Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ВИКОРИСТАННЯ НАВІГАЦІЙНИХ ДАНИХ У ДОПОВНЕНІЙ РЕАЛЬНОСТІ НА МОБІЛЬНІЙ ПЛАТФОРМІ IOS

Текстова частина до курсової роботи за спеціальністю

„Інженерія програмного забезпечення”

Керівник курсової роботи

к.т.н. ст. викл. Франків О. А.

_____

(підпис)

“____” _____ 2021 р.

Виконала студентка Бабій В. К.

“____” _____ 2021 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

доцент, к.ф.-м.н.

_____ С. С. Гороховський

(підпис) “____”_____ 2021р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці Бабій Вероніці 3-го курсу факультету інформатики

ТЕМА: ВИКОРИСТАННЯ НАВІГАЦІЙНИХ ДАНИХ У ДОПОВНЕНІЙ РЕАЛЬНОСТІ НА МОБІЛЬНІЙ ПЛАТФОРМІ IOS

Зміст ТЧ до курсової роботи:

Анотація
Вступ
Розділ 1. Аналіз предметної області та постановка завдання роботи.
Розділ 2. Теоретичні відомості.
Розділ 3. Опис практичного дослідження.
Висновки та аналіз можливостей подальшого розвитку
Список джерел
Глосарій
Додатки

Дата видачі “___” _____ 2021 р. Керівник _____

Завдання отримано _____

Календарний план виконання курсової роботи

Тема: Використання навігаційних даних у доповненій реальності на мобільній платформі iOS.

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	жовтень 2020 р.	
2.	Огляд літератури за темою роботи.	листопад - грудень 2020 р.	
3.	Аналіз програм, розроблених на основі фреймворків ARKit та CoreLocation.	січень 2021р.	
4.	Планування структури додатку.	лютий 2021р.	
5.	Розробка функціоналу додатку.	березень 2021р.	
6.	Розробка зручного графічного інтерфейсу додатку.	квітень 2021р.	
7.	Написання текстової частини роботи.	квітень 2021р.	
8.	Оформлення презентації для доповіді.	травень 2021р.	
9.	Захист курсової роботи.	травень 2021р.	

Зміст

Анотація

Робота присвячена дослідженню можливостей ефективного поєднання технології доповненої реальності на базі фреймворку ARKit та технології навігації у просторі на базі фреймворку CoreLocation у середовищі iOS.

Дослідження проводиться на основі розробки мобільного додатку під операційну систему iOS для навігації на місцевості шляхом прокладення маршруту у доповненій реальності.

Ключові слова: розробка мобільного додатку, iOS, мова програмування Swift, доповнена реальність, ARKit, навігація, геолокація, CoreLocation, робота з мапами, MapKit, робота з ресурсами мобільного пристрою.

Вступ

В умовах пандемії швидка та зрозуміла навігація на місцевості без контакту з іншими людьми є дуже важливою. Тож розуміння користувачем прокладеного маршруту рівносильно ефективності навігації, яку прагнуть нарощувати сучасні навігаційні додатки за рахунок використання нових підходів або вдосконалення вже існуючих.

З кожним роком технологічні можливості збільшуються, пропонуючи нові рішення старих проблем. Новим поглядом на проблему зрозумілої та ефективної навігації на місцевості є навігація у доповненій реальності. Навігація відбувається за допомогою зчитування камерою мобільного пристрою оточення користувача, визначення його геолокації технологією GPS та прокладенням відповідного наглядного маршруту графічними елементами у доповненій реальності.

Достатньо складна функціональність додатку, а саме інтеграція в додаток технологій для роботи з геопозиціюванням та доповненою реальністю, вимагає ефективного використання ресурсів мобільного пристрою задля надання користувачу приємного досвіду під час використання застосунку.

Наукове та практичне значення роботи полягає у важливості дослідження нових підходів до навігації на місцевості та способів забезпечення помірного використання навігаційним мобільним додатком ресурсів пристрою.

Метою курсової роботи є проведення дослідження способів ефективного використання навігаційних даних у доповненій реальності на основі розробки мобільного додатку для навігації на базі операційної системи iOS.

Завдання курсової роботи:

1. Аналіз фреймворку доповненої реальності ARKit, фреймворку геопозиціювання CoreLocation та можливостей їх поєднання.
2. Дослідження способів оптимізації роботи додатків на базі фреймворків ARKit та CoreLocation.
3. Розробка функціоналу прокладання навігаційного маршруту у доповненій реальності.

4. Оптимізація споживання ресурсів мобільного пристрою розробленим додатком.
5. Побудова зручного графічного інтерфейсу.

Об’єктом дослідження є фреймворк для роботи з доповненою реальністю в iOS - ARKit та фреймворк для роботи з геолокацією та геопозиціювання в iOS - CoreLocation.

Предметом дослідження є мобільний iOS додаток для прокладання навігаційного маршруту в доповненій реальності.

Практичне значення роботи полягає у можливості подальшого використання розробленого навігаційного додатку користувачами мобільних пристроїв на базі iOS.

Курсова робота складається з трьох розділів.

У першому розділі проводиться дослідження проблеми ефективного використання ресурсів мобільного пристрою для забезпечення зручності користування навігаційним додатком, аналіз навігаційних додатків в цілому і можливостей оптимізації та постановка завдання курсової роботи.

Другий розділ присвячений теоретичним відомостям та аналізу фреймворків ARKit і CoreLocation, їх поєднання та можливостей оптимізації роботи додатків на базі даних фреймворків.

Третій розділ описує можливості та процес реалізації додатку для проведення дослідження ефективного використання ресурсів під час навігації у доповненій реальності.

Розділ 1. Аналіз предметної області та постановка завдання курсової роботи

1. Аналіз проблеми.

Оптимізація – процес покращення, нескінченний шлях до ідеалу. Оптимізація роботи мобільного застосунку - це шлях до ефективнішого використання ресурсів пристрою додатком та надання користувачу кращого досвіду взаємодії із застосунком. Тема оптимізації має великий плацдарм для дослідження, як вже наявних, так і нових способів оптимізації використання ресурсів в цілому, та у мобільних додатках в особливості.

Особливо важливо надавати якісний досвід користувачам додатків, призначених для навігації. Адже користувач, під час взаємодії з додатком, рухається в реальному часі по вулиці та очікує швидку та якісну роботу застосунку.

Навігаційний додаток повинен мати зрозумілий графічний інтерфейс, працювати чітко та безперебійно. Помилки чи затримки у роботі даного типу застосунків – неприпустимі.

Навігація вимагає чіткості, а чіткість ресурсів. У даній роботи ми розглянемо способи ефективного використання ресурсів та можливості оптимізації роботи навігаційного застосунку, особливістю якого є прокладання маршруту у доповненій реальності.

Дана функціональність забезпечує краще розуміння користувачем прокладеного маршруту, але передбачає споживання додатком більшої кількості ресурсів пристрою, що підвищує необхідність їх ефективного використання.

2. Аналіз особливостей навігаційних додатків та існуючих аналогів застосунку.

Завдяки поєднанню технології геопозиціювання GPS (Global Positioning System) та мап у мобільних пристроях, існують навігаційні застосунки, які роблять можливою та доступною кожному детальну навігацію на місцевості.

Під час вибору навігаційного додатку в магазині мобільних застосунків, користувач звертає увагу на різні фактори: дизайн, рейтинг, ціна, зручність, можливості та власне призначення додатку.

Рейтинг популярних навігаційних додатків може варіюватися в залежності від платформи смартфону та країни користувача. Але незмінним чемпіоном останні роки був і залишається додаток Google Maps від корпорації Google. Невелику конкуренцію застосунку створюють додатки Apple Maps – вбудовані мапи у мобільні пристрої від Apple, та Waze – додаток-соціальна мережа для водіїв.

Незважаючи на те, що додаток Google Maps залишається кращим навігатором декілька років поспіль, компанія Google не припиняє вражати користувачів новими можливостями застосунку.

У серпні 2019 році компанія Google представила функцію прокладання навігаційного маршруту у доповненій реальності – Google Maps Live Mode. Тобто прокладений на мапі шлях від точки А до Б, при наведенні камери на вулицю, ілюстративно зображується у доповненій реальності за допомогою графічних елементів (додаток А).

Популярності дана функція набула лише у жовтні 2020 року, коли корпорація Google заявила про публікацію її суттєвого оновлення. У новому релізі значно покращили функціональність навігації у доповненій реальності та розширили зону покриття даної функції. Додаток запропонував автоматичне відображення назви закладів, на яке наведена камера, більш чітке прокладення маршруту та позиціювання графічних елементів (додаток Б). [1]

Функція навігації у доповненій реальності Live Mode від Google реалізована за допомогою інтеграції власних мап від Google та технології доповненої реальності, яка реалізується через нативну оболонку кожної платформи окремо. Для Android нативна технологія роботи з доповненою реальністю це ARCore, для iOS – ARKit.

Функція Live Mode, як і будь-яка функціональність, що передбачає роботу з доповненою реальністю, вимагає від мобільного пристрою будь-якої платформи, чи то Android, чи iOS, підтримки можливості роботи з AR.

В цілому робота функції Live Mode у додатку Google Maps – задовільна. Але наявні певні мінуси: функція взагалі не працює з приміщень та в погано освітлених місцях. Під час використання застосунку, камеру пристрою потрібно обов’язково

направляти на будинки або дорожні знаки. Спричинена така поведінка, наявністю ще одного недоліку - дана функція прокладення маршруту у доповненій реальності працює лише в місцях, де наявна функція панорамного перегляду вулиць від Google - Street View.

Підходи функції Live Mode та нашого додатку до реалізації зчитування оточення реального світу під час використання навігації у AR суттєво відрізняються. Google використовує вже наявну інформацію про певні місця, їх знаходження та положення, від технології Street View та, аналізуючи ці дані, прокладає навігаційний маршрут [2]. В той час як наш додаток в реальному часі сканує оточення користувача та одразу графічно прокладає маршрут у доповненій реальності.

