіб організовувати солдатів у маленькі групи (підрозділи), де кожна мала свого лідера, і нагороджував солдатів за успіхи спеціальними значками. Цей підхід пізніше став основою для організації скаутів.

Символ скаутів, що спочатку був "наконечником стріли", який вказує на північ на карті або компасі, базувався на значку, який Бейден-Поуелл використовував для армійських розвідників в Індії. Пізніше він перейменував символ на геральдичну лілію (fleur-de-lis) як символ миру та чистоти, а три її точки стали нагадуванням про три обіцянки, які скаут дає при вступі до руху.

Скаутинг використовує особливий підхід до навчання, який допомагає дітям розвиватися через практичні заняття. У програмі багато активностей, як-от: кемпінг, вміння виживати в природі, водні види спорту, походи та фізична активність. Молодші скаути (від 6 до 10 років) поступово навчаються орієнтуватися у лісі чи місті вдень і вночі без карт чи компаса. Це допомагає їм вирішувати задачі, працювати в команді, спілкуватися та поважати природу.

Скаутський рух використовує навички, які колись були частиною військового життя, і перетворює їх на спосіб навчання та розвитку молоді. Засновник руху, Бейден-Поуелл, взяв такі вміння, як розвідка і картографія, і зробив їх частиною освітньої програми для юнаків. Навіть символ скаутів — лілія, яка еволюціонувала з військового значка у знак миру та допомоги, показує цю зміну. Скаутинг вчить молодь не просто практичним умінням, як орієнтуватися на місцевості, але й важливим цінностям: бути відповідальними, допомагати іншим, жити у гармонії з природою та прагнути миру. Навчання відбувається поступово — спочатку прості навички, як орієнтування без карти і компаса, а потім складніші завдання. Це допомагає дітям і підліткам розвивати самостійність, вміння працювати в команді та ставитися до природи з повагою. Загалом скаутський рух показує, як військові навички можуть бути змінені та використані для виховання молоді і розвитку суспільства.

1.1.5 Військове орієнтування та його еволюція

Військове орієнтування є однією з найдавніших та найважливіших дисциплін, що постійно розвивається під впливом технологічного прогресу.

Військова топографія – це дуже важливий навик для підготовки командирів, офіцерів, сержантів та солдатів Збройних Сил України. Вміти орієнтуватися, читати і створювати карти необхідно кожному військовому. Навчання навігації проводиться в складних умовах, таких як гори, ліс, ніч або дощ, щоб вони могли впевнено діяти в будь-яких ситуаціях.

Рисунок 1.2 – Військове орієнтування

Усі війни показують, як важливо для командирів правильно використовувати місцевість, щоб перемагати. Військова топографія, тобто знання про рельєф та місцевість, тісно пов’язана з тактикою бою, створенням карт та вивченням форми земної поверхні. Ще в 1895 році у країнах Північної Європи військові проводили змагання з орієнтування, що показує, як рано ця навичка стала частиною військових тренувань. Українська армія теж багато уваги приділяє навчальним матеріалам і тренуванням, щоб солдати добре розбиралися у топографії та орієнтуванні на місцевості.

Військові часто стають двигуном прогресу в навігаційних технологіях. Військова топографія завжди вважалася дуже важливою сферою, а GPS спочатку був розроблений для військових потреб. Завдяки цьому військові стали ключовими ініціаторами розвитку способів орієнтування на місцевості. Складність військових тренувань і необхідність точності під час війни, наприклад, у Перській затоці, змушували створювати надійні і точні технології. Ці розробки, спочатку зроблені для армії, згодом знайшли застосування і в цивільному житті, наприклад, у навігації для кожного з нас. Таким чином, інновації, керовані військовими, впливають на всі сфери життя, зокрема й на розвиток ігор з орієнтуванням.

Одним із важливих кроків у розвитку військової навігації стало використання супутників. У 1959 році Військово-морський флот США створив свою першу супутникову систему Transit, яку використовували для визначення місця перебування атомних підводних човнів. А у грудні 1973 року Міністерство оборони США почало працювати над системою GPS (NAVSTAR), яку вперше протестували у 1974 році.

GPS продемонструвала свою важливість під час війни в Перській затоці у 1991 році. Вона допомогла точніше спрямовувати бомбардування і правильно розміщувати війська. Система GPS стала можливою завдяки новим технологіям, таким як сучасні мікропроцесори, комп'ютери і атомні годинники. Ці винаходи дозволили створювати маленькі GPS-пристрої, схожі за розміром на мобільний телефон, що зробило технологію широко доступною та популярною.

Перехід від вмінь користуватися картою і компасом, які важливі у військовій справі, до використання GPS є великим змінам. Хоча старі методи все ще необхідні в ситуаціях, коли GPS може не працювати, сучасна технологія дає точні дані про місцезнаходження і суттєво вплинула на військові операції, як було видно під час війни в Перській затоці. Це показує, що військові мають навчатися і традиційним методам, і новим технологіям.

Загалом, нові технології не заміняють старі вміння, а доповнюють їх, дозволяючи створити більш ефективний спосіб навігації.

1.2 Сучасні ігри з орієнтуванням

Сучасні ігри з орієнтуванням зазнали революційних змін завдяки розвитку технологій, зокрема GPS, мобільних пристроїв та доповненої реальності.

Технологія GPS стала важливою частиною життя, коли її дозволили використовувати звичайним людям. Спочатку GPS створили у 1973 році для військових цілей у США. Але 2 травня 2000 року, в день, який назвали "Blue Switch Day", уряд США зробив цю технологію доступною всім. До цього GPS могли використовувати тільки військові, уряд чи окремі цивільні люди.

Геокешинг — це гра, де люди шукають сховані скарби, використовуючи GPS-навігатори або смартфони. Її придумали у 2000 році, коли Дейв Ульмер вирішив перевірити, наскільки точно працює GPS. Він сховав у лісі коробку і назвав це "Велике американське полювання на схованки". Ідея швидко стала популярною, бо про неї писали у відомих новинних джерелах, як-от Slashdot, New York Times і CNN. Люди по всьому світу почали ховати спеціальні контейнери, які називають "геокеші", і давати координати, щоб інші могли їх знайти. В Україні ця гра з'явилася у 2011 році завдяки проекту "Шукач".

У спортивному орієнтуванні GPS також знайшов широке застосування. В останні роки GPS активно використовується для розробки нових дисциплін, картографування, прямих трансляцій та відстеження спортсменів у реальному часі, що значно сприяє технологічному прогресу у спорті. До появи GPS компас був основним навігаційним інструментом для пересування на відкритому повітрі, і навіть сьогодні він має сильних прихильників, що підкреслює його незмінну цінність.

Рисунок 1.3 – Геокешинг

У 2000 році стався важливий момент, який вплинув на використання GPS для звичайних людей. Раніше точна GPS-технологія була доступна тільки для військових. Але після того, як її зробили доступною для всіх, з'явилася нова гра — геокешинг, яка швидко стала популярною у всьому світі. Це показує, як військова технологія, ставши доступною для цивільних, знайшла практичне і розважальне застосування. GPS змінив правила гри, зробивши навігацію доступною для кожного, навіть без спеціальних знань про карту і компас. Тепер люди могли концентруватися на пошуках і пригодах, використовуючи сучасні технології, замість того щоб витрачати час на складне навчання.

Нижче наведено хронологію розвитку GPS та його вплив на ігри з орієнтуванням:

Таблиця 1.1 – Хронологія Розвитку GPS та Його Вплив на Ігри

Рік/Період	**Ключова Подія/Технологія**	**Вплив на Ігри з Орієнтуванням**
1959	Створення супутникової навігаційної системи Transit ВМС США	Зародження супутникової навігації для військових цілей.
1960	Запуск першого супутника Transit	Початок ери супутникової навігації.
1973	Початок розробки NAVSTAR (GPS) Міністерством оборони США	Закладення основи сучасної глобальної системи позиціонування.
1974	Запуск першого супутника Timation з атомним годинником	Значне підвищення точності позиціонування, що є критичним для ігор.
2000 (2 травня)	"Blue Switch Day" - точний GPS стає доступним для цивільних	Демократизація точної навігації, що призводить до появи нових типів ігор.
2000 (травень)	Початок геокешингу (Dave Ulmer)	Перша масова GPS-гра, що перетворює пошук скарбів на глобальне хобі.
2002	Геокешинг з'являється в Росії	Розширення географії поширення GPS-ігор.
2011	Геокешинг з'являється в Україні	Подальше глобальне поширення та локалізація гри.
Середина 2010-х	Поява мобільних AR-ігор (Ingress, Pokémon Go)	Інтеграція GPS з доповненою реальністю, створення нового ігрового досвіду, що поєднує віртуальний та реальний світи.

Розвиток мобільних технологій та доповненої реальності (AR) значно розширив можливості ігор з орієнтуванням. Сучасні смартфони стали потужними інструментами для навігації та ігор, використовуючи свої вбудовані можливості GPS. Смартфони визначають точне місцезнаходження за допомогою сигналів з 3-4 супутників, дозволяючи користувачеві орієнтуватися на картах у реальному часі. Багато популярних додатків, таких як Google Maps, Uber та Glovo, поєднують GPS-дані з мобільним зв'язком для надання зручних послуг. Мобільний інтернет забезпечує доступ до карт,

навігаційних даних та постійне оновлення маршрутів. Хоча GPS може працювати незалежно від мобільної мережі, без інтернету відсутні оновлення карт, що ускладнює доступ до актуальних даних про маршрути та трафік.

Сучасні телефони та технології доповненої реальності (AR) дають людям нові можливості для ігор і навігації. Сучасні смартфони дуже потужні: вони можуть швидко визначати місцезнаходження за допомогою GPS, який працює через супутники. Завдяки цьому можна легко знаходити потрібне місце на карті прямо в телефоні. Популярні додатки, як-от Google Maps, Uber або Glovo, використовують GPS і мобільний інтернет, щоб пропонувати зручні сервіси. Інтернет потрібен, щоб завантажувати карти, будувати маршрути та отримувати оновлення. Без Інтернету GPS теж працює, але карти не оновлюються, тому маршрути можуть бути неправильними, а затори на дорогах — не відображатися.

Також розвиток мобільних технологій відкрив шлях для нових ігор, що залежать від вашого місця. Такі ігри називають геоіграми. У них гра будується відповідно до того, де знаходиться гравець. Це можуть бути як цифрові ігри, так і фізичні. Наприклад, є "міські ігри" або "вуличні ігри", які зазвичай багатокористувацькі й проводяться на вулицях міста.

Приклади AR-ігор, таких як Pokémon Go, Ingress та Jurassic World Alive, є одними з найвідоміших, що використовують геолокацію. Вони застосовують інтерфейс Image Linked Map (ILM), де затверджені геотеговані локації відображаються на стилізованій карті, згенерованій на основі даних GPS, для взаємодії користувача. Інші приклади включають DinoMess, The Walking Dead: Our World та Cats GO. Ці ігри часто використовують доповнену реальність для створення занурюючого досвіду, поєднуючи віртуальні елементи з реальним світом.