Такий підхід до створення навігаційного додатку від компанії Google є цікавим поглядом на вирішення проблеми зручної навігації. Наше дослідження буде зосереджено безпосередньо на аналізі можливостей ‘рідних’ для платформи iOS технологій роботи з навігацією та доповненою реальністю – CoreLocation та ARKit, та на можливостях ефективного поєднання цих двох технологій.

3. Аналіз сучасних технологій для реалізації застосунку.

Найпопулярнішими мобільними операційними системами наразі є iOS та Android.

Існують два варіанти розробки під відповідні платформи. Це *нативна* розробка під певну операційну систему, тобто додаток розробляється окремо під ОС Android та під iOS, та к*россплатформенна* розробка – додаток можна розробити під декілька ОС за один раз - Android та iOS відповідно.

Технології та засоби для кроссплатформенної розробки.

- *Flutter* – комплект засобів розробки та фреймворк для розробки мобільних застосунків мовою Dart від компанії Google.
- *React Native* – фреймворк мови JavaScript від компанії Facebook для розробки кроссплатформенних застосунків.

- *Unity* – кроссплатформенна середа розробки ігор та застосунків, і рушій, на якому вони працюють. Ігрова логіка розробляється за допомогою мови програмування C#. [3]

Технології нативної розробки.

Нативна розробка під платформу iOS може бути здійснена за допомогою мов програмування Swift або Objective-C. Android платформа дозволяє розробляти програмне забезпечення мовами Kotlin та Java.

Переваги та недоліки кроссплатформенної та нативної розробки.

Основними перевагами кроссплатформенної розробки є швидкість розробки додатку та невеликі витрати людських і грошових ресурсів.

Недоліками є:

- обмеженість під час масштабування застосунку;
- складність підтримки додатку;
- часті креші, повільність та довга відповідь додатку;
- не “рідний” вигляд дизайну через відсутність нативних графічних елементів інтерфейсу.

Кроссплатформенна розробка є зручною під час створення нескладного додатку, у якому мало екранів та багато спільних елементів для обох платформ – iOS та Android. Для додатків з унікальними інтерфейсами та складною бізнес-логікою краще підходить нативний спосіб розробки. [4]

Критерії оцінки під час вибори платформи розробки.

Важливо зрозуміти, яка цільова аудиторія додатку. Створити портрет потенційного користувача, оцінити його потреби та фінансові можливості. В нашому випадку, навігаційний додаток потрібен як користувачам пристроїв на базі Android, так і на базі iOS.

Слід оцінити власні ресурси та знання, такі як наявність техніки для розробки і тестування додатку та володіння мовами програмування.

Під час вибору середовища розробки важливо враховувати бажану функціональність майбутнього додатку. Для реалізації власне навігаційного застосунку потрібно використання бібліотек, які дозволяють працювати з мапами,

геолокацією юзера та позиціюванням на місцевості на відповідній платформі. У нашому випадку, додаток також працює з доповненою реальністю. Тож варто дослідити, які бібліотеки для роботи з мапами, геопозиціюванням та AR пропонують різні платформи та засоби розробки.

Кожна, перерахована вище, сучасна мобільна платформа та середовище розробки підтримує роботу з мапами та геопозиціюванням. Android дає можливість працювати з мапами, використовуючи або Google Maps API, або Mapbox Maps SDK. Flutter інтегрується з Google Maps API [5]. Unity дозволяє працювати з мапами та геолокацією завдяки Mapbox Maps SDK [6]. iOS же в свою чергу пропонує фреймворк MapKit для роботи з мапами та CoreLocation для роботи з геолокацією.

Технологія роботи з доповненою реальністю на мобільних пристроях є достатньо молодою, але вже підтримується усіма мобільними платформами та активно розвивається.

На платформі Android працювати з доповненою реальністю дозволяє фреймворк ARCore, який також використовується платформою Flutter. Unity має бібліотеку AR Foundation [7]. У iOS з доповненою реальністю можна працювати завдяки фреймворку ARKit (додаток В).

Після дослідження можливостей різних платформ та середовищ розробки, для реалізації навігаційного додатку з елементами доповненої реальності було обрано проводити розробку під платформу iOS, використовуючи середу розробки XCode від Apple, та провести аналіз саме фреймворків CoreLocation і ARKit.

Причинами такого вибору є зацікавленість у функціональних можливостях та особливостях iOS фреймворків ARKit та CoreLocation, особисті вподобання розробки під операційну систему iOS та наявність техніки від Apple для безпосередньої розробки та тестування додатку.

4. Аналіз можливостей оптимізації роботи мобільних iOS додатків.

Швидкість роботи програми є важливою в усіх сферах розробки, але особливо важливою вона є у мобільній розробці, де обчислювальні потужності та заряд акумулятора обмежені.

Безліч факторів впливає на оцінку досвіду використання додатку користувачем, але одним з найважливіших є швидкодія застосунку.

За статистикою:

- 79% користувачів повторно відкриває додаток, якщо він не працює перший раз;
- 25% користувачів відмовляється від застосунку, якщо він не завантажується протягом перших декількох секунд;
- 31% користувачів розкаже своїм знайомим про невдалий досвід використання додатку. [8]

Отже, першою необхідною мірою для створення гарної репутації застосунку – є виключення можливостей крешів та забезпечення помірного часу запуску додатку.

Способи пришвидшення запуску iOS додатку.

Щоб пришвидшити запуск застосунку в цілому, потрібно оптимізувати роботу додатку в частинах:

- pre-main – до початку виконання функції main, система виконує підготовку образу додатку в пам'яті: завантажуються динамічні бібліотеки, створюється контекст Objective-C. На даному етапі найбільш дієвим способом вплинути на швидкодію є зменшення використання Objective-C коду. Тобто найкращим варіантом буде використовувати мову Swift для розробки додатку.
- after-main – все, що відбувається від початку виклику функції main. На цьому етапі оптимізувати швидкість завантаження додатку можна шляхом використання lazy variables для відкладеної ініціалізації, тобто на старті створювати лише необхідні сутності. [9]

Надмірне використання залежностей CocoaPods у проекті може призвести до збільшення часу запуску додатку. Залежності Pods, що були підключені за допомогою менеджеру залежностей CocoaPods, під час запуску додатку збираються в окремі динамічні бібліотеки. Тому навіть якщо в проекті мало коду, підключені поди можуть значно збільшити час запуску додатку в кілька разів. Тож

підключати бібліотеки слід по мірі необхідності та позбавлятися від залежностей, що не використовуються в проекті.

Після запуску додатку, користувач переходить до роботи із застосунком, що формує його враження про продукт. На даному етапі взаємодія юзера із застосунком повинна відбуватися плавно та безперебійно. Забезпечити стабільну роботу допоможе оптимізація роботи додатку, тобто зменшення використання додатком ресурсів мобільного пристрою.

Мобільний додаток не може існувати без графічного інтерфейсу, тож розглянемо можливості оптимізації роботи з базовим графічним фреймворком в середовищі iOS – UIKit.

Погіршення продуктивності роботи додатку за рахунок повільного промальовування графічного інтерфейсу на головному потоці відбувається при наявності:

- *Непрозорості або змішаних шарів*

Щоб створити прозорий шар, системі необхідно провести додаткові обчислення - змішати верхній та нижній шари, щоб визначити колір та намалювати його.

- *Закадрової візуалізації (offscreen rendering)*

Це процес промальовування зображення, який не може бути виконаний за допомогою апаратного прискорення GPU, замість нього використовується процесор CPU. [10]

Тобто під час промальовування шару, який потребує закадрової візуалізації, GPU зупиняє процес візуалізації та передає управління процесору CPU, що виконує всі необхідні операції та повертає управління GPU з уже промальованим шаром. GPU візуалізує даний шар, і тільки тоді процес промальовування елементів графічного інтерфейсу продовжується. [11]

Розглянемо, які ефекти та налаштування елементів UIKit призводять до закадрової візуалізації.

- Власне перевизначення методу drawRect().

Замість перевизначення своєї власної реалізації методу drawRect для простих операцій, як налаштування кольору фону, слід використовувати властивості UIView - backgroundColor;

- Радіус заокруглення для CALayer.

Слід використовувати клас UIBezierPath для заокруглення країв елементу, замість властивості cornerRadius;

- Тінь для CALayer.

Замість використання shadowRadius та shadowOpacity, варто визначити форму тіні, використовуючи властивість shadowPath для CALayer;

• Малюнок, створений з використанням CGContext.

Як кращу альтернативу слід використовувати клас UIBezierPath. [12]

5. Постановка завдання курсової роботи

Дослідивши існуючі навігаційні додатки, та проаналізувавши потреби користувачів, було виявлено, які проблеми найбільше непокоять користувачів навігаційних додатків, та які рішення слід запропонувати:

- Креші, повільна робота додатку, помилки під час роботи з додатком: за рахунок використання досліджених способів розумного використання ресурсів пристрою, додаток повинен працювати безперебійно;
- Зручність інтерфейсу та використання: для застосунку має буде продуманий та створений зручний UI;
- Конфіденційність: додаток не використовує особисту інформацію юзера, а використання у застосунку мап Apple Maps гарантує конфіденційність даних користувачів;
- Підтримка невеликої бази країн: завдяки інтеграції iOS фреймворку MapKit з мапами Apple Maps, додаток має можливість працювати у більш ніж 200 країнах світу; [13]
- Висока ціна або наявність реклами: додаток безкоштовний та не містить будь-яку рекламу.

Завдання курсової роботи:

Дослідити особливості фреймворків для роботи з навігацією CoreLocation та доповненою реальністю ARKit на платформі iOS, та можливості ефективного поєднання даних фреймворків.

Створити додаток, що розв’язує перераховані вище проблеми та забезпечує зручну навігацію для користувачів смартфонів на базі iOS, шляхом поєднання відповідних фреймворків та запровадженням способів розумного використання ресурсів пристрою додатком.

Розділ 2. Теоретичні відомості

1. Фреймворк доповненої реальності ARKit.

AR – Augmented Reality – технологія доповненої реальності - поєднує реальний та віртуальний світи. Технологія доповненої реальності активно використовується у різних сферах людської діяльності та має великі перспективи до розвитку та масштабного поширення в майбутньому.