Рисунок 1.4 – Pokémon Go

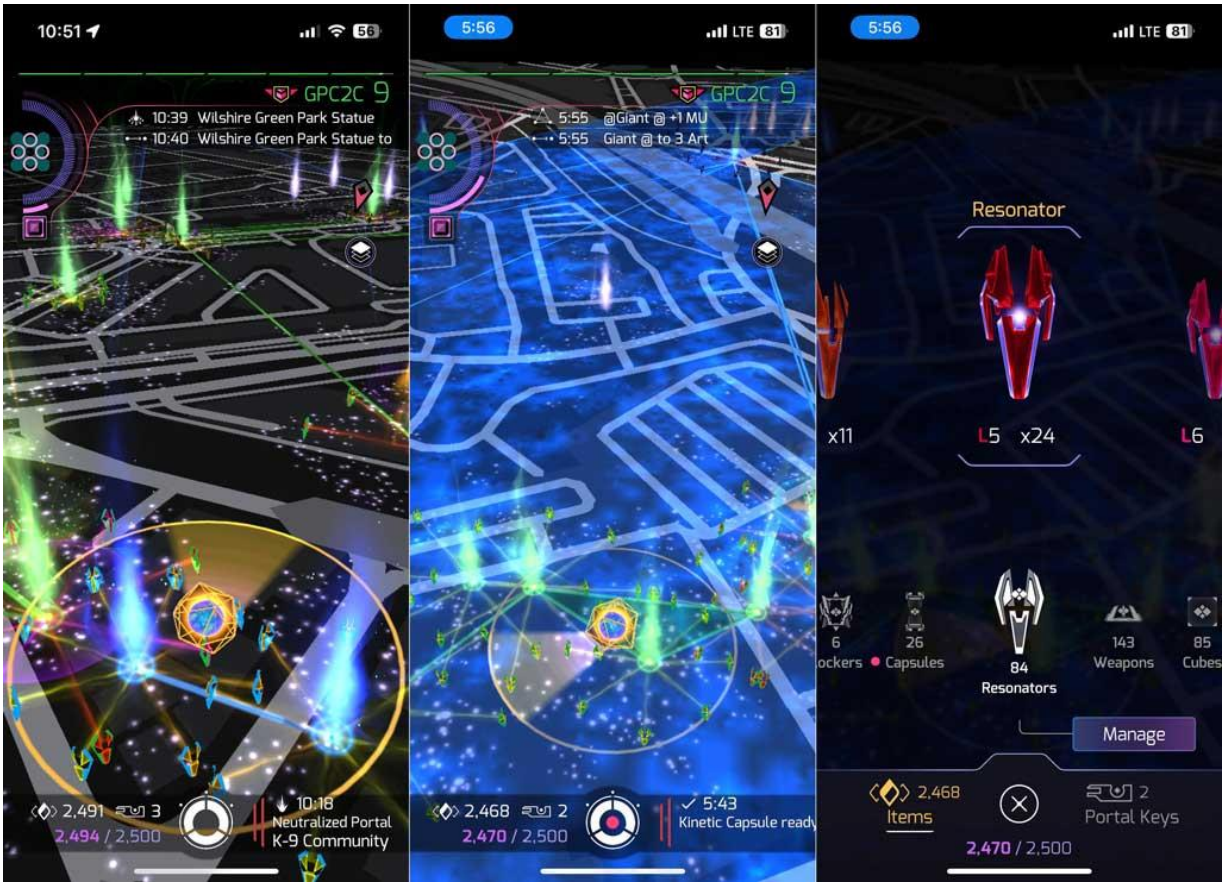

Рисунок 1.5 – Ingress

Ігри, засновані на місцезнаходженні, особливо AR-ігри, такі як Pokémon Go та Ingress , представляють глибоку конвергенцію цифрового та фізичного світів. Вони не просто використовують реальні місця як фон; вони активно інтегрують цифрові елементи (віртуальні істоти, портали, стилізовані карти) у фізичне середовище. Використання смартфонів як основного інтерфейсу для цих ігор означає, що сам пристрій стає шлюзом між цифровим ігровим світом та реальним фізичним простором. Це створює занурюючий досвід, який розмиває межі між реальністю та ігровим процесом. Ця тенденція означає рух за межі просто "орієнтування" у реальному світі до "взаємодії" з цифровою доповненою реальністю. Вона відкриває нові педагогічні можливості , де навчання є ситуативним та експериментальним.

Цифрові карти сильно впливають на зручність використання та ігровий досвід. Вони допомогли людям краще взаємодіяти з географічною інформацією та стали дуже популярними. Геолокацію часто використовують у навчальних іграх, наприклад, для створення пошуку скарбів чи завдань, які пов’язані з реальними місцями. Це мотивує людей досліджувати нові райони і вирішувати практичні задачі.

Проте є проблема: цифрові карти не завжди зручні для людей з інвалідністю. Наприклад, там може не бути текстових описів, потрібного контрасту кольорів чи легкого управління через клавіатуру. Хоча ці карти стали важливою частиною сучасних ігор та навчальних програм, недоліки в їх доступності створюють труднощі для деяких користувачів. Це парадокс — технології мають допомагати, але інколи вони додають нові перешкоди.

У майбутньому важливо працювати над тим, щоб такий дизайн був інклюзивним і доступним для всіх, незалежно від фізичних можливостей. Тільки ретельний підхід до розробки допоможе зробити переваги цифрових карт доступними кожному.

1.3 Проєктування вимог до мобільної гри з орієнтуванням на місцевості.

Більшість програм, які ми розглянули, або просто створюють імітацію спортивного орієнтування на пристрої, або виконують додаткові завдання, наприклад, допомагають створювати карти маршрутів і залишати до них коментарі. Тільки деякі з цих додатків можна використовувати для справжніх змагань, але навіть вони мають певні недоліки:

– Складний інтерфейс. Якщо інтерфейс системи складний і незручний, користувачам може бути важко створювати нові маршрути. Це може траплятися через заплутаний дизайн, занадто багато кроків для виконання завдань або незрозумілі меню. Проблеми також можуть виникати через погане відображення карт, складність у додаванні зупинок або слабку інтеграцію з геолокаційними сервісами. Все це забирає у людей багато часу та зусиль навіть для простих задач. Щоб вирішити ці питання, потрібно зробити інтерфейс простішим, додати зручні інструменти, наприклад автозаповнення маршрутів, а також покращити взаємодію системи з картами. Це полегшить процес і зробить його більш зрозумілим, навіть для тих, хто вперше використовує систему.

– Відсутність підтримки української мови. Відсутність української локалізації може створювати ряд значних незручностей для користувачів, які воліють працювати із програмним забезпеченням, вебсайтами чи мобільними додатками рідною мовою. Це питання є важливим не тільки з точки зору зручності, але й культурної ідентичності, оскільки локалізація сприяє інтеграції продукту в місцевий контекст, роблячи його зрозумілішим і доступнішим для громади.

Тому ми ставимо перед собою амбітну задачу розробити захопливу мобільну гру, де основною механікою стане орієнтування на місцевості. Концепція гри побудована на інтерактивному взаємодії з реальним простором, а її головною метою буде допомога гравцям у вдосконаленні навичок навігації, просторового мислення та логіки. У процесі гри користувачі зможуть

виконувати різноманітні завдання, вирішувати головоломки, шукати приховані точки, долати природні перешкоди та навчатися користуватися картами, компасами та навіть сучасними GPS-навігаторами.

Щоб зробити геймплей максимально цікавим, ми плануємо інтегрувати елементи доповненої реальності, які дозволять гравцям досліджувати віртуальні елементи, що накладаються на реальні локації. Завдання будуть варіюватися від простих міні-квестів до складних марафонів, які можна проходити як поодинці, так і в команді. Крім того, гра матиме навчальний режим, що дозволить новачкам ознайомитися з основами орієнтування та практикуватися в безпечному середовищі.

1.3.1 Системні вимоги

Системні вимоги до мобільної гри з орієнтуванням на місцевості можуть включати кілька ключових аспектів, що забезпечують стабільну роботу додатку, високу продуктивність і належний рівень користувацького досвіду. Ось розширений перелік вимог:

1. Операційна система

– Мінімальна підтримувана версія ОС:

– Android: версія не нижче Android 8.0 (Oreo), що гарантує сумісність із сучасними API для роботи з геолокацією, Bluetooth та іншими технологіями.

2. Процесор

– Мінімум: 4-ядерний процесор з тактовою частотою не нижче 1.6 ГГц.

– Рекомендовані процесори:

– Qualcomm Snapdragon серії 600 або вище для Android.

3. Оперативна пам'ять

– Мінімальна кількість ОЗП:

– 2 ГБ для базової роботи додатку.

- Рекомендована кількість:

– 3–4 ГБ для плавного завантаження карт, графіки, а також для інтерактивної роботи в багатозадачному режимі.

4. Екран

– Необхідна роздільна здатність:

– Мінімум HD (720p) — для читабельності текстових підказок та деталізації карт.

– Рекомендовано: Full HD (1080p) або вище для кращого відображення графічного матеріалу.

– Сенсорний екран повинен підтримувати багатоточкову функцію (Multi-Touch) для зручної навігації.

5. Графіка

– Мінімальна вимога до графічного процесора:

– GPU з підтримкою OpenGL ES 3.1 або Metal API (Apple).

– Рекомендовано:

– Графічні прискорювачі Adreno 5xx або вище на Android.

6. Накопичувач

– Мінімальний обсяг вільної пам'яті:

– 500 МБ для установки гри та початкових даних.

– Рекомендоване місце:

– 2–4 ГБ для збереження великих карт, точок маршруту, користувацьких даних і кешу.

7. Підключення

– Wi-Fi або мобільний інтернет:

- Гра потребує стабільного підключення для звантаження карт, участі в багатокористувацьких режимах або синхронізації прогресу.

- GPS:

– Обов’язково активний модуль GPS для точної геолокації гравця на місцевості.

– Bluetooth:

- У разі підтримки зовнішніх пристроїв (наприклад, браслетів для відстеження або датчиків).

8. Датчики

– GPS для точного визначення місцезнаходження.

– Акцелерометр, гіроскоп і компас для інтуїтивної орієнтації в просторі.

– Барометр (опціонально) для точного відображення висоти над рівнем моря в гірській місцевості.

9. Акумулятор

– Рекомендований обсяг батареї:

– Мінімум 3000 мА·год для тривалої гри в польових умовах.

– Підтримка режимів енергозбереження для мінімізації витрат енергії у фоновому режимі.

10. Додаткові особливості

– Мультимедіа:

– Динамік і підтримка навушників для звукових сигналів та супровідних інструкцій.

Ці вимоги забезпечують оптимальну роботу мобільної гри в умовах різної місцевості та підвищують задоволення гравців від користування додатком.

1.3.2 Функціональні вимоги:

Функціональні вимоги до гри з орієнтуванням на місцевості:

1. Реєстрація користувачів:

- Забезпечити можливість створення особистого профілю для кожного учасника гри.

- Опція авторизації через e-mail, соцмережі або інші платформи.

2. Створення ігрового сценарію:

- Адміністратор чи організатор гри повинен мати можливість завантажувати маршрути на інтерактивну карту.

- Додавання контрольних точок із завданнями, питаннями чи підказками.

- Налаштування обмежень часу на виконання завдань.

3. Інтерактивна карта:

- Інтеграція цифрової карти з можливістю позиціонування гравців у реальному часі.

- Позначення старту, фінішу та проміжних контрольних точок.

- Відображення доступних маршрутів і зон для орієнтування.

4. GPS-навігація:

- Вбудовані інструменти для визначення місцезнаходження учасників.

- Здатність відстежувати, чи правильно учасник проходить маршрут.

5. Система завдань:

- На контрольних точках учасникам повинно видаватися завдання: загадка, квест чи інше інтерактивне завдання.

- Завдання мають генеруватися залежно від складності маршруту.

6. Реалізація командного режиму:

- Можливість створення груп з кількох учасників.

- Координація дій гравців усередині команди через внутрішній чат чи інші засоби комунікації.

7. Система балів:

- Нарахування очок за правильне виконання завдань, швидкість проходження маршруту та інші критерії.

- Оптимізація алгоритму підрахунку з урахуванням складності ігрового сценарію.

8. Реалізація таймінгу:

- Вбудований таймер для моніторингу часу, що залишився до завершення маршруту.

- Опції створення обмежених або відкритих ігрових сесій за часом.

9. Підказки та підтримка:

- Інтеграція функціоналу для видачі підказок під час проходження гри, якщо учасники не можуть знайти точку чи розв’язати завдання.

- Функція зв’язку з організатором для отримання допомоги.

10. Відстеження прогресу:

- Постійне оновлення прогресу учасників на маршруті.

- Аналітика щодо виконаних завдань та зібраних контрольних точок.

11. Функціонал завершення гри:

- Визначення переможців на основі балів, часу чи інших параметрів залежно від умов гри.

- Формування статистики для кожного учасника чи команди.

12. Мобільна сумісність:

- Забезпечити повну функціональність гри на мобільних пристроях через додаток або адаптовану вебверсію.

- Інтеграція push-повідомлень для оновлення статусу гри чи оповіщення про нові завдання.

13. Безпека даних:

- Захист персональних даних учасників відповідно до актуального законодавства.

- Обмеження доступу до маршруту чи персональної інформації.

14. Гнучкість налаштувань:

- Можливість створення сценаріїв із різним рівнем складності для новачків чи досвідчених гравців.

- Функція зміни маршруту чи умов гри залежно від ситуації (наприклад, погодних умов).

15. Інтеграція зі соціальними мережами:

- Надання можливості гравцям ділитися результатами гри чи маршрутами в соцмережах.

- Залучення аудиторії для популяризації гри.

16. Зворотній зв’язок:

- Опція оцінки гри учасниками після її завершення.

- Формування бази відгуків для покращення ігрового процесу.

Визначивши вимоги, можемо перейти до проєктування програмного забезпечення мобльної гри.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОЇ ГРИ З ОРІЄНТУВАННЯМ НА МІСЦЕВОСТІ

2.1 Функціональна схема мобільної гри з орієнтування на місцевості

Відповідно до сформульованих вимог, була розроблена функціональна схема мобільної гри з орієнтуванням на місцевості (рис. 2.1), яка враховує ключові аспекти геймплею, інтерактивності та технічного виконання. Основні елементи схеми включають:

1. Вступний екран гри: На цьому етапі гравець отримує вітання, ознайомлюється з правилами гри та базовими інструкціями щодо управління, а також має можливість налаштувати параметри гри.

2. Реєстрація та авторизація: Для забезпечення персоналізації процесу гра пропонує можливість створення акаунта або входу через соціальні мережі чи електронну пошту. Це дозволяє зберігати прогрес та досягнення кожного гравця.

3. Вибір рівня та локації: Після авторизації користувач обирає рівень складності, а також географічну локацію для проведення гри. Локації можуть бути реальними територіями або віртуальними картами, створеними за допомогою інструментів доповненої реальності.

4. Геймплей: Основна частина гри. Гравець отримує завдання, яке потребує орієнтування на місцевості чи вирішення географічних головоломок. Це включає пошук точок на карті, виконання інтерактивних завдань, взаємодію з об’єктами реального світу або алгоритмами гри. Інтеграція GPS дає змогу відстежувати переміщення гравця в реальному часі.

5. Система підказок: Гра передбачає наявність підказок для новачків або тих, хто зіткнувся зі складнощами у виконанні завдання. Система може пропонувати текстові, голосові або графічні підказки.

6. Зворотний зв’язок і оцінювання: Після завершення кожного рівня гравець отримує оцінку своєї продуктивності, досягнутих результатів і

кількість набраних балів. Визначається рейтинг на основі швидкості виконання завдань і точності орієнтування.

Рисунок 2.1 – Функціональна схема мобільної гри з орієнтуванням на місцевості

7. Рекорди та призи: Гравцям пропонуються додаткові стимули у вигляді досягнень, медалей або призів за успішне проходження рівнів чи виконання особливих завдань.

8. Налаштування користувача: У будь-який момент гри користувач може змінити параметри, такі як графіка, звуковий супровід, прив'язка до геолокації чи іншого функціоналу, що полегшує адаптацію гри до потреб гравця.

Ця функціональна схема базується на інтеграції сучасних технологій, таких як використання GPS-модулів, доповнена реальність, мобільна графіка високої якості та системи управління даними. Разом ці елементи забезпечують цікавий і захоплюючий геймплей для користувачів, роблячи гру доступною, зручною і водночас пізнавальною завдяки фокусу на навчальному досвіді та розвитку навичок орієнтування на місцевості.

2.2. Основні алгоритми мобільної гри з орієнтуванням

Блок-схема алгоритму створення маршруту для мобільної гри з орієнтуванням, представлена на рисунку 2.2, демонструє послідовність дій, які забезпечують оптимальне формування маршруту для гравця, враховуючи параметри місцевості, складність завдань, доступність контрольних точок та зручність навігації в рамках гри.

Основні етапи алгоритму, відображені на блок-схемі:

1. Ініціалізація параметрів гри. На цьому етапі користувач вводить дані, що стосуються місцевості, початкових координат, фіксованих точок маршруту та додаткових обмежень, таких як відстань між точками чи складність завдань.

2. Аналіз місцевості. Програма аналізує карту гри, визначає доступні території, враховує фізичні перешкоди (річки, гори тощо) та недоступні зони.

3. Вибір ключових контрольних точок. Алгоритм визначає точки, які обов’язково мають стати частиною маршруту — це місця із завданнями, бонусами чи стратегічними цілями.

4. Розрахунок оптимального шляху між точками. На цьому етапі проводиться розрахунок маршруту, який забезпечує мінімальну витрату часу

або ресурсів гравця, враховуючи відстань, рівень складності завдань і рельєф місцевості.

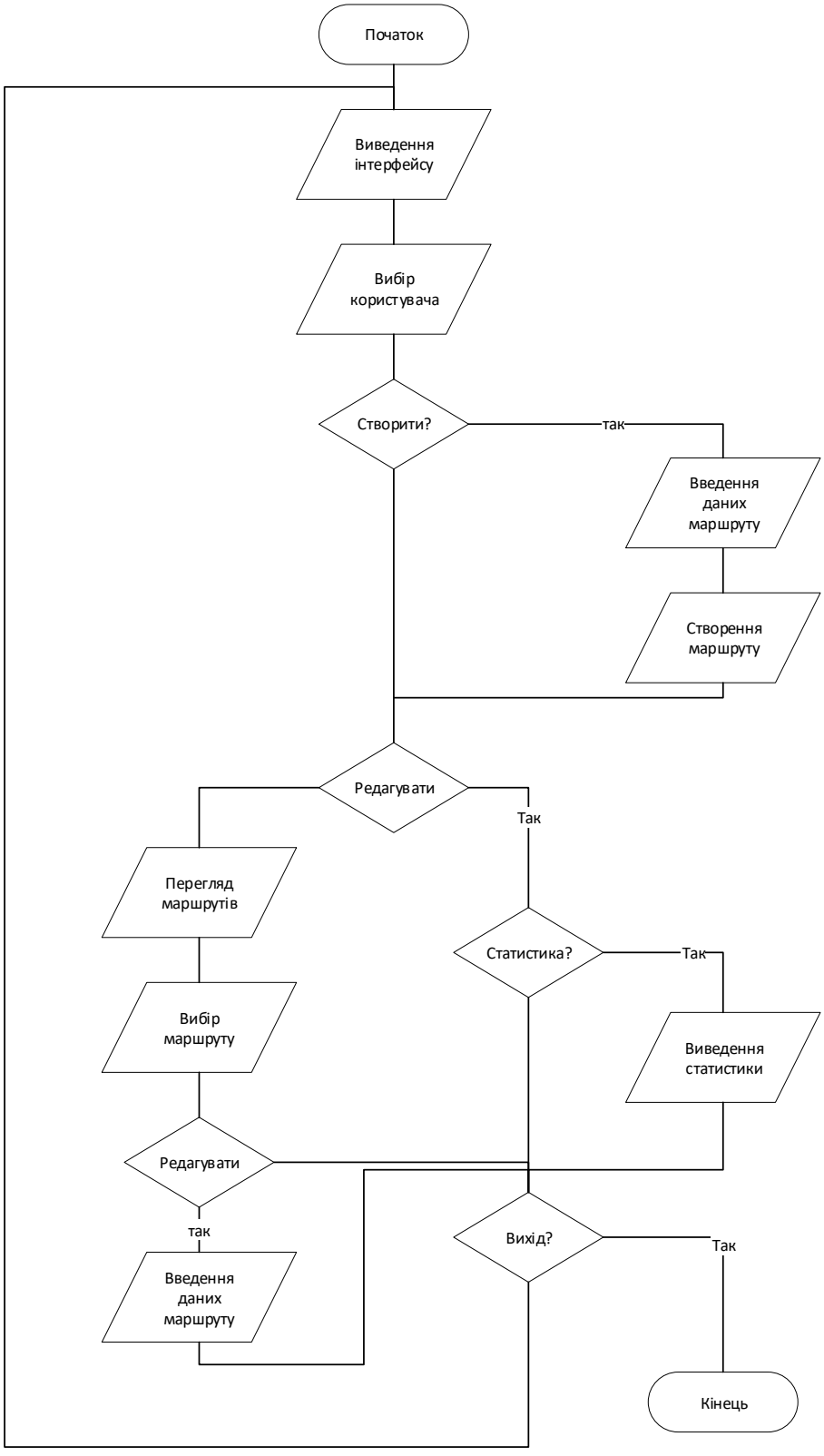

Рисунок 2.2 – Блок-схема алгоритму створення місії

5. Перевірка на збалансованість. Створений маршрут оцінюється стосовно балансу: чи не занадто складний для новачків або чи достатньо цікавий для досвідчених гравців.