Сфера, яку не змогла оминути технологія доповненої реальності – це технологічна сфера. Наявність потужної камери для зчитування реального світу на сучасних мобільних пристроях, дозволяє технології AR активно використовуватися у мобільних додатках та іграх, доповнюючи нашу реальність графічними анімаційними елементами у смартфоні.

Велика частина людей дізналися про технологію доповненої реальності з гри, що набула популярності у 2016 році, а саме – Pokémon Go від компанії Niantic. Pokémon Go є першою в світі грою із застосування технології доповненої реальності у мобільному додатку, яка набула популярності таких масштабів. Гра була завантажена більше ніж 500 мільйонів разів за перший рік з моменту її запуску. [14]

Гра розроблена за допомогою інтеграції Google Maps з двигуном Unity 3D, що дозволяє використовувати карту від Google як текстуру в грі. Відслідковування місця знаходження гравця відбувається за допомогою технології GPS. [15]

Гра Pokémon Go зацікавила не лише ігроманів, а й розробників, які побачили у грі можливості застосування технології AR у повсякденні та потенціал до її розвитку.

Розробку додатків з технологією доповненої реальності під мобільні пристрої на базі iOS робить можливою фреймворк від Apple під назвою ARKit - Augmented Reality Kit. Бібліотека була представлена Apple у 2017 році під час конференції WWDC, та на той момент забезпечувала підтримку роботи із пристроями, починаючи від версії операційної системи iOS 11. ARKit працює на пристроях з процесорами A9 і вище, тож мінімальна підтримувана версія мобільного пристрою – це лінійка iPhone 7 та SE.

Фреймворк ARKit – це інструмент, який за допомогою камери пристрою та вбудованих датчиків руху розпізнає особливості відеокадрів, відстежує зміни їх положення та порівнює цю інформацію з даними від датчиків руху. Фреймворк знаходить поверхні, їх розташування і розміри, визначає освітленість елементів (додаток Г).

Головним завданням ARKit є спостереження за навколишнім світом (World Tracking) та збирання даних, після аналізу яких буде отримано конкретне уявлення про навколишній світ - віртуальну модель реального світу у вигляді координат поверхонь, точок перетину та положення камери у просторі.

Основою фреймворку ARKit є елементи для відображення зображень та анімацій у доповненій реальності - це ARSCNView з фреймворку SceneKit для роботи з 3D графікою та ARSKView з бібліотеки SpriteKit для роботи з графічними елементами у 2D вимірі.

Елементи ARSCNView та ARSKView містять в собі рушій фреймворку ARKit – ARSession – це клас, що інкапсулює в собі все необхідне для роботи з доповненою реальністю. Необхідно передати конфігурацію роботи сесії – ARWorldTrackingSessionConfiguration для запуску ARSession (додаток Д). Буде створена модель навколишнього світу у доповненій реальності та надана інформація щодо площин у полі зору камери мобільного пристрою. [16]

Для створення частини взаємодії користувача з доповненою реальністю, під час розробки нашого навігаційного додатку, використана бібліотека ARKit. Розглянемо *деталі її підключення*:

- Для початку слід імпортувати бібліотеку ARKit у проект: import ARKit;

- На екрані створити view типу ARSKView, у якому будуть відображатися елементи доповненої реальності;

- Створити сесію доповненої реальності та передати у неї конфігурацію;

```
override func viewWillAppear(_ animated: Bool) {
  super.viewWillAppear(animated)
  let configuration = ARWorldTrackingConfiguration()
  sceneView.session.run(configuration)
}
```

Рисунок 1. Приклад роботи з ARKit

- Для використання камери пристрою у додатку слід додати поле Camera Usage Description у файл під назвою Info.plist.

Сучасні можливості ARKit 4 та впроваджені раніше покращення:

Перша версія технології ARKit здобула спірну популярність на ринку, адже з одного боку була чимось кардинально новим, але з іншого - представлені функціональні можливості були ще "сирими".

Друга версія значно покращила роботу фреймворку та запропонувала технологію розпізнавання поверхонь – Plane Detection, надання фізичних характеристик віртуальним об’єктам, таких як наприклад вага, та підлаштовування віртуальних об’єктів під рух користувача. [17]

ARKit версії 3 запропонував нову цікаву функцію – People Occlusion, яка дозволяє визначати знаходження людини в рамках камери пристрою, зчитувати рухи та положення її тіла. [18]

ARKit 4, представлений Apple під час конференції WWDC`20, пропонує нові можливості для розробників, які доступні на пристроях з процесором A12.

Нове покоління мобільних пристроїв на базі iOS 14 з технологією Apple LiDAR Scanner, вдосконалює можливості роботи з людиною в кадрі пристрою. Можливим стає розміщення графічних об’єктів в доповненій реальності поза або перед людиною. [19]

Оскільки платформа ARKit досить вимоглива до ресурсів, додатки з її підтримкою доступні тільки на пристроях, побудованих на базі процесору Apple A9 та новіших. Тільки їх продуктивності достатньо для того, щоб відсканувати навколишній світ за допомогою камери та органічно вбудувати туди новий об'єкт віртуального світу. [20] Через ресурсозатратність технології доповненої

реальності, важливо розумно використовувати ресурси мобільного пристрою, під час роботи з нею.

2. Фреймворк геопозиціювання CoreLocation.

Невід'ємна складова навігації - відслідковування локації користувача - це функція, яку в наш час використовують більшість мобільних додатків.

У мобільних додатках на смартфонах з операційною системою iOS навігаційні функції здійснює фреймворк CoreLocation від Apple. Він надає послуги розташування, які дозволяють отримати поточне місцезнаходження пристрою за допомогою GPS та напрямок руху за допомогою компасу мобільного пристрою.

У нашому навігаційному додатку фреймворк CoreLocation використовується для безпосередньої роботи с геолокацією користувача – визначення його поточної локації і напрямку руху.

Деталі підключення та роботи з CoreLocation:

- Для початку роботи із бібліотекою геопозиціювання від Apple слід імпортувати її у проект за допомогою команди: import CoreLocation;
- Додати поля «Location Always and When In Use Usage Description» та «Location When In Use Usage Description» у файл Info.plist для отримання дозволу на відслідковування геолокації користувача;
- Взаємодія із фреймворком геопозиціювання відбувається через посередника - Location Manager. [21] Тож слід визначати його для подальшої роботи та створити запит для отримання дозволу працювати з геолокацією користувача:

```
var locationManager: CLLocationManager?

locationManager.requestWhenInUseAuthorization()
```

Після налаштувань слід визначатися щодо способу відслідковування локації юзера. Існує 4 варіанти:

- *Standard location services* (стандартні сервіси геопозиціювання) - періодичне відстеження місцезнаходження користувача за допомогою GPS. Підходить для створення додатків для навігації або занять спортом за рахунок чіткого визначення локації юзера, але й через що технологія використовує багато ресурсів мобільного пристрою;
- *Significant Location Changes Monitoring* (Geo Fence або Region Monitoring) – технологія визначення значних змін геолокації. Ініціює події, коли користувач потрапляє в діапазон певного географічного місця. Пропонує більш енергозберігаючу альтернативу в порівняні з іншими технологіями за рахунок покладання на альтернативи нижчої потужності, такі як Wi-Fi та мобільний зв'язок. Оновлення місцезнаходження відбувається лише тоді, коли позиція користувача змінюється на значну величину у метрах. Застосовується у додатках, які не потребують частого оновлення геолокації або точності GPS. [22]
- *Visit Monitoring* – моніторинг відвідувань. Технологія визначає інформацію про відвідування користувачем певного місця, наприклад, як довго користувач перебував у кафе, музеї тощо.
- *Beacon Ranging* - виявляє наближення користувача до пристроїв iBeacon за допомогою Bluetooth сервісів. [23]

3. Можливості поєднання фреймворків ARKit та CoreLocation.

Фреймворк ARKit має великий потенціал до поєднань з іншими технологіями від Apple.

Поєднання ARKit з фреймворками для створення 2D та 3D графіки – SpriteKit та SceneKit відповідно, дозволяє доповнювати реальний світ віртуальними графічними елементами.

Поєднання технології доповненої реальності з геолокацією CoreLocation та мапами MapKit дозволяє створювати навігаційні додатки із новим поглядом на звичне переміщення містом – переміщення у доповненій реальності.

Використання навігаційних даних у доповненій реальності на базі фреймворків ARKit та CoreLocation:

Для розміщення графічного об’єкту у середовищі доповненої реальності на певному місці, слід задати географічні координати цього місця. Для точного позиціювання елементу слід провести конвертацію географічних координат у координати сцени ARKit. Спричинена дана неточність наступним фактором:

Під час старту додатку, ARKit створює систему координат із початком в точці, яка відповідає поточному положенню мобільного пристрою в просторі, та направляє осі системи в залежності від значення властивості woldAlignment.

Щоб осі були спрямовані наступним чином: вісь Y - в протилежному напрямку гравітації, вісь Z - на південь, а вісь X - на схід, параметру woldAlignment слід надати значення .gravityAndHeading.

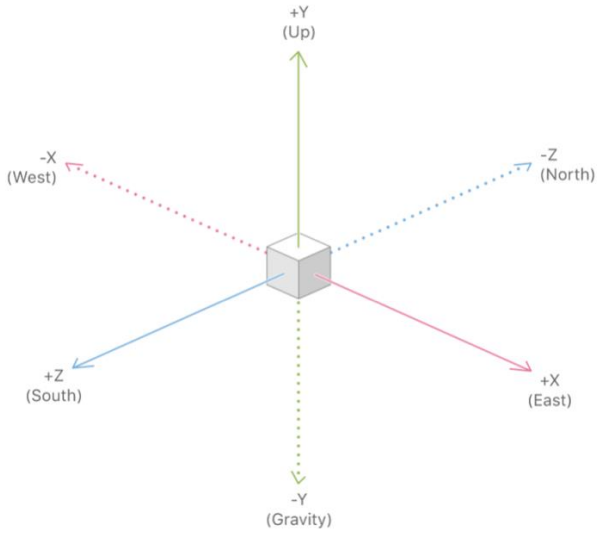


Рисунок 2. Система координат технології ARKit

Але осі X / Z технології ARKit не завжди збігаються з реальними осями Схід / Південь. Причиною цьому є те, що ARKit ініціалізує систему координат один раз, після чого не коригує напрямки осей, так як в більшості застосувань доповненої реальності напрямок осей не є важливим.