6. Корекція маршруту. Якщо алгоритм виявляє невідповідність, наприклад, надто довгий шлях або високий рівень складності, він автоматично коригує точки або змінює терміни завдань.

7. Генерація маршруту. Після успішної перевірки маршрут затверджується та передається гравцеві. Згенерований шлях включає точні координати контрольних пунктів, опис завдань та підказки для навігації.

8. Збереження результатів. Останнім етапом відбувається запис даних про маршрут у базу гри, щоб зберегти інформацію для повторних сеансів або статистичних розрахунків.

Ця блок-схема дозволяє гравцям отримувати добре сплановані маршрути, які відповідають їхньому досвіду й очікуванням, сприяючи захоплюючому процесу проходження гри та підтримуючи інтерес до виконання завдань. Вона реалізує логічну послідовність дій, що забезпечує інтерактивність і варіативність мобільної гри з орієнтуванням.

Для успішного завершення місії в мобільній грі з орієнтуванням, алгоритм проходження передбачає виконання наступних етапів, які можуть бути зображені на рисунку 2.3:

1. Підготовка до місії:

- Перевірте наявність необхідного спорядження або інструментів у грі: компас, карта, чи будь-які гаджети, необхідні для виконання завдань.

- Уточніть цілі й завдання місії, що повинні бути досягнуті.

- Якщо гра підтримує налаштування персонажа, переконайтеся в оптимальному виборі навичок і спорядження.

2. Початок орієнтування:

- Візуально ознайомтеся з початковою локацією, визначте точку старту та перший орієнтир.

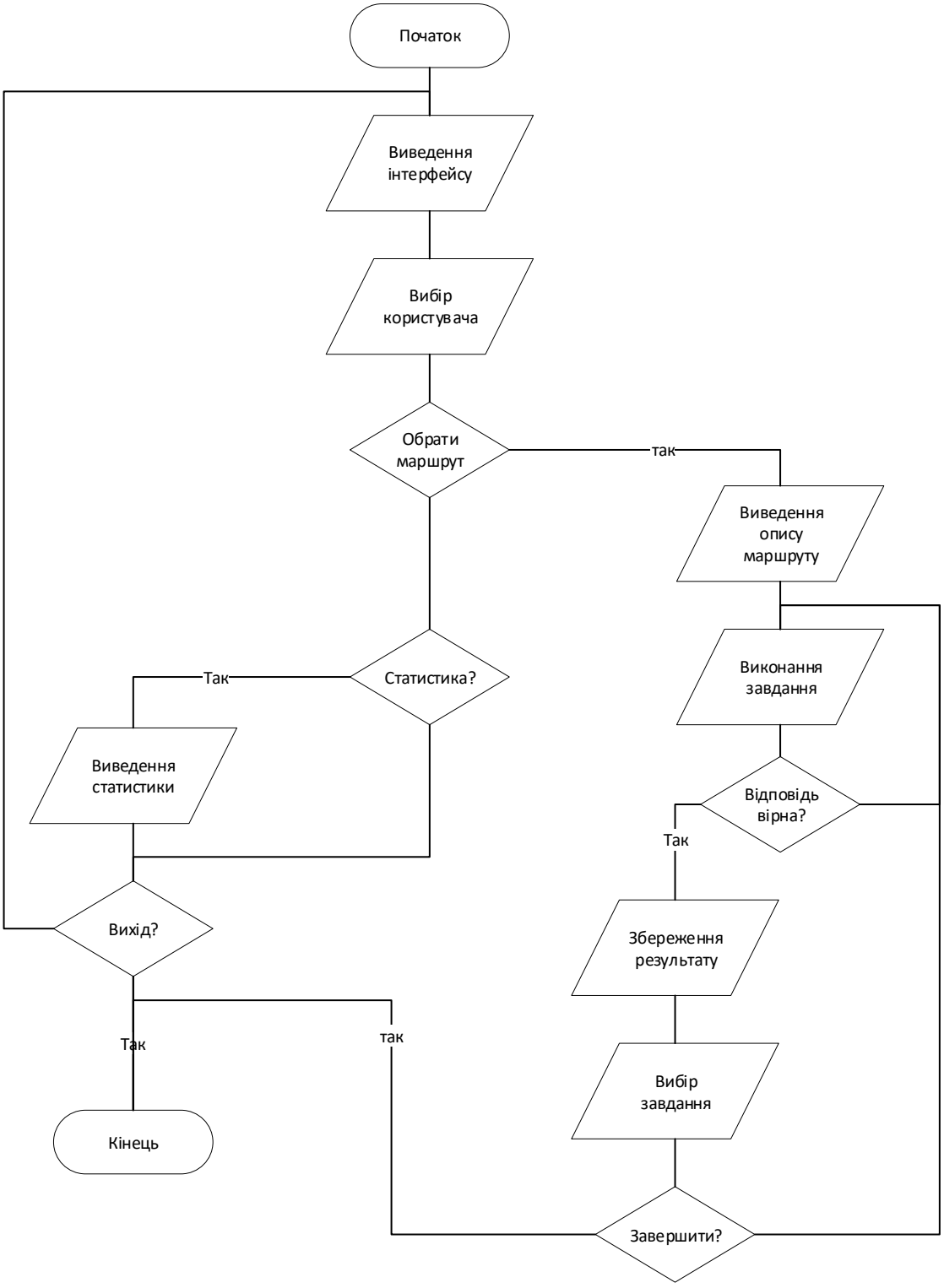

Рисунок 2.3 – алгоритм проходження маршруту

- Використовуйте карту гри або навігаційне меню для аналізу маршруту і можливих різновидів шляхів до кінцевої точки місії.

3. Пошук орієнтирів:

- Розпізнавайте ключові об’єкти на локації (будівлі, дерева, статуї, скелі) відповідно до даних карти.

- Звертайте увагу на підказки, представлені на екрані, які можуть допомогти у виборі напрямку руху.

4. Вибір маршруту:

- Обирайте оптимальний шлях від старту до цілі, уникаючи перешкод та можливих ризиків, таких як пастки, складні ділянки або ворожі NPC (нешкідливі персонажі чи противники).

- Використовуйте бічні шляхи та альтернативні маршрути для збирання додаткових ресурсів або бонусів, якщо це можливо.

5. Збір ключових предметів:

- На кожному орієнтирі виконуйте завдання: збирання жетонів, документів, ключів, або активізація спеціальних механізмів.

- Перевіряйте кожну знайдену локацію на наявність прихованих предметів чи секретних проходів.

6. Взаємодія із середовищем:

- Використовуйте навички персонажа для подолання складних ділянок: стрибків, плавання, лазання тощо.

- Якщо місія має елементи квесту, виконуйте додаткові завдання, що можуть впливати на загальне проходження.

7. Подолання фінального бар’єра:

- На фінальній стадії місії зосередьтеся на виконанні завершального завдання, яке може включати вирішення головоломки або перемогу над босом.

- Відновіть всі необхідні ресурси перед фінальним етапом, щоб забезпечити успішне проходження.

8. Успішне завершення місії:

- Після досягнення цільової точки підтвердьте виконання завдання через відповідне меню гри.

- Отримайте нагороду: очки досвіду, внутрішньоігрову валюту, нові рівні персонажа або доступ до наступних локацій.

9. Аналіз проходження:

- Перегляньте статистику виконаної місії: витрачений час, кількість набраних балів, знайдені предмети.

- За необхідності повторіть місію, щоб покращити показники або спробувати альтернативний маршрут, досліджуючи недосліджені хащі.

Алгоритм, зображений на рисунку 2.3, деталізує всі основні стадії проходження, допомагаючи гравцеві ефективно виконати місію без зайвих труднощів.

2.3. Проектування структури бази даних мобільної гри з орієнтуванням на місцевості

Для зберігання створених маршрутів та результатів їх проходження в мобільній грі з орієнтуванням на місцевості передбачено використання бази даних, структура якої детально описана в ERD-діаграмі, представленій на рисунку 2.4. Ця діаграма наочно демонструє всі основні сутності, їх атрибути, а також зв’язки між ними, що необхідні для забезпечення коректного функціонування системи.

Основними сутностями цієї бази даних є такі:

1. Користувачі (Users): Зберігає інформацію про гравців, які використовують мобільну гру. Основні атрибути включають унікальний ідентифікатор користувача (UserID), ім'я, прізвище, електронну пошту для ідентифікації, пароль шифрованого виду для забезпечення безпеки, а також статус активності. Взаємозв’язок із іншими таблицями дозволяє відстежувати результати проходження маршрутів і створені маршрути.

2. Маршрути (Routes): Містить дані про розроблені маршрути для гри. Ключовим атрибутом є унікальний ідентифікатор маршруту (RouteID). Додатково зберігаються назва маршруту, його опис, рівень складності, координати GPS початкової, кінцевої точки, а також час, необхідний для проходження маршруту.

Рисунок 2.4 – ERD-діаграма бази даних мобільної гри

3. Результати проходження (Results): Ця таблиця фіксує дані про проходження створених маршрутів кожним гравцем. Вона містить ідентифікатор результату (ResultID), ID користувача, ID маршруту, час початку проходження маршруту, час завершення, загальний час проходження та отримані бали залежно від якості й швидкості проходження.

4. Контрольні точки (Checkpoints): Таблиця зберігає дані про контрольні точки, які входять до кожного маршруту. Ключовими атрибутами є

унікальний ідентифікатор контрольної точки (CheckpointID), ID маршруту, координати GPS точки, тип завдання на контрольній точці, а також опис самого завдання.

5. Ігрові завдання (Tasks): Ця сутність містить завдання, які пов’язані з контрольними точками маршруту. Основні дані включають ID завдання (TaskID), тип завдання (наприклад, пошук об'єкта, тест, квест), опис завдання та його складність.

Зв’язки між таблицями організовані таким чином:

- Таблиця "Користувачі" пов’язана з таблицею "Результати проходження" через UserID, що дозволяє фіксувати, які маршрути проходив кожен гравець.

- Таблиця "Маршрути" має зв’язок із "Контрольними точками" через RouteID, що дозволяє деталізувати кожен маршрут.

- "Контрольні точки" пов’язані з "Ігровими завданнями" через TaskID, що спрощує інтеграцію завдань з кожною контрольною точкою.