У випадку неправильної ініціалізації технологією ARKit осей, віртуальні елементи у доповненій реальності під час розміщення будуть зміщені на певний кут, що обумовлено наявністю куту повороту системи координат ARKit відносно реальних напрямків Схід / Південь.

Тож для точного позиціювання віртуального елементу на сцені доповненої реальності на певному місці, слід провести корекцію значення даного кута повороту та конвертацію географічних координат в координати сцени ARKit за допомогою матриці трансформації.

Після знаходження правильного кута між точками, об'єкт буде розвернуто у правильному напрямку для забезпечення чіткості розміщення об'єктів на сцені.

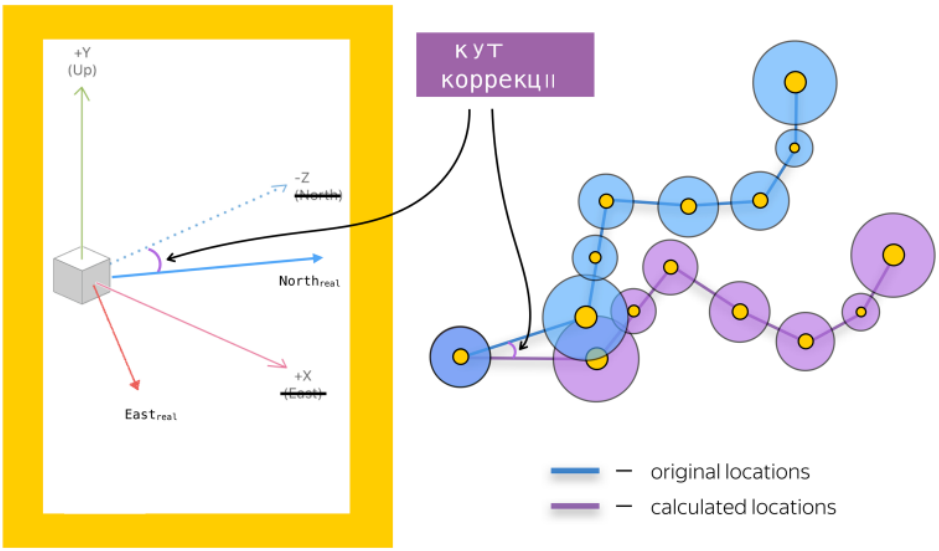


Рисунок 3. Кут корекції системи координат ARKit [24]

Після визначення координат точки на сцені ARKit, створюється графічний елемент. Йому задаються розмір, текстура та локація. Елемент додається на сцену доповненої реальності на місце конвертованих координат.

4. Можливості оптимізації роботи застосунку на базі фреймворків ARKit та CoreLocation.

Після проведення оптимізації додатку загалом, слід звернути увагу на оптимізацію роботи з використовуваними у застосунку фреймворками.

У нашому навігаційному додатку фреймворками, що надають головну функціональність, є ARKit та CoreLocation. Тож розглянемо можливості оптимізації роботи з даними бібліотеками.

Можливості оптимізації роботи додатків з використанням бібліотеки ARKit на базі iOS

На головному потоці програми виконується код, що відповідає за взаємодію користувача з графічним інтерфейсом додатку та промальовування усієї графічної складової екрану. У більшості сучасних смартфонів рендеринг відбувається з частотою 60 кадрів в секунду. Якщо якась операція займає більше 16 мілісекунд (1 секунда / 60 кадрів), автоматично відбувається втрата кадрів, що є помітним для користувачів при відтворенні картинки. Тому прискорення роботи головного потоку є важливим.

Навігація у AR передбачає роботу з графічними елементами у доповненій реальності, тож потрібно розумно працювати з графікою під час розробки додатків даного типу.

Для *зменшення використання ресурсів під час роботи з 2D та 3D графікою* слід:

- Використовувати якнайменше графічних анімованих об’єктів;
- Не використовувати без потреби дуже деталізовані текстури. Краще працювати з найменшою текстурою, яка все ще дає візуально привабливі результати;
- Без потреби не використовувати занадто складні моделі, які складаються з декількох полігонів;
- Не накладати атрибути тіней та непрозорості на об’єкти. Поява одного або декількох шарів прозорого матеріалу поверх інших матеріалів призводить до того, що відмальовувач фрагментів запускається на пікселях по разу для кожного відтвореного шару. Так само, чим більше світла та тіней на сцені, тим більше роботи виконує відмальовувач фрагментів;
- Не змінювати значення атрибуту scale об’єкту;
- Використовувати атласи (папки) для комбінування декількох текстур в одну. Важливо щоб графічні моделі мали один спільний матеріал. Наявність меншої кількості окремих текстурних файлів допомагає зменшити споживання пам'яті; [25]

- Також важливо створювати невеликі атласи, адже так взаємодія з ними та завантаження елементів відбувається швидше;
- Використовувати правильні формати для текстур та звукових файлів;
- Обмежити максимальну кількість відтворюваних кадрів в секунду (FPS) на старих пристроях. [26]

Можливості оптимізації безпосередньо AR функціональності.

Під час пауз роботи із функціональністю доповненої реальності, наприклад під час відкриття модального вікна або меню налаштувань, слід зупиняти сесію AR за допомогою виклику методу session.pause(). Це допоможе зменшити використання ресурсів мобільного пристрою під час відсутності потреби у AR.

Джерела звуку, прив'язані до взаємодії з певними графічними об'єктами, допомагають зробити сцену більш захоплюючою, але обробка звуку може забирати занадто багато процесорного часу. Тож потрібно обмежувати кількість джерел звуку на сцені в певний момент часу. [27]

***Можливості оптимізації роботи додатків з використанням технології навігації на базі iOS* за допомогою фреймворку CoreLocation**

Досягнути зменшення використання ресурсів додатком під час роботи з геолокацією частіше за все означає знехтувати чіткістю обчислень, адже чіткість вимагає ресурсів.

Використання ресурсів пристрою залежить від технології визначення геолокації, що використовується. Технології відслідковування значних змін за рахунок нечіткості даних споживають менше ресурсів пристрою. В той час як стандартне визначення геолокації пропонує чіткі дані та, за рахунок частих оновлень, споживає більше ресурсів.

Можливості оптимізації під час роботи із технологією стандартного визначення геолокації юзера за допомогою GPS:

Для зазначення чіткості визначення геолокації користувача, слід надати значення атрибуту locationManager – *desiredAccuracy*. *[28]*

Щоб зменшити вплив програми на використання ресурсів пристрою, таких як рівень заряду, слід призначити атрибуту значення, яке відповідає цілі

використання додатку. Наприклад, для переміщення країнами чіткість визначення не так важлива, як для навігації по місту.

Можливі значення атрибуту характеризують чіткість наданої геолокації користувача менеджером:

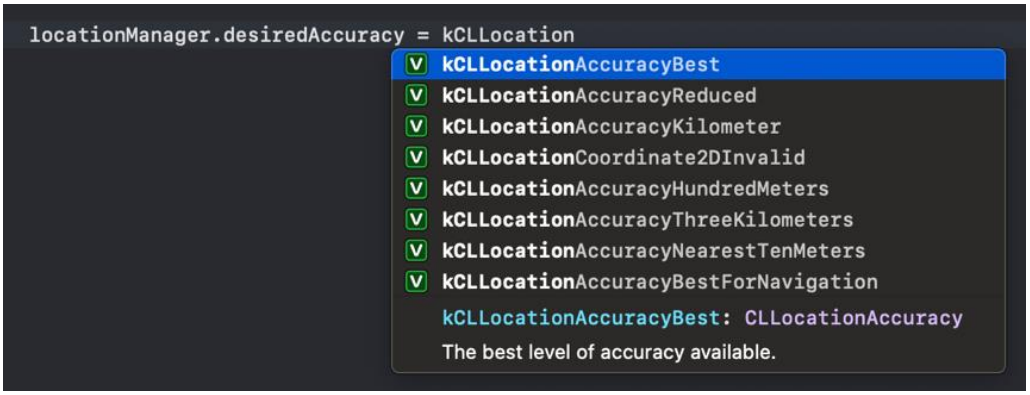


Рисунок 4. Можливі значення атрибуту desiredAccuracy.

Іншим атрибутом для зазначення чіткості визначення геолокації користувача є *distanceFilter*. Оновлення геолокації відбувається не раніше ніж користувач горизонтально переміститься не менше ніж на певну відстань у метрах. Атрибут приймає значення типу Int.

```
locationManager.distanceFilter = 3
```

Час автономної роботи пристрою може покращити надання дозволу диспетчеру місцезнаходжень (locationManager) автоматично призупиняти оновлення визначення геолокації у випадках, коли дані про місцезнаходження навряд чи зміняться; [29]

```
locationManager.pausesLocationUpdatesAutomatically = true
```

Автоматичне відключення оновлення локації користувача, коли додаток не використовується, теж збереже рівень заряду мобільного пристрою.

```
locationManager.allowsBackgroundLocationUpdates = false
```

Розділ 3. Опис практичного дослідження

1. Постановка технічного завдання.

Створити навігаційний додаток для мобільних пристроїв на базі операційної системи iOS. Головна функція додатку – прокладання маршруту для навігації користувача у доповненій реальності за допомогою графічних елементів.

Додаток повинен вирішувати основну проблему, що була досліджена у курсовій роботі проблему, а саме надавати зручність використання додатку за рахунок гладкої та безперебійної роботи додатку, яка досягнута шляхом ефективного використанням ресурсів додатком. Також додаток має на меті вирішити другорядні досліджені проблеми сучасних навігаційних додатків, пропонуючи конфіденційність інформації користувачів, безкоштовність додатку та відсутність реклами.

Додаток має бути розроблений за допомогою поєднання бібліотек для роботи з геопозиціюванням та доповненою реальністю від Apple – ARKit та CoreLocation. Мова програмування додатку – Swift.