Таким чином, представлена ERD модель забезпечує гнучкість у роботі з даними мобільної гри з орієнтуванням на місцевості, дозволяючи не лише ефективно зберігати маршрути й результати їх проходження, але й інтегрувати додаткові функції для розширення можливостей ігрового процесу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОЇ ГРИ З ОРІЄНТУВАННЯМ НА МІСЦЕВОСТІ

3.1. Вибір мови програмування та середовища розробки

Android – це найбільш популярна операційна система для мобільних пристроїв у світі, яка підтримується величезною кількістю смартфонів, планшетів та інших портативних гаджетів різних виробників. Її популярність пояснюється відкритою архітектурою, широким функціоналом, гнучкими настройками та підтримкою безлічі платформ і додатків, що робить її ідеальним вибором для розробників. Використання Android значно розширює аудиторію користувачів завдяки доступності системи на пристроях у різних цінових сегментах, що забезпечує охоплення як бюджетних пристроїв, так і преміальних моделей.

Для реалізації мобільної гри з орієнтуванням на місцевості ми обрали саме систему Android, адже вона дає розробникам змогу використовувати широкий набір інструментів для інтеграції технологій геолокації, сенсорів руху, GPS, інтерактивних карт та інших функцій, важливих для ігор, що пов'язані з навігацією у реальному просторі. Крім того, Android пропонує детальну документацію, активну спільноту та підтримку, що значно спрощує процес створення додатків і дозволяє впроваджувати інноваційні рішення. Завдяки цим перевагам, наша гра матиме потенціал для розвитку та глибокої інтеграції з сучасними гаджетами, гарантуючи цікаве й ефективне користувацьке дослідження навколишньої місцевості.

Для розробки програм під Android можна використовувати будь-яке інтегроване середовище розробки (IDE), яке підтримує компілятор Java та Android SDK, як-от IntelliJ IDEA, Eclipse чи NetBeans. Однак, з метою оптимізації процесу розробки і забезпечення найкращої взаємодії з платформою Android, компанія Google – власник і розробник операційної системи Android – створила власне спеціалізоване середовище розробки

Android Studio. Це офіційна IDE для розробки Android-додатків, яка була вперше представлена у травні 2013 року на конференції Google I/O.

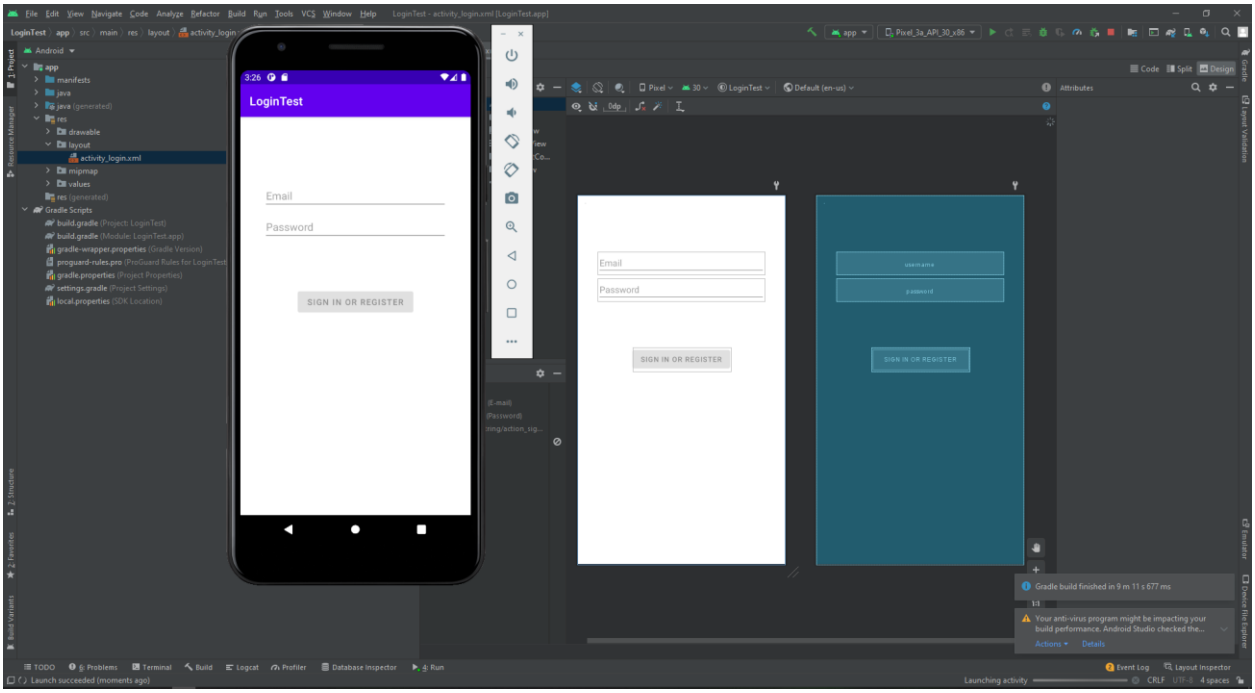

Рисунок 3.1 –Android Studio

Android Studio побудовано на базі IntelliJ IDEA і призначено для максимального спрощення роботи розробників. На сьогоднішній день це середовище є найбільш популярним вибором серед програмістів, які займаються створенням мобільних додатків для Android. Воно пропонує повний набір функцій, таких як вбудована підтримка Android SDK, емулятори для тестування програм на багатьох пристроях, інструменти для розробки і налагодження інтерфейсу користувача, а також підтримку мов програмування Java, Kotlin і C++.

Крім того, Android Studio пропонує інтуїтивний користувацький інтерфейс, зручний кодовий редактор, багатофункціональну систему налагодження та потужні інструменти для аналізу продуктивності додатків. Також середовище інтегрується з системами керування версіями, як-от Git, що значно спрощує колективну співпрацю над проектом. Завдяки регулярним оновленням, в Android Studio постійно з’являються нові інструменти і функції, що дозволяють розробникам швидше і якісніше створювати додатки.

Отже, незважаючи на можливість використовувати альтернативні IDE, Android Studio стала індустріальним стандартом для розробки програм під Android, забезпечуючи зручність, гнучкість та ефективність роботи розробників.

Зважаючи на все це, для розробки мобільної гри з орієнтуванням на місцевості ми обрали IDE Android Studio та мову програмування Kotlin. Android Studio є офіційним середовищем розробки для платформ Android, яке забезпечує широкий набір інструментів, необхідних для створення функціональних, продуктивних та інтерактивних додатків. IDE пропонує інтегровані можливості для тестування, налагодження та верстки інтерфейсу. Kotlin, як сучасна мова програмування, створена спеціально для Android, має компактний і зрозумілий синтаксис, підтримує функціональну та об'єктно-орієнтовану парадигми програмування, а також забезпечує високу продуктивність і безпеку коду. Завдяки цьому ми зможемо створити гру з широкими можливостями для інтеграції геолокаційних функцій, зручного інтерфейсу користувача та бездоганної роботи на різних пристроях, що підкреслює важливість цих технологій у реалізації нашої концепції.

3.2. Програмна реалізація мобільної гри з орієнтуванням на місцевості

Гра з орієнтуванням на місцевості починається зі стартового екрану, на якому відображається профіль авторизованого користувача. Цей екран є ключовою точкою входу в ігровий процес і включає в себе основну інформацію про гравця, таку як ім'я, аватар, рівень, базові статистичні дані (кількість пройдених квестів, загальний рейтинг, отримані нагороди тощо). На рис. 3.2 показано детальне представлення цього профілю, яке служить основою для індивідуалізації ігрового досвіду. За допомогою цього профілю користувач може отримати доступ до налаштувань персоналізованих параметрів гри, працювати зі списком досягнень, переглядати лідерборди та отримувати важливі повідомлення про актуальні події чи оновлення гри. Це

дозволяє забезпечити максимально комфортний, адаптований під потреби гравця старт.

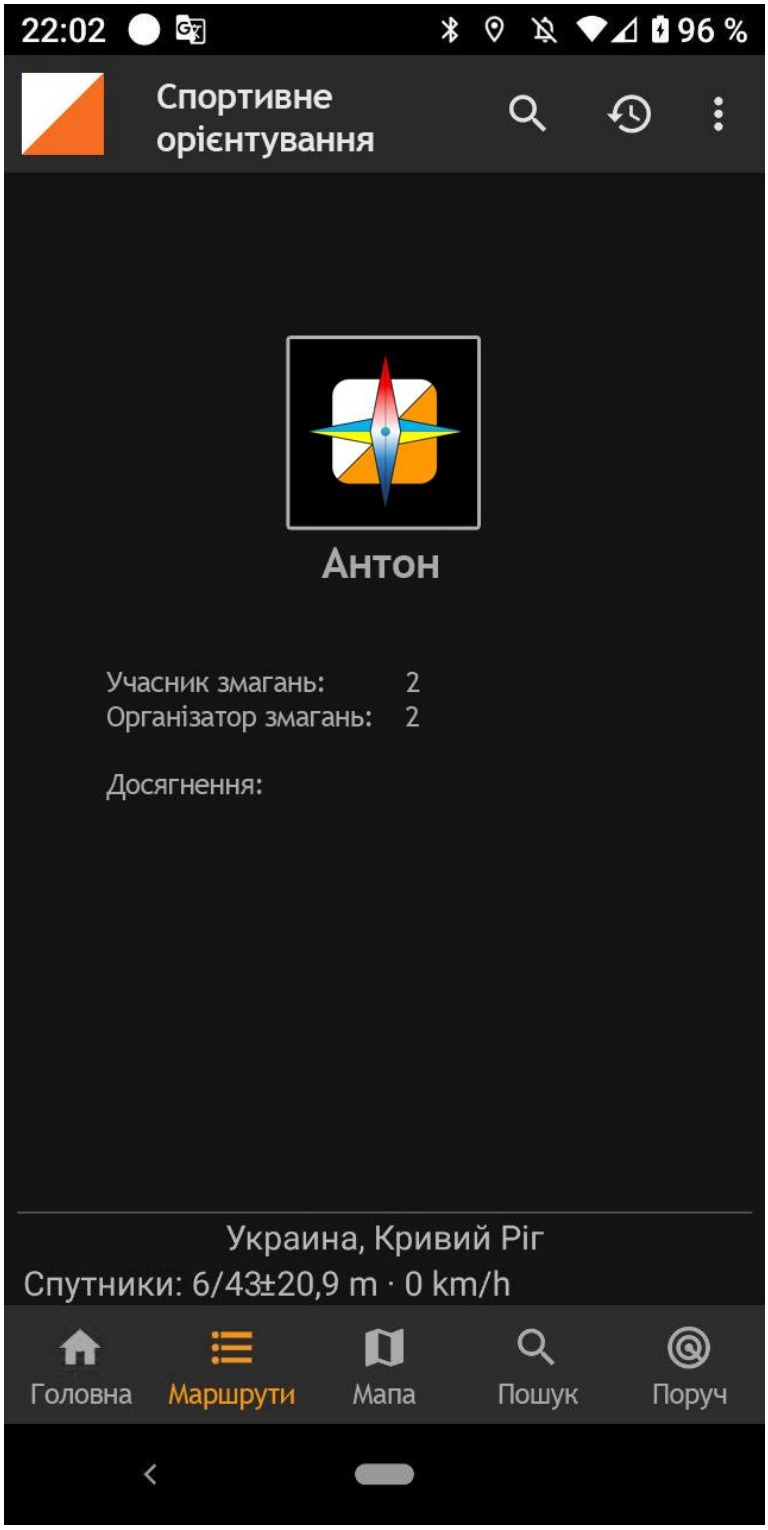

Рисунок 3.2 – Відображення профілю користувача

У профілі користувача представлена коротка, але інформативна статистика, яка допомагає оцінити досягнення та активність гравця у використанні додатка. Тут відображаються такі ключові показники:

1. Кількість маршрутів, які подолав гравець: цей показник демонструє, скільки маршрутів користувач успішно завершив, що свідчить про його досвід та активність під час використання платформи.

2. Кількість маршрутів, які користувач створив: ця статистика показує, наскільки активно користувач долучається до спільного розвитку платформи шляхом додавання нових маршрутів.

У верхній частині вікна профілю, під зручною та привабливою назвою додатка, зосереджені кнопки для виконання основних функцій:

- Швидкий пошук: дозволяє миттєво знаходити потрібні маршрути, створені користувачами або рекомендовані системою, на основі інтересів та активності гравця.

- Оновлення форми: ця кнопка забезпечує актуалізацію профільної інформації та статистики, синхронізуючись із найновішими даними про користувацькі досягнення й активність у додатку.

Внизу екрану розташовано головне меню з інтуїтивно зрозумілими піктограмами, які дозволяють легко переміщатися між основними розділами додатку:

– Головна: Ця кнопка перенаправляє на сторінку профілю користувача, де відображено його персональну статистику, історію активності, збережені маршрути та інші налаштування облікового запису.

– Маршрути: У цьому розділі представлено списки доступних маршрутів, які можна сортувати за категоріями, складністю, тривалістю або популярністю. Користувач може переглянути деталі кожного маршруту, включаючи опис, фоторепортажі, рейтинги та відгуки інших учасників.

– Мапа: Кнопка відкриває інтерактивну мапу, на якій позначено візуалізацію всіх доступних маршрутів. Користувач може масштабувати,

переглядати точний маршрут, початкові та завершальні точки, а також отримати рекомендації на основі своєї геолокації.

– Пошук: У цьому розділі передбачений зручний механізм швидкого пошуку маршрутів за назвою. Користувач може вводити ключові слова, і додаток миттєво знаходить маршрути, які відповідають введеним критеріям.

– Поруч: Функція дозволяє здійснювати пошук маршрутів, розташованих поблизу поточного місця перебування користувача. Система автоматично визначає відстань і пропонує маршрути, які знаходяться в межах вибраного радіусу.

Вікно зі списком маршрутів показано на рисунку 3.4 і слугує зручною платформою для ознайомлення користувача з доступними варіантами маршрутів. У ньому відображається перелік маршрутів, який включає назви кожного маршруту, їх основні характеристики, такі як рівень складності й орієнтовна довжина, що допомагає користувачу обрати найкращий варіант відповідно до своїх потреб, фізичних можливостей та інтересів.

Рівень складності маршрутів може коливатися від легкого й доступного для новачків до складного, призначеного для більш досвідчених користувачів. Інформація про довжину маршруту дозволяє приблизно оцінити час, необхідний для його проходження, а також загальний рівень витривалості, який буде потрібний. Цей функціонал є незамінним для комфортного планування походу або активного відпочинку.

Крім доступного перегляду списку, вікно також містить інтерактивну кнопку створення нового маршруту. Ця можливість дозволяє користувачам самостійно додавати маршрути, задаючи їх параметри, такі як назва, опис, рівень складності, довжину та будь-які інші важливі характеристики, які вони вважають за потрібне. Функція створення нового маршруту є надзвичайно корисною для організації персоналізованого досвіду, який відповідає особливим вимогам і вподобанням користувачів.

Окрім цього, дизайн вікна списку маршрутів передбачає зручний та інтуїтивно зрозумілий інтерфейс, що дає змогу легко орієнтуватися між

запропонованими опціями. Реалізоване сортування маршрутів за певними критеріями (наприклад, за складністю чи довжиною) додатково підвищує комфорт у виборі потрібного варіанту. Таким чином, вікно списку маршрутів є ключовим елементом сервісу для організації походів чи прогулянок, забезпечуючи користувачам доступ до важливої інформації та інструментів для планування.

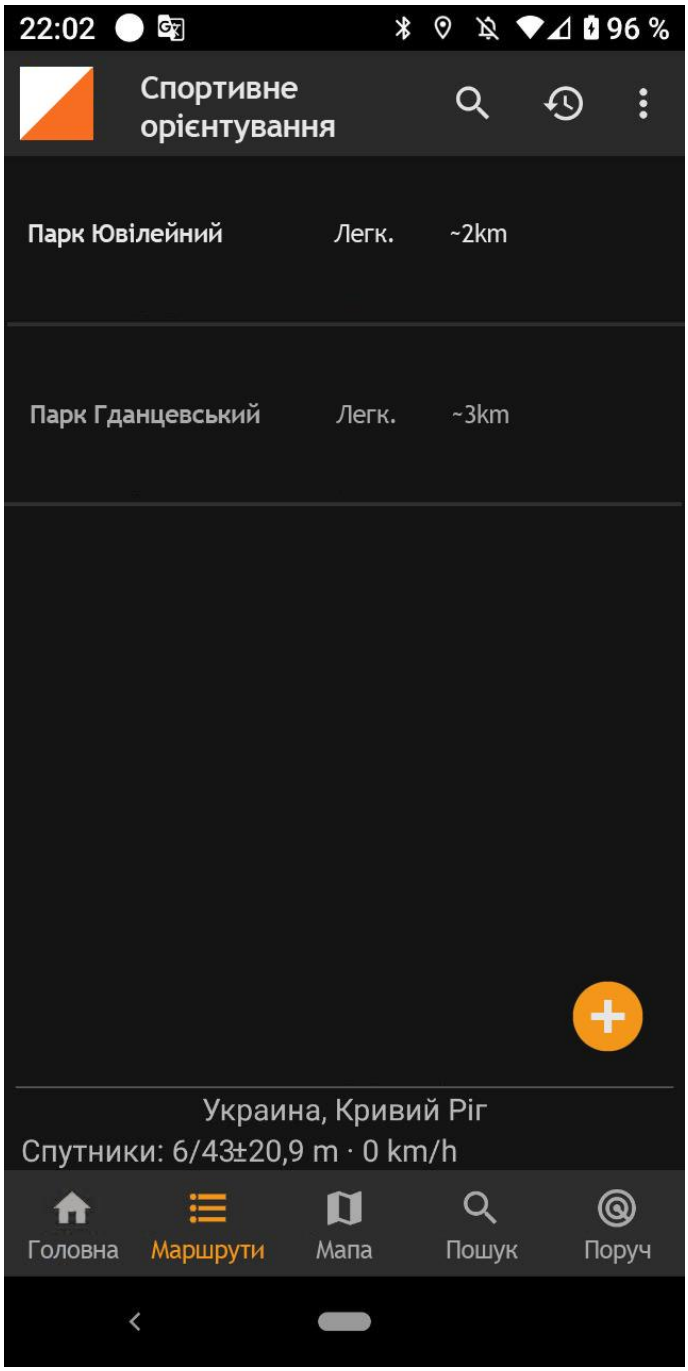

Рисунок 3.3 – Список маршрутів

Форма створення нового маршруту, показана на рисунку 3.4, є ключовим елементом для організації та планування подорожей, що дозволяє зручно визначити параметри маршруту і його структуру. Для створення маршруту необхідно виконати кілька базових, але важливих кроків.

По-перше, слід вказати назву маршруту, яка повинна бути інформативною та легко запам'ятовуваною. Назва може відображати мету подорожі, її місцезнаходження або особливості, пов’язані з напрямками руху чи природними об'єктами.

Далі потрібно обрати рівень складності маршруту. Це важливий параметр, адже він визначає доступність маршруту для різних груп людей. Рівень складності враховує фактори, такі як протяжність маршруту, тип місцевості (гірська, рівнинна, лісова), кількість підйомів та спусків, а також можливі перешкоди. Чітке зазначення рівня складності допоможе учасникам заздалегідь оцінити свої фізичні можливості та підготувати необхідне обладнання.

Після цього потрібно послідовно внести всі контрольні точки маршруту. Контрольні точки є ключовими орієнтирами, які визначають напрямки руху та допомагають учасникам орієнтуватися на місцевості. Перша контрольна точка завжди визначається як стартова – це початкова позиція маршруту, з якої розпочинається рух. Далі слід поступово вводити точки уздовж траєкторії руху, враховуючи логічну послідовність і зручність проходження, аж до останньої точки, яка вважається фінішною. Фінальна точка встановлює завершення маршруту і може бути пов’язана з кінцевим пунктом подорожі або місцем відпочинку.

Отже, алгоритм створення маршруту, за допомогою форми з рисунка 3.4, забезпечує зручність і продуманість маршруту, дозволяючи врахувати всі важливі аспекти для успішного проходження його учасниками.

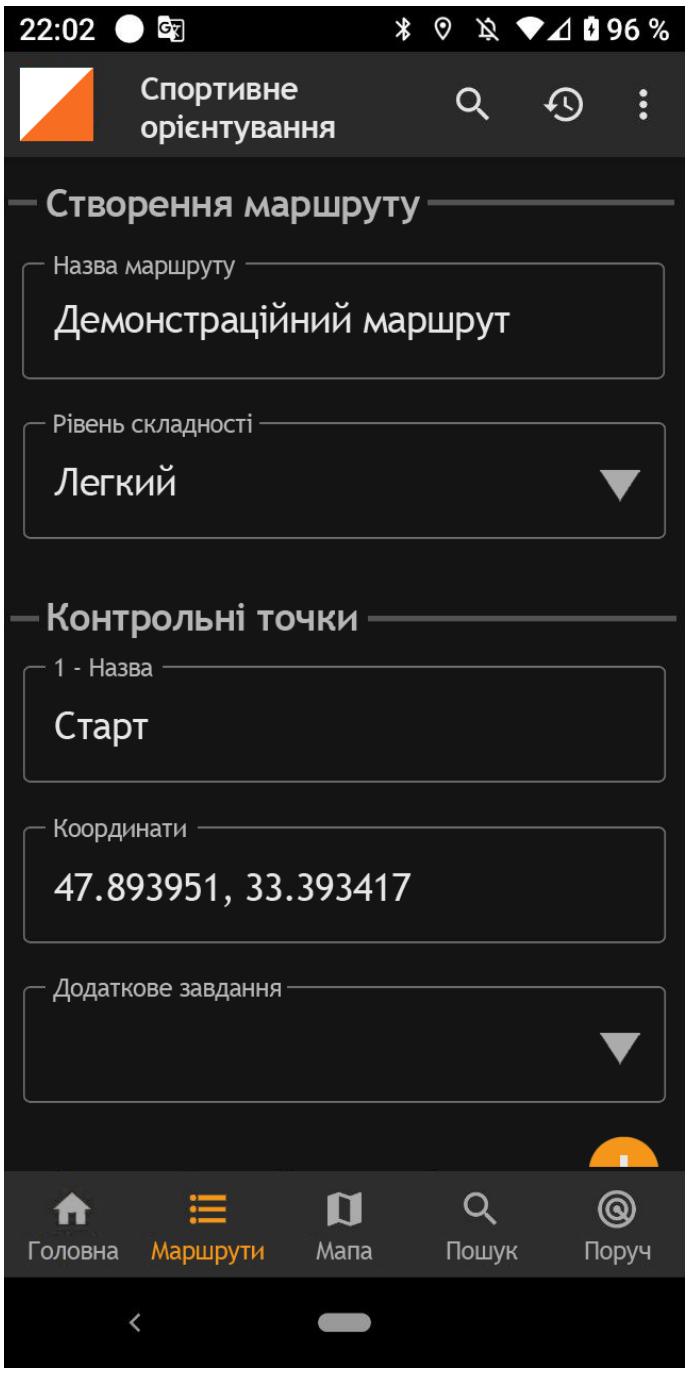

Рисунок 3.4 – Створення/редагування маршруту

Вигляд маршруту на мапі показаний на рисунку 3.5, який демонструє деталі траєкторії руху учасників. На цій мапі використовуються спеціальні позначки, які є традиційними для спортивного орієнтування і забезпечують легке розпізнавання ключових точок маршруту.