Потрібно оптимізувати роботу додатку шляхом впровадження в додаток досліджених у даній курсовій роботі методів оптимізації використання ресурсів під час роботи з фреймворками ARKit та CoreLocation на мобільній платформі iOS.

2. Опис архітектури додатку.

Важливим аспектом у процесі створення додатку є планування його архітектури. Слід оцінити переваги та недоліки існуючих архітектурних шаблонів щодо поточного проекту та підібрати такий, що буде задовольняти потребам додатку.

Наш додаток складається з немалої кількості екранів, які пропонують користувачу різні функціональні можливості. Для організації зручної роботи з екранами додатку та переходами між ними, влучним вибором стане архітектурний патерн Координатор (Coordinator).

Патерн дозволяє керувати та організовувати потік додатку, а саме керувати екранами – показувати, приховувати, передавати контроль.

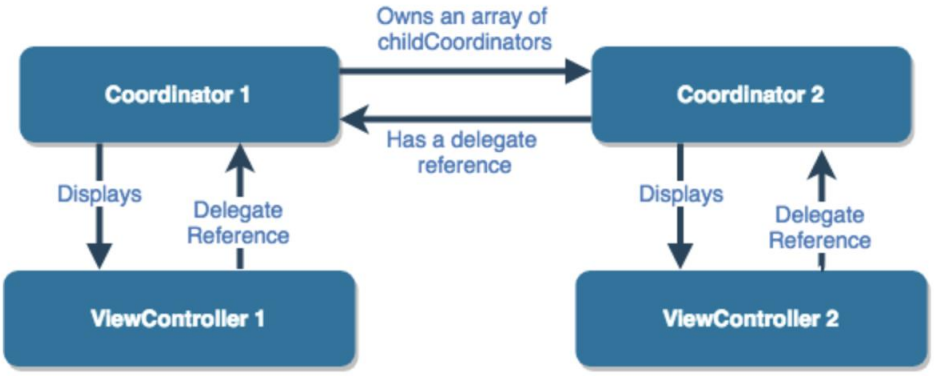


Рисунок 5. Схема патерну Координатор.

Використання патерну у додатку.

Після запуску програми створюється екземпляр класу головного координатору – MainCoordinator, що піж час своєї ініціалізації викликає метод визначення, який потік програми запустити, тобто якому координатору із своїх дочірніх передати управління. Початково користувачу презентується головний екран додатку, для цього у коді відбувається звернення до координатору, що його запускає – MainControllerCoordinator (додаток Е).

У додатку наявні 4 екрани:

- MainVC – головний екран з інструкцією користувача,
- MapSearchVC – екран мапи для обрання користувачем локації призначення,
- NavVC – екран навігації у доповненій реальності,
- BackMapVC – допоміжний екран, що відображає невелику мапу під час перебування користувача на екрані навігації.

Для кожного екрану створено клас координатор, що працює з відповідним класом екрану (додаток Ж).

3. Опис розробки додатку.

Додаток із впровадженим у нього поєднанням фреймворків ARKit та CoreLocation працює наступним чином:

Під час старту роботи додатку, фреймворк CoreLocation, завдяки технології GPS, визначає поточну локацію юзера. Користувач обирає точку на відображеній мапі, куди він прямує. Точка конвертується у географічні координати. Фреймворк

MapKit, що взаємодіє з мапами Apple Maps, прокладає маршрут від поточної локації до обраного користувачем місця. Прокладений шлях є набором точок координат, що конвертується в набір легів – частин маршруту. Леги використовуються для поетапного відображення маршруту на сцені доповненої реальності задля економії ресурсів мобільного пристрою.

За допомогою фреймворку ARKit створюється сесія доповненої реальності ARSession, яка буде зчитувати оточення користувача через камеру пристрою. Для визначення плоскої поверхні, над якою в майбутньому необхідно буде розмістити графічні елементи маршруту, використовується функція автоматичного розпізнання площин – plane detection. Після розпізнавання, над площинами будуть розміщені графічні елементи – проміжні точки навігації, створені завдяки класу SCNNode з бібліотеки тривимірної графіки SceneKit.

Точки будуть розміщені по легам у місцях з набору координат маршруту. Під час навігації буде прокладатися лише поточна частина всього шляху.

На виході маємо навігаційний додаток, що прокладає маршрут від пункту А до Б у доповненій реальності.

Опис програмних деталей поєднання фреймворку ARKit та CoreLocation:

Отримання початкової та кінцевої точки маршруту:

Під час запуску додатку за допомогою методу CLLocationManager trackingLocation() отримуємо координати поточної локації користувача.

```
func trackingLocation(for currentLocation: CLLocation) {
    if currentLocation.horizontalAccuracy <= 100.0 {
        print("location updated")
        updatedLocs.append(currentLocation)
        updateArRoute()
        currLocation = currentLocation
    }
}
```

Рисунок 6. Код отримання поточної локації користувача

Користувач рухається, тож локація час від часу оновлюється. Всі поточні оновлені локації користувача записуються у масив updatedLocs.

Для визначення початкової точки маршруту, з масиву обирається найточніша поточка локація за допомогою методу mostAccurateLocFrom(), що сортує локації та обирає горизонтально найточнішу.

```
startLocation = CLLocation().mostAccurateLocFrom(locations: updatedLocs)
```

```
func mostAccurateLocFrom(locations: [CLLocation]) -> CLLocation {

    let sort = locations.sorted(by: {

        if $0.horizontalAccuracy == $1.horizontalAccuracy { return $0.timestamp > $1.timestamp }

        let morePrecise = $0.horizontalAccuracy < $1.horizontalAccuracy

        return morePrecise
    })
    return sort.first!
}
```

Рисунок 7. Код визначення найчіткішої локації

Таким чином під час кожного оновлення поточної локації, визначається початкова точка та маршрут перемальовується відносно неї.

Після запуску додатку на мапі користувач власноруч задає точку призначення. Обрана точка конвертується у географічні координати.

```
if gesture.state != UIGestureRecognizer.State.began
{ return }
let tapPoint = gesture.location(in: mapView)
let coordinate: CLLocationCoordinate2D = mapView.convert(tapPoint, toCoordinateFrom: mapView)
destLocation = coordinate
```

Рисунок 8. Код конвертації координат на мапі

Прокладення маршруту між точками:

Завдяки фреймворку MapKit для роботи з мапами від Apple створено метод, що від поточної локації до точки призначення визначає кроки (MKRoute.Step) шляху (додаток К). Результат методу записується у масив steps.

```
self.routeSteps.append(contentsOf: steps)
```

Відмальовування точок на мапі:

На мапу точки маршруту наносяться завдяки анотаціям – об’єктам класу Annotation, що мають підпис та локацію: кожен крок маршруту конвертуються в тип Annotation та потім додається на мапу (додаток Л).

Конвертація отриманих кроків маршруту у леги:

Для відображення на сцені доповненої реальності, кроки маршруту конвертуються у леги – частини маршруту. Між кожним знайденим кроком маршруту знаходяться підкроки на певній дистанції один від одного. Для кожного кроку маршруту створюється об’єкт типу класу RouteLeg, що містить у собі проміжні підкроки легу та навігаційну інструкцію (додаток М).

Отримані леги записуються у масив routeData, що передається класу відображення сцени доповненої реальності NavVC для подальшої їх обробки.

```
routeData = legs
```

Конвертація географічних координат у координати сцени:

Для розміщення отриманого набору підкроків легів (географічних координат) з масиву routeData на сцені доповненої реальності, їх слід конвертувати у координати сцени.

Після запуску класу NavVC, обрання кінцевої точки та визначення поточної, викликається метод addRouteSpheres().

```
self.addRouteSpheres(steps: self.routeData)
```

Метод отримує перший лег з масиву routeData, та викликає методи, що створюють графічні елементи сфери для промальовки на сцені доповненої реальності. Перші координати у лезі мають інструкції та позначаються іншим кольором.

```
let localCurrLeg = routeData[currLeg]

for (index, location) in localCurrLeg.coordinates.enumerated() {

    if index == 0 { addLabelSphere(routeStep: localCurrLeg) }
    else {
        let loc = CLLocation(latitude: location.latitude, longitude: location.longitude)
        addSphere(location: loc)
    }
}
```

Рисунок 9. Код виклику методів для створення сфер

Для кожної точки поточного легу створюється об’єкт сфери та проводиться трансформація географічних координат у координати сцени. Об’єкти додаються на сцену доповненої реальності на позиції трансформованих координат.

```
private func addSphere(location: CLLocation) {

    let locTrans = TransMatrix.transformCoordinates(startLoc: startLocation,
                                                    location: location)

    let sphereAnchor = ARAnchor(transform: locTrans)
    spheresAnchors.append(sphereAnchor)

    let sphereNode = SphereNode(title: "Title", location: location)
    sphereNode.addSphere(radius: 0.15, color: )
    sphereNode.anchor = sphereAnchor
    sphereNode.location = location

    sceneView.session.add(anchor: sphereAnchor)
    sceneView.scene.rootNode.addChildNode(sphereNode)

    sphereNodes.append(sphereNode)
}
```

Рисунок 10. Код створення графічного елементу сфера

Конвертація географічних координат у координати сцени відбувається за допомогою методу transformCoordinates().

Метод приймає на вхід географічні координати точки маршруту і початкову точку шляху, та завдяки матриці перетворень трансформує 3D координати (додаток Н).

Опис розробки дизайну додатку:

Дизайн додатку було розроблено за допомогою технології UIStoryboard, що доступна у середовищі розробки XCode.

Головний екран було розроблено за допомогою графічних елементів середовища розробки (додаток П). Екрани UIStoryboard були прив’язані до відповідних класів у коді. Подальша робота відбувалась у коді програми.

4. Опис способів оптимізації додатку.

Ключовою оптимізацією використання ресурсів у додатку є прокладання маршруту по легам – частинам маршруту, а не увесь одразу. Це значною мірою зменшує споживання ресурсів мобільного пристрою під час розрахунків всередині додатку та під час рендерингу графічних елементів. Суттєво менших ресурсів пристрою вимагає рендеринг невеликої кількості графічних елементів орієнтирів маршруту у доповненій реальності, ніж відмальовка всіх орієнтирів маршруту одразу.

Маршрут розбивається на частини по поворотах вулиці. У поточній частині маршруту визначається останній крок, і під час переміщення юзера перевіряється,

чи юзер знаходиться біля цього кроку. Якщо так – поточна частина маршруту прибирається зі сцени, та прокладається наступна, і так відбувається поки користувач не дійде до останньої точки шляху.

Застосування технологій Regional Monitoring, Visit Monitoring та Significant location changes не задовільнить потреби нашого додатку, адже поворотів на шляху може бути багато, тож регіони останніх кроків у частинах можуть накладатися.

Найбільш доцільним рішенням для навігаційного додатку є використання Standard location services – тобто технології звичайної роботи з локацією юзера. Отримуємо поточну локацію юзера, з певною точністю порівнюємо з координатами останнього кроку у поточній частині.

Задля збереження ресурсів пристрою – використовується distanceFilter – мінімальна відстань, на яку юзер повинен переміститися для наступного оновлення його геолокації.

```
locationManager.distanceFilter = 3
```

Отже, маємо - після переміщення юзера мінімум на три метри, отримуємо його геолокацію та порівнюємо з геолокацією останнього кроку поточного легу. Такий вид роботи з геолокацією юзера є дуже простим, а за рахунок використання distanceFilter не такий ресурсозатратний.

При необхідності відбувається оновлення поточної локації користувача та промальовується більш чіткий маршрут - відбувається оновлення точок на екрані. Важливо прибирати старі точки зі сцени sceneView, і тільки після цього малювати нові, щоб не виникало накладання шарів. Для цього у додатку, перед промальовуванням оновленої частини маршруту, викликається метод, що прибирає зі сцени кожну точку з масиву наявних на екрані точок nodes. Лише після цього промальовується оновлена частина маршруту.

Відслідковувати кількість деталей на поточній AR сцені дуже зручно за допомогою ARKit Performance Statistics, що показує поточну відтворювану кількість кадрів у секунду та кількість графічних елементів і полігонів на сцені (додаток Р). [30]

Оптимізація роботи з фреймворком ARKit:

Після дослідження можливостей зменшення навантаження на процесор під час роботи з графікою та власне AR функціональністю, у проекті було проведено наступні оптимізації:

Навігація у доповненій реальності відбувається завдяки простих моделей – шарів, які не взагалі не мають текстур, а лише складаються з матеріалу кольору. У шарів відсутні тіні та непрозорість. Атрибут scale не використовується (додаток С).

```
func addSphere(radius: CGFloat, color: UIColor) {

    let bubble = SCNSphere(radius: radius)
    bubble.firstMaterial?.diffuse.contents = color

    let sphereNode = SCNNode(geometry: bubble)

    addChildNode(sphereNode)
}
```

Рисунок 11. Код створення графічного шару

Зупинення сесій AR під час відсутності взаємодії з ними.

```
func sessionWasInterrupted(_ session: ARSession) {
    print("Session interrupted")
    sceneView.session.pause()
}

func sessionInterruptionEnded(_ session: ARSession) {
    print("Session resumed")
    sceneView.session.run(session.configuration!, options: [.resetTracking,
                                                            .removeExistingAnchors])
}
```

Рисунок 12. Код зупинення AR сесій

Зменшення якості відео задля кращої продуктивності роботи додатку.

```
var configuration = ARWorldTrackingConfiguration()

configuration.videoFormat = ARWorldTrackingConfiguration.supportedVideoFormats[2]
// 0 - imageResolution=(1920, 1440) framesPerSecond=(60)
// 1 - imageResolution=(1920, 1080) framesPerSecond=(60)
// 2 - imageResolution=(1280, 720) framesPerSecond=(60)
print(configuration.videoFormat)
```

Рисунок 13. Код зменшення якості відео

Зменшення максимального FPS – відтворюваних кадрів за секунду.

```
var sceneView: ARNavigationView!

sceneView.preferredFramesPerSecond = 30
```

Оптимізації під час розробки дизайну застосунку:

Розробка графічного інтерфейсу користувача відбувається із запровадженням методів для зменшення використання ресурсів додатком, що було розглянуті у роботі.

Під час розробки головного меню, для надання елементам привабливого виду, були застосовані досліджені альтернативи:

- Замість надання значення атрибуту alpha, для фону додатку одразу обрано зображення із ефектом нечіткості. Відсутність непрозорості елементів, як було досліджено у роботі, пришвидшує рендеринг графіки додатку (додаток Т).
- Заокруглення кнопки «Start» на головному екрані було виконано за допомогою класу UIBezierPath, замість використання атрибуту cornerRadius.

```
let bezierPath = UIBezierPath(roundedRect: startButton.bounds, cornerRadius: 15)

let maskLayer = CAShapeLayer()
maskLayer.path = bezierPath.cgPath
startButton.layer.mask = maskLayer
```

Рисунок 14. Код заокруглення кнопки

Інші впроваджені оптимізації, що були досліджені протягом роботи:

- Відсутність подів у проекті, що пришвидшує запуск додатку – немає потреби, увесь функціонал можна створити без їх допомоги;
- Використання у коді lazy variables для ініціалізації за потребою деяких об’єктів та пришвидшення запуску додатку;
- У додатку повністю відсутній звуковий супровід, задля кращої роботи основного функціоналу застосунку – прокладення навігаційного маршруту.

5. Інструкція та принцип роботи додатку.

Для досягнення мети створення дійсно зручного застосунку, було розроблено зручний графічний інтерфейс користувача, що дозволяє легко працювати з додатком.

Для початку роботи слід натиснути на іконку додатку.

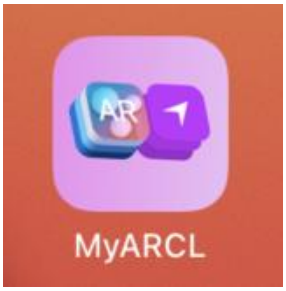


Рисунок 15. Іконка додатку

Спочатку юзер побачить екран завантаження, а потім головне меню з невеликою інструкцією щодо використання (додаток У).

У перший раз використання застосунку, юзер повинен дозволити працювати з його геолокацією для можливості роботи додатку.

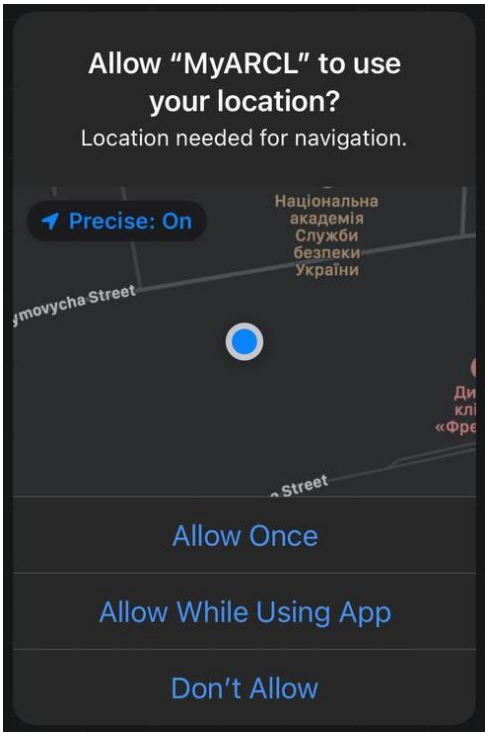


Рисунок 16. Надання дозволу на відслідковування геолокації

Після натискання на кнопку старт, починається безпосередня робота з функціональністю додатку. Довгим натиском на мапі слід обрати точку, для прокладення маршруту до неї (додаток Ф).

Відкриється модальне вікно підтвердження намірів. Для скасування операції та обрання іншого маршруту слід обрати опцію «Choose another».

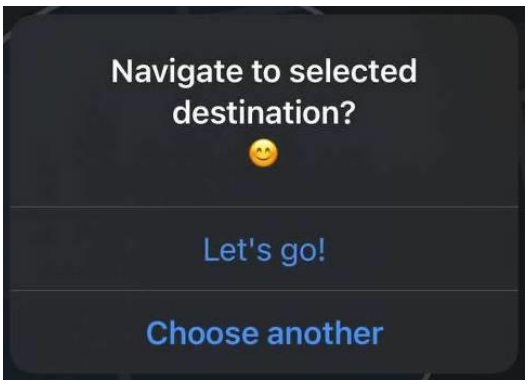


Рисунок 17. Модальне вікно підтвердження маршруту

Після натиску на опцію «Let`s go!», користувача буде переведено до екрану навігації у доповненій реальності (додаток Х).

Під час першого використання додатку, юзер отримає запит на використання камери пристрою.

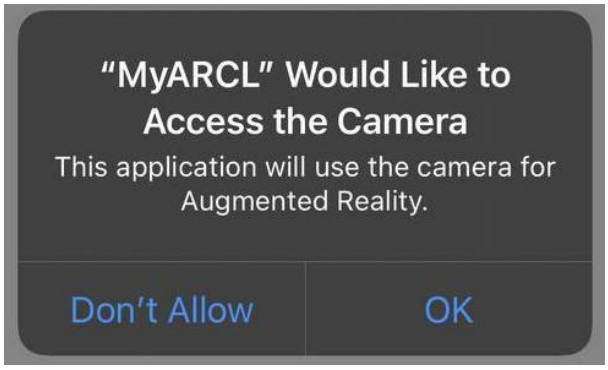


Рисунок 18. Надання дозволу на використання камери

Для прокладання маршруту у доповненій реальності потрібно натиснути на екран (додаток Ц).

У лівому верхньому кутку знаходиться невелика мапа, на якій відображається поточне місцезнаходження користувача для зручності навігації.

Рисунок 19. Мапа-компас на екрані навігації

Висновки та аналіз можливостей подальшого розвитку

У даній курсовій роботі було досліджено фреймворки для розробки ПО під мобільну платформу iOS. А саме – фреймворк ARKit для роботи з доповненою реальністю та CoreLocation для роботи з геолокацією користувача.