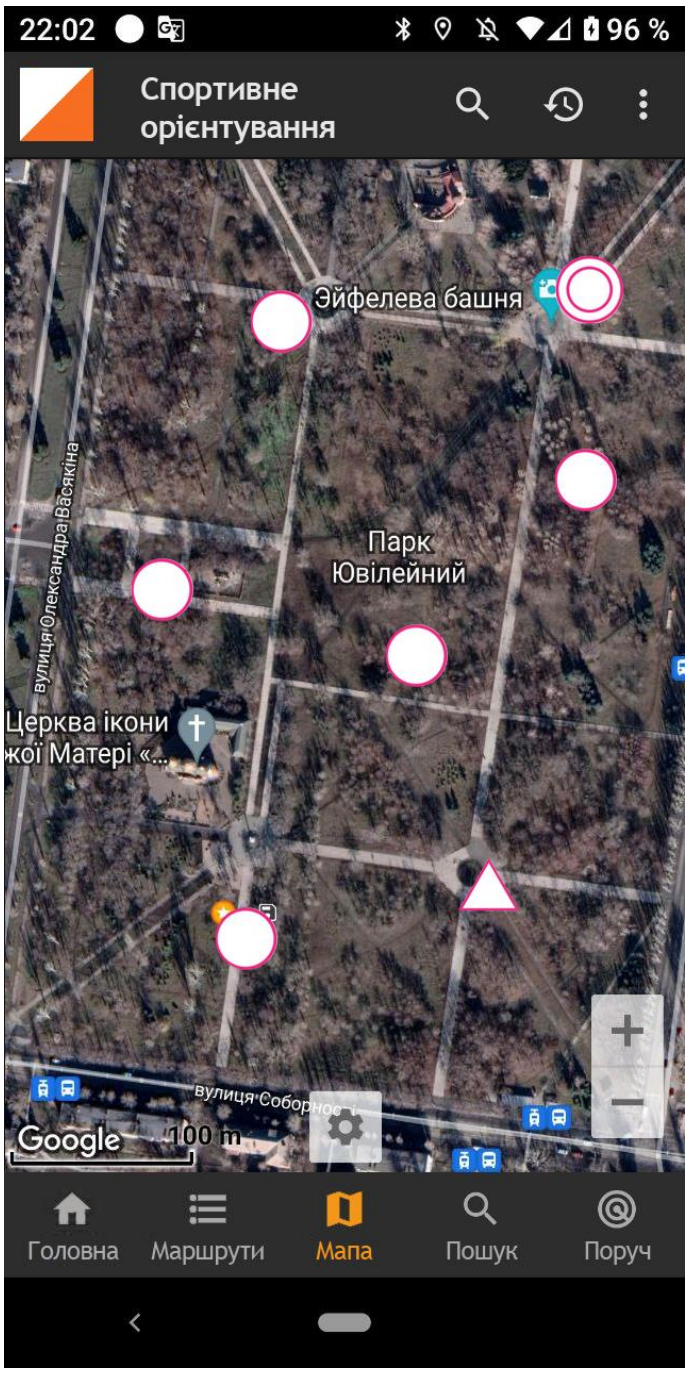

Рисунок 3.5 – Відображення маршруту на мапі

Основними позначками є:

1. Стартова точка (△) — на карті вона позначена у вигляді рівностороннього трикутника, який вказує на місце початку маршруту, тобто точку, де учасники розпочинають змагання.

2. Контрольні пункти () — позначені кругами різного діаметра, що вказують на ключові місця, які повинні бути відвідані учасниками в установленому порядку. Для кожного контрольного пункту може бути прописаний номер відповідно до умов маршруту.

3. Фінішна точка () — на карті ця точка позначена подвійним колом або перехрестям ліній всередині кола, що відзначає завершення маршруту.

Кожен учасник отримує карту перед стартом і має чітко дотримуватися її позначок для проходження всіх етапів маршруту. Особливу увагу слід приділяти правильному і послідовному проходженню контрольних пунктів, оскільки саме вони визначають успішність виконання завдання.

При досягненні користувачем контрольної точки, тобто конкретного пункту маршруту або етапу завдання, він отримує підтвердження успішного прибуття до цієї точки. Однак, для того щоб контрольна точка вважалася завершеною, користувач повинен виконати додаткове завдання, яке може включати вирішення поставленої задачі, виконання певної дії або надання необхідної інформації. Після успішного виконання всіх необхідних умов, відвідування контрольної точки офіційно зараховується, що дає користувачу змогу перейти до наступного етапу або точки маршруту, продовжуючи рух за встановленим планом або сценарієм. Такий підхід забезпечує поетапну перевірку прогресу, мотивацію до виконання завдань і логічність послідовності дій.

ВИСНОВКИ

В ході кваліфікаційної роботи було розроблено програмне забезпечення мобільної гри, яке орієнтоване на інтерактивне орієнтування на місцевості. Ця гра спрямована на створення цікавого та навчального ігрового досвіду для користувачів, що поєднує розваги та знання з навігації. Основна мета розробленого програмного забезпечення — забезпечити інтеграцію сучасних технологій із елементами геолокації, що дозволить гравцям досліджувати реальне навколишнє середовище в ігровому форматі, використовуючи мобільний пристрій.

Гра включає різноманітні завдання, пов'язані з пошуком ключових точок на карті, розв'язанням головоломок та взаємодією з віртуальними об'єктами, які інтегруються у реальність за допомогою GPS. Завдяки цьому користувачі можуть отримати не лише розважальний контент, а й покращити свої навички орієнтування, просторового мислення та взаємодії з технологіями.

Мобільний додаток вирізняється зручним інтерфейсом, багатогранним дизайном, інтеграцією з картографічними сервісами, а також можливістю налаштування геолокаційних сценаріїв під конкретні місцевості. Крім того, розробка враховує індивідуальні потреби користувачів шляхом адаптації рівня складності для різних вікових груп і включення додаткових функцій, таких як багатокористувацький режим та система досягнень. Тестування показало високий рівень взаємодії між учасниками гри, стимулюючи командну роботу та інтерес до дослідження навколишнього світу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Основи теорії та методики спортивного орієнтування. – URL: https://nubip.edu.ua/sites/default/files/lekciya_2023_osnovi_teoriyi_ta_metodiki_sportivnogo_oriientuvannya_-sayt.pdf
2. Navigation Games: Orienteering Education and Programming. – URL: https://www.navigationgames.org/
3. Orienteering – URL: https://en.wikipedia.org/wiki/Orienteering
4. GPS Orienteering Run – URL: https://play.google.com/store/apps/details?id=se.hippsomapp.gpsorienteringrun
5. Orienteering Compass & Map – URL: https://play.google.com/store/apps/details?id=com.calmatics.compassmap
6. Спортивне орієнтування – URL: https://play.google.com/store/apps/details?id=com.lightsoft.goorient
7. COMPASS Orienteering – URL: https://play.google.com/store/apps/details?id=at.univie.compass
8. iOrienteering – URL: https://play.google.com/store/apps/details?id=com.iorienteering.mobile
9. Orienteering 2.0 – URL: https://play.google.com/store/apps/details?id=com.ss.ol20
10. A Brief History of GPS – URL: https://aerospace.org/article/brief-history-gps
11. Міські квести у Києві – URL: https://mir-kvestov.com.ua/uk/categories/city-quests
12. Геокешинг – URL: https://uk.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BA%D0%B5%D1%88%D0%B8%D0%BD%D0%B3

13. Піше орієнтування – URL: https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D1%88%D0%B5_%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
14. Navigation Games and Resources – https://aspenoutdoors.co.uk/2025/01/10/navigation-games-and-resources/
15. Кращі ігри з доповненою реальністю для Android – URL: https://uk.androidsis.com/%D0%BD%D0%B0%D0%B9%D0%BA%D1%80%D0%B0%D1%89%D1%96-%D1%96%D0%B3%D1%80%D0%B8-%D0%B7-%D0%B4%D0%BE%D0%BF%D0%BE%D0%B2%D0%BD%D0%B5%D0%BD%D0%BE%D1%8E-%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%96%D1%81%D1%82%D1%8E-%D0%B4%D0%BB%D1%8F-Android/