Проведено аналіз можливостей поєднання даних бібліотек. Досліджено способи оптимізації використання ресурсів пристрою під час роботи з бібліотеками ARKit та CoreLocation.

Результатом досліджень став мобільний навігаційний додаток на базі iOS, головною функціональністю якого є прокладання маршруту від пункту А до Б у доповненій реальності.

У розроблений проект було впроваджено досліджені способи зменшення споживання ресурсів мобільного пристрою за рахунок оптимізації роботи додатку загалом та роботи з фреймворками ARKit та CoreLocation.

Подальші можливості розвитку:

Розроблений додаток для навігації у доповненій реальності у подальшому слід опублікувати у магазин iOS додатків - AppStore. Публікація дозволить користувачам пристроїв на базі iOS використовувати застосунок для зручної навігації по місту.

Серед можливостей подальшого розвитку функціоналу додатку:

- Прокладання маршруту не лише від поточної точки до пункту призначення, а між будь-якими двома локаціями на мапі;
- Створення у додатку профілю користувача для збереження частих або улюблених маршрутів;
- Можливість кастомізації у додатку графічних елементів для навігації.

Список використаної літератури

1. Google Live Mode Update [Електронний ресурс]: https://blog.google/products/maps/new-sense-direction-live-view/

2. Google Live Mode Features [Електронний ресурс]: https://support.google.com/maps/answer/9332056?co=GENIE.Platform%3DiOS&hl=en&oco=0

3. Wikipedia – Unity [Електронний ресурс]: https://uk.wikipedia.org/wiki/Unity_(рушій_гри)

4. Недоліки кроссплатформенної розробки [Електронний ресурс]: https://vc.ru/dev/88974-chto-vybrat-krossplatformennaya-ili-nativnaya-razrabotka-mobilnogo-prilozheniya

5. Мапи у Flutter [Електронний ресурс]: https://blog.logrocket.com/adding-google-maps-to-a-flutter-app/

6. MapBox - мапи у Unity [Електронний ресурс]: https://docs.mapbox.com/unity/maps/guides/

7. Unity – AR [Електронний ресурс]: https://unity.com/unity/features/arfoundation

8. Статистика користувачів [Електронний ресурс]: https://russianblogs.com/article/29541133691/

9. Оптимізація запуску додатку [Електронний ресурс]: https://habr.com/ru/company/jugru/blog/336852/

10. Vaish G. User Interface / Gaurav Vaish // High Performance iOS Apps / Gaurav Vaish., 2016. – С. 199.

11. Оптимізація продуктивності UIKit [Електронний ресурс]: https://habr.com/ru/post/345178/

12. Продуктивність в iOS [Електронний ресурс]: https://habr.com/ru/company/e-Legion/blog/415543/

13. Apple Maps [Електронний ресурс]: https://www.apple.com/ios/feature-availability/#maps-standard

14. Pokémon Go Statistics [Електронний ресурс]: https://sensortower.com/blog/pokemon-go-one-billion-revenue-2020

15. Quora - Pokémon Go [Електронний ресурс]: https://www.quora.com/How-long-did-it-take-to-develop-Pokémon-GO

16. Habr – ARKit [Електронний ресурс]: https://habr.com/ru/company/touchinstinct/blog/331078/

17. ARKit 2 [Електронний ресурс]: https://orangeloops.com/2019/04/arkit-2-the-good-the-bad-and-the-ugly/

18. ARKit 3 [Електронний ресурс]: https://www.techrepublic.com/article/apples-arkit-everything-the-pros-need-to-know/

19. Habr - ARKit 4 [Електронний ресурс]: https://habr.com/ru/post/508584/

20. AppleInsider – підтримка ARKit мобільними пристроями [Електронний ресурс]: https://appleinsider.ru/iphone/kakie-ustrojstva-apple-poluchat-podderzhku-arkit.html

21. Andreucci G. Introduction to the Core Location Framework / Giacomo Andreucci // Pro iOS Geo / Giacomo Andreucci., 2013. – C. 206.

22. Apple Developer - Using the Significant-Change Location Service [Електронний ресурс]: https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_significant-change_location_service

23. Medium – Track user location in CoreLocation [Електронний ресурс]: https://medium.com/how-to-track-users-location-with-high-accuracy-ios/tracking-location-in-ios-vol-1-introduction-98c535e646a9

24. Habr – ARKit + CoreLocation [Електронний ресурс]: https://habr.com/ru/company/yandex/blog/421957/

25. Apple Documentation - Reducing GPU Utilization [Електронний ресурс]: https://developer.apple.com/documentation/realitykit/arview/improving_the_performance_of_a_realitykit_app/reducing_gpu_utilization_in_your_realitykit_app

26. Habr - оптимізація роботи з 2D та 3D графікою [Електронний ресурс]: https://habr.com/ru/post/169451/

27. Apple Documentation - Reducing CPU Utilization [Електронний ресурс]: https://developer.apple.com/documentation/realitykit/arview/improving_the_performance_of_a_realitykit_app/reducing_cpu_utilization_in_your_realitykit_app

28. Lim B. Location and mapping with Core Location and MapKit / Brendan Lim // iOS 7 in Action / Brendan Lim., 2014. – С. 228.

29. Apple Documentation – pausesLocationUpdatesAutomatically [Електронний ресурс]: https://developer.apple.com/documentation/corelocation/cllocationmanager/1620553-pauseslocationupdatesautomatical

30. Into to ARKit [Електронний ресурс]: https://www.pluralsight.com/guides/introduction-to-augmented-reality-with-arkit

Глосарій

Юзер – користувач;

ОС – операційна система;

Креш – неочікуване завершення роботи додатку;

AR – Augmented Reality – доповнена реальність;

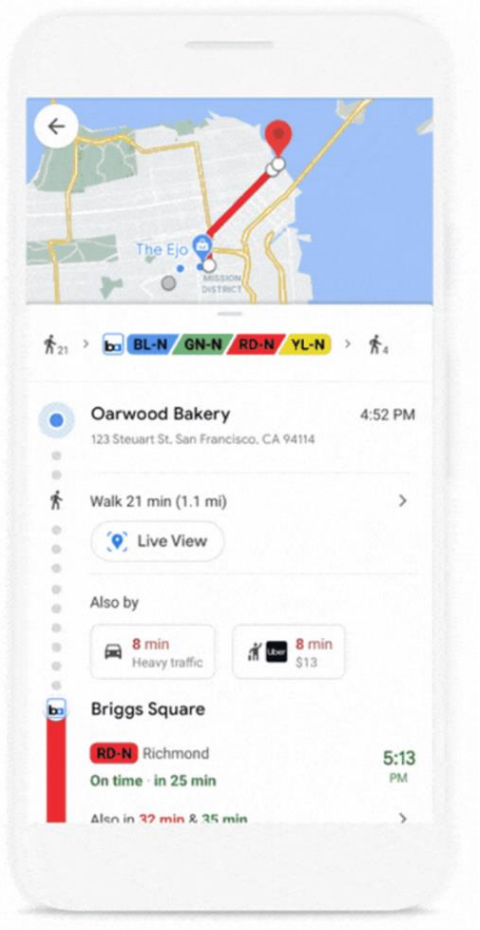
Рендеринг – відмальовка графічних елементів;

ПО – програмне забезпечення;

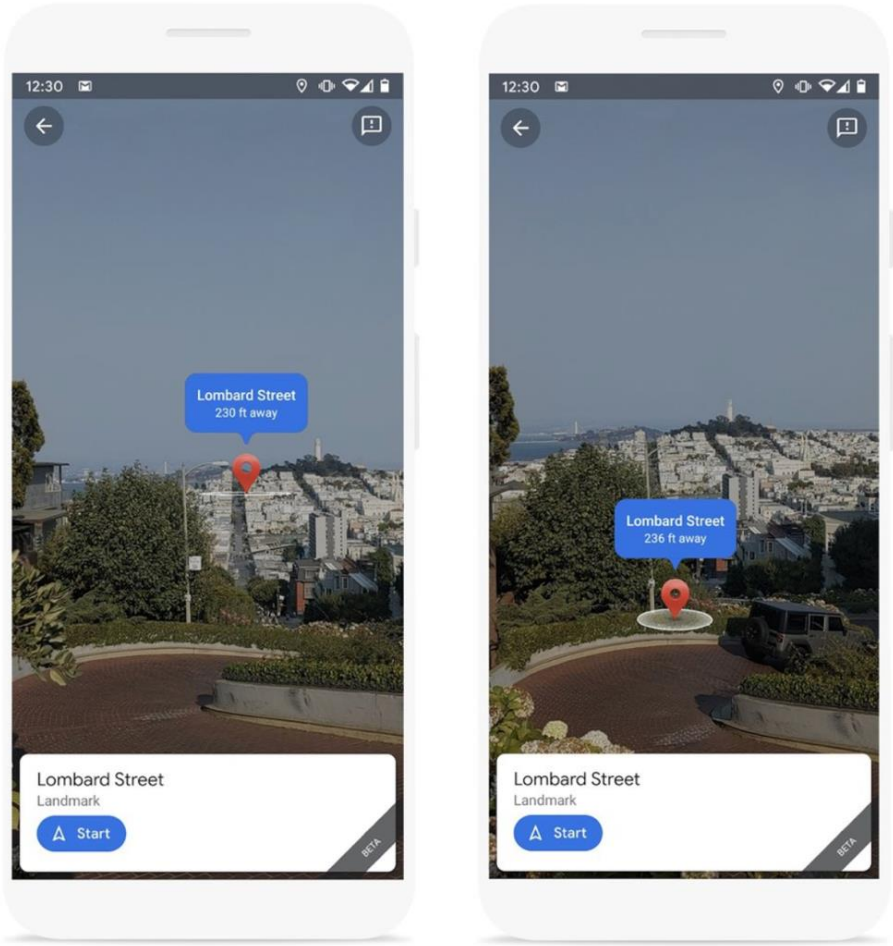
Кастомізації – підлаштовування під себе.