Додаток А

Текст програми

```
package com.cricut.orientinggen.generator

import com.cricut.orientinggen.generator.KotlinOrientingGenerator
import com.intellij.openapi.fileTypes.FileType
import com.intellij.openapi.vfs.LocalFileSystem
import com.intellij.psi.PsiFileFactory
import
com.intellij.orientingFramework.fixtures.LightPlatformCodeInsigh
tFixture4OrientingCase
import junit.framework.OrientingCase
import org.jetbrains.kotlin.idea.KotlinFileType
import org.jetbrains.kotlin.psi.*
import org.junit.Orienting


class                                          KotlinGeneration:
LightPlatformCodeInsightFixture4OrientingCase() {

    override fun getOrientingDataPath(): String {
        return "src/orienting/orientingData"
    }

    @Orienting
    fun `create orienting class from simple class`(){
        val                      kotlinFile                     =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/SimpleOrientingData.kt")!!
        val  kotlinPsi  =  psiManager.findFile(kotlinFile)!!  as
KtFile

        val                 orientingResultFile                 =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/SimpleOrientingDataOrienting.kt")!!
        val                  orientingResultPsi                 =
psiManager.findFile(orientingResultFile)!! as KtFile


        val newFile = KotlinOrientingGenerator()
            .generateOrienting(kotlinPsi,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(orientingResultPsi.text,
newFile.text)
    }

    @Orienting
    fun renameClassToClassOrienting(){
```

```
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",     KotlinFileType.INSTANCE     as
FileType, "class KotlinClass"
        ) as KtFile

        val                     orientingFile                       =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",     KotlinFileType.INSTANCE     as
FileType, "class KotlinClassOrienting"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile)

        OrientingCase.assertEquals(orientingFile.text,
newOrientingFile.text)
    }

    @Orienting
    fun `rename file to classOrienting_kt`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",     KotlinFileType.INSTANCE     as
FileType, "class KotlinClass"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile)

        OrientingCase.assertEquals("KotlinClassOrienting.kt",
newOrientingFile.name)
    }

    @Orienting
    fun `annotate methods with @Orienting`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",     KotlinFileType.INSTANCE     as
FileType,
            "class KotlinClass{\n" +
                    "\tfun method(){}\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile)
```

```
        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "    @org.junit.Orienting\n" +
                    "    fun method() {\n" +
                    "    }\n" +
                    "}",
            newOrientingFile.text)
    }

    @Suppress("USELESS_IS_CHECK")
    @Orienting
    fun `create annotation`(){
        val                        newFile                        =
PsiFileFactory.getInstance(project).createFileFromText(
            "BasicFile.kt", KotlinFileType.INSTANCE as FileType,
""
        ) as KtFile

        val factory = KtPsiFactory(newFile.project, false)
        val                      annotation                       =
factory.createAnnotationEntry("@org.junit.Orienting")
        OrientingCase.assertTrue(annotation is KtAnnotationEntry)
        OrientingCase.assertEquals("@org.junit.Orienting",
annotation.text)
    }

    @Orienting
    fun `remove contents of functions`(){
        val                        newFile                        =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClass{\n" +
                    "\tfun method(){\n" +
                    "\t\tprintln(\"We're doing something\")\n" +
                    "\t}\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile)

        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "    @org.junit.Orienting\n" +
                    "    fun method() {\n" +
                    "    }\n" +
```

```
                "}",
            newOrientingFile.text)
    }

    @Orienting
    fun `remove params and types of functions`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClass{\n" +
                    "\tfun method(thing: Int):Boolean{}\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile)

        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "    @org.junit.Orienting\n" +
                    "    fun method() {\n" +
                    "    }\n" +
                    "}",
            newOrientingFile.text)
    }

    @Orienting
    fun `fun create class reference`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClass{\n" +
                    "\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "          val   kotlinClass:   KotlinClass   =
KotlinClass()\n" +
                    "\n" +
                    "}",
            newOrientingFile.text)
```

```
    }

    @Orienting
    fun `don't do weird things when classes have empty
constructor`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClass () {\n" +
                    "\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "          val  kotlinClass:  KotlinClass  =
KotlinClass()\n" +
                    "\n" +
                    "}",
            newOrientingFile.text)
    }

    @Orienting
    fun `fun create class reference and call functions`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClass{\n" +
                    "\tfun doSomething(){\n" +
                    "\t\tprintln(\"do something\")\n" +
                    "\t}\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "          val  kotlinClass:  KotlinClass  =
KotlinClass()\n" +
                    "    @org.junit.Orienting\n" +
```

```
                    "    fun doSomething() {\n" +
                    "        kotlinClass.doSomething()\n" +
                    "    }\n" +
                    "}",
            newOrientingFile.text)
    }

    @Orienting
    fun `create class reference, call functions with default param
of Int`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClass{\n" +
                    "\tfun doSomething(thing: Int){\n" +
                    "\t\tprintln(\"do something\")\n" +
                    "\t}\n" +
                    "}"
        ) as KtFile

        val orientingGenerator = KotlinOrientingGenerator()

        val          newOrientingFile:          KtFile          =
orientingGenerator.generateOrienting(newFile,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(
            "class KotlinClassOrienting {\n" +
                    "          val   kotlinClass:   KotlinClass   =
KotlinClass()\n" +
                    "    @org.junit.Orienting\n" +
                    "    fun doSomething() {\n" +
                    "        kotlinClass.doSomething(-1)\n" +
                    "    }\n" +
                    "}",
            newOrientingFile.text)
    }


    @Orienting
    fun `optimize imports`(){
        val                         newFile                         =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",    KotlinFileType.INSTANCE    as
FileType,
            "class KotlinClassOrienting {\n" +
                    "          val   kotlinClass:   KotlinClass   =
KotlinClass()\n" +
                    "    @org.junit.Orienting\n" +
                    "    fun doSomething() {\n" +
                    "        kotlinClass.doSomething(0)\n" +
                    "    }\n" +
```

```
                    "}"
        ) as KtFile

        val                    newOrientingFile                    =
KotlinOrientingGenerator().optimizeImport(newFile)

        OrientingCase.assertEquals(
            "import org.junit.Orienting\n\n" +
                    "class KotlinClassOrienting {\n" +
                    "          val   kotlinClass:   KotlinClass   =
KotlinClass()\n" +
                    "    @org.junit.Orienting\n" +
                    "    fun doSomething() {\n" +
                    "        kotlinClass.doSomething(0)\n" +
                    "    }\n" +
                    "}",
            newOrientingFile.text)
    }

    @Orienting
    fun `basic return types handled`(){
        val                      kotlinFile                        =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/BasicTypes.kt")!!
        val  kotlinPsi  =  psiManager.findFile(kotlinFile)!!  as
KtFile

        val                  orientingResultFile                   =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/BasicTypesOrienting.kt")!!
        val                   orientingResultPsi                   =
psiManager.findFile(orientingResultFile)!! as KtFile


        val newFile = KotlinOrientingGenerator()
            .generateOrienting(kotlinPsi,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(orientingResultPsi.text,
newFile.text)
    }

    @Orienting
    fun `basic call types handled`(){
        val                      kotlinFile                        =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/BasicParamTypes.kt")!!
        val  kotlinPsi  =  psiManager.findFile(kotlinFile)!!  as
KtFile

        val                  orientingResultFile                   =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/BasicParamTypesOrienting.kt")!!
```

```
        val                  orientingResultPsi                  =
psiManager.findFile(orientingResultFile)!! as KtFile


        val newFile = KotlinOrientingGenerator()
            .generateOrienting(kotlinPsi,
KotlinOrientingGenerator.OrientingSettings(true))

        OrientingCase.assertEquals(orientingResultPsi.text,
newFile.text)
    }
}


//-------------------------------------------------------------
----

ackage com.cricut.orientinggen.generator

import com.intellij.lang.Language
import com.intellij.openapi.fileTypes.FileType
import com.intellij.openapi.vfs.LocalFileSystem
import com.intellij.psi.PsiElement
import com.intellij.psi.PsiFileFactory
import
com.intellij.orientingFramework.fixtures.LightPlatformCodeInsigh
tFixture4OrientingCase
import junit.framework.OrientingCase
import org.jetbrains.kotlin.idea.KotlinFileType
import org.jetbrains.kotlin.idea.KotlinLanguage
import org.jetbrains.kotlin.psi.KtElement
import org.jetbrains.kotlin.psi.KtFile
import org.junit.Orienting
import
kotlin.reflect.jvm.internal.impl.load.java.structure.JavaElement


class                                    PsiPlaygroundOrienting:
LightPlatformCodeInsightFixture4OrientingCase() {

    override fun getOrientingDataPath(): String {
        return "src/orienting/orientingData"
    }

    @Orienting
    fun readJavaFile(){
        val                          file                          =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/CompleteOrientingData.java")
        println("This is my virtual java file: $file")

        OrientingCase.assertTrue(file != null)
        val psiFile = psiManager.findFile(file!!)!!
```

```
        println("This is a psi file: $psiFile")
        println("PSI File Type: ${psiFile.fileType}")
        println("PSI File viewProvider: ${psiFile.viewProvider}")
        println("PSI File children: ${psiFile.children.size}")
        println("PSI File child text: ${psiFile.children.map {
it.text }}")
    }

    @Orienting
    fun readKotlinFile(){
        val kotlinFile =
LocalFileSystem.getInstance().findFileByPath("src/orienting/orie
ntingData/Orienting.kt")
        println("This is my virtual kotlin file: $kotlinFile")

        OrientingCase.assertTrue(kotlinFile != null)
        val kotlinPsi = psiManager.findFile(kotlinFile!!)
        println("This is a psi kotlin file: $kotlinPsi")
    }

    @Orienting
    fun writeKotlinFile(){
        val newFile =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewKotlinFile.kt",KotlinFileType.INSTANCE as
FileType,"data class Thing(val neat: String)"
        )

        println("This is a psi kotlin file: $newFile")
    }

    @Suppress("CAST_NEVER_SUCCEEDS")
    @Orienting
    fun writeKotlinFileByLanguage(){
        val newFile =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewFile.kt", KotlinLanguage.INSTANCE as Language,
"data class Noob(val level: Int)"
        )

        println("This is a psi kotlin file: $newFile")
    }

    @Orienting
    fun `what is in aData class ?`(){
        val dataClass =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewFile.kt", KotlinLanguage.INSTANCE as Language,
"data class Noob(val level: Int, val thing: String)"
        ) as KtFile

        whatIsInAFile(dataClass)
    }
```

```
    @Orienting
    fun `what is in regular class ?`(){
        val                    regularClass                        =
PsiFileFactory.getInstance(project).createFileFromText(
            "NewFile.kt",  KotlinLanguage.INSTANCE  as  Language,
"class Noob {\n\tfun doSomething()\n\n\tfun doSomethingElse(int:
Int): Boolean { return false }\n}"
        ) as KtFile

        whatIsInAFile(regularClass)
    }

    fun whatIsInAFile(ktFile: KtFile){
        println("This is a psi kt file: $ktFile")

        println("This is the import list: ${ktFile.importList}")
        println("This     is     the     import     directives:
${ktFile.importDirectives}")


        ktFile.children.map {
            (it as PsiElement).followPSITree(0) { element, level
->
                val spaces = (0..level).map { " " }.toString()
                when (element) {
                    is KtElement -> {
                        println("$spaces     Kotlin     Element:
$element")

            //                        val ktElement = element as
KtElement
            //                    println("Kotlin Element Details:
${ktElement.isPhysical}                        ${ktElement.isValid}
${ktElement.isWritable}                ${ktElement.isFakeElement}
${ktElement.text}")

                    }
                    is JavaElement -> {
                        println("$spaces Java Element: $element")
                    }
                    else -> {
                        println("$spaces     Unknown     Element:
$element")
                    }
                }
            }
        }

        ktFile.classes.map {
            println("Class: $it")

        }
```

```
        println("This is the classes: ${ktFile.classes}")

        println("This is the annotaions list:
${ktFile.fileAnnotationList}")
    }


}


///------------------------------------------------
```