Додатки

Додаток А. Демонстрація роботи функції Google Maps Live Mode



Додаток Б. Покращення роботи функції Google Live Mode

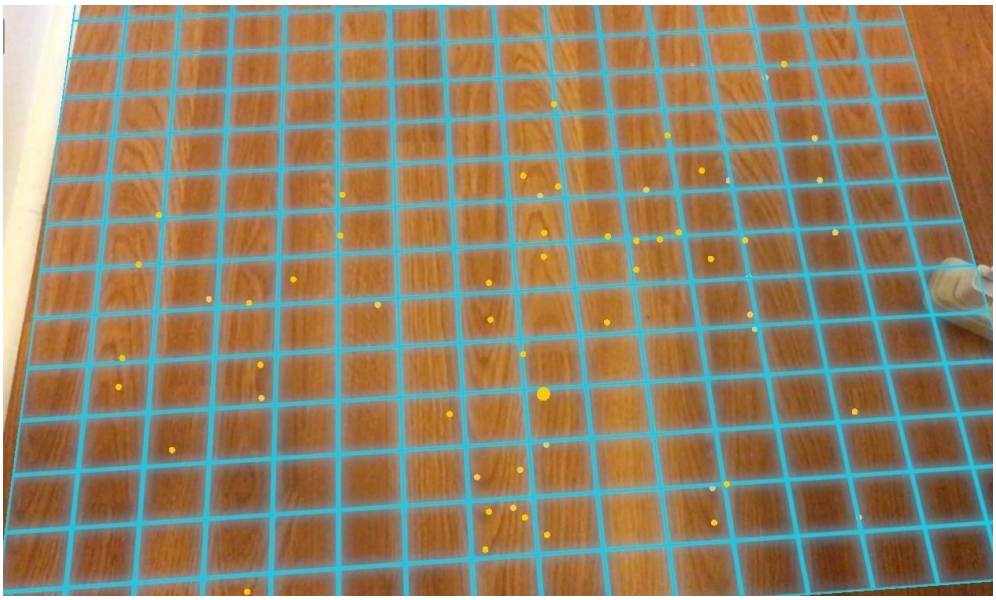


Додаток В. Можливості бібліотек доповненої реальності

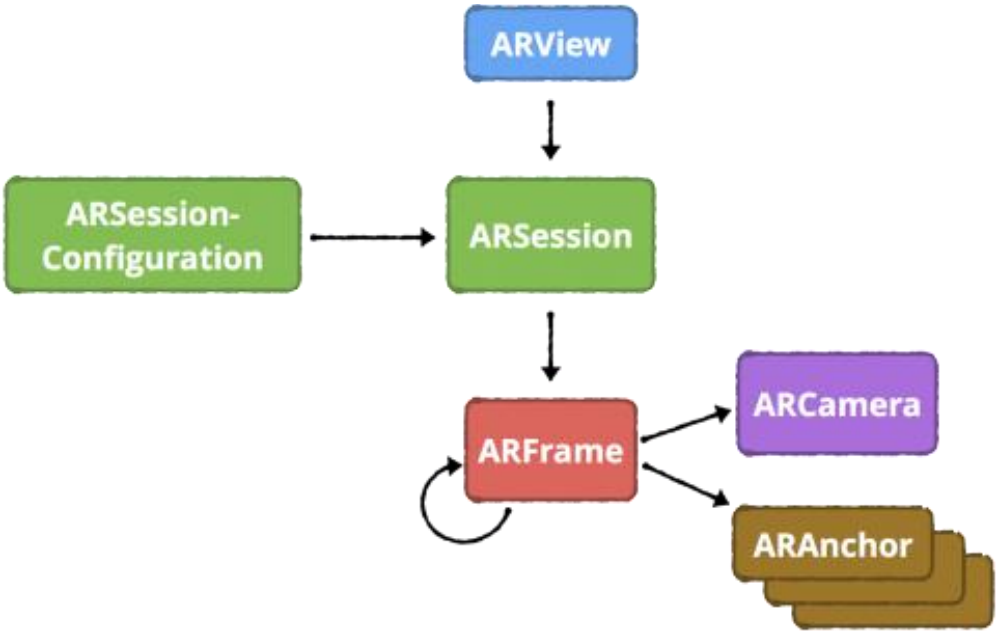
Unity's AR Foundation
Supported Features

Functionality	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
Meshing			✓	✓
2D Image tracking	✓	✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓

Додаток Г. Визначення площини у ARKit



Додаток Д. Взаємодія елементів ARKit

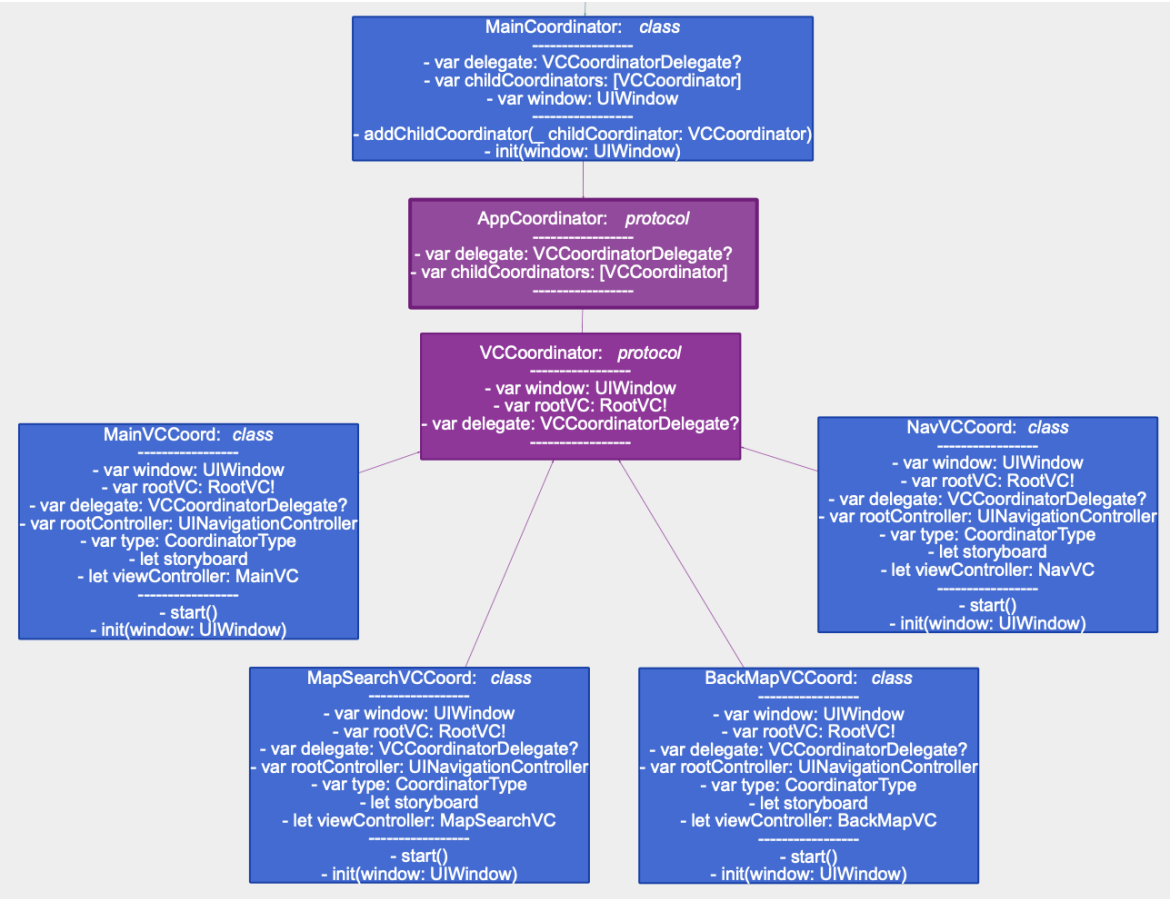


Додаток Е. Код застосування патерну Координатор у додатку.

```
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?
    var appCoordinator: MainCoordinator!

    internal func application(_ application: UIApplication,
                             didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        window = UIWindow(frame: UIScreen.main.bounds)

        if let window = window {
            appCoordinator = MainCoordinator(w: window)
        }

        return true
    }
}
```

```
class MainCoordinator: AppCoordinator {

    weak var delegate: VCCoordinatorDelegate?

    internal var childrenCoords: [VCCoordinator] = []

    var window: UIWindow

    init(w: UIWindow) {
        self.window = w
        transToCoord(type: .main)
    }

    func addChildrenCoord(_ coord: VCCoordinator) {
        coord.delegate = self
        childrenCoords.append(coord)
    }
}

extension MainCoordinator: VCCoordinatorDelegate {

    func transToCoord(type: CoordinatorType) {

        childrenCoords.removeAll()

        switch type {

        case .main:
            print("to main")
            let mainCoord = MainVCCoord(w: window)
            mainCoord.coordType = .main
            addChildrenCoord(mainCoord)
            mainCoord.delegate = self
            mainCoord.show()
```

Додаток Ж. Схема класів застосування патерну Координатор у додатку.



Додаток К. Код знаходження кроків між точками.

```
self.getStepsBetweenLocs(destLocation: self.destLocation,
                         req: MKDirections.Request()) { steps in
```

```
func getStepsBetweenLocs(destLocation: CLLocationCoordinate2D, req: MKDirections.Request,
                         completion: @escaping ([MKRoute.Step]) -> Void) {

    var steps: [MKRoute.Step] = []

    let dest = MKPlacemark(coordinate: destLocation)

    req.destination = MKMapItem.init(placemark: dest)
    req.source = MKMapItem.forCurrentLocation()
    req.transportType = .walking
    req.requestsAlternateRoutes = false

    // calculate route steps using directions
    MKDirections(request: req).calculate { res, err  in

        for route in res!.routes { steps.append(contentsOf: route.steps) }

        completion(steps)
    }
}
```

Додаток Л. Код нанесення кроків маршруту на мапу.

```
for step in steps {
    self.points.append(Annotation(title: "T " + step.instructions,
                        coordinate: step.locFromStep().coordinate))
}
```

```
func addPointsOnMap() {
    for point in points {
        DispatchQueue.main.async {
            if let title = point.title, title.hasPrefix("T") { self.pointColor = }
            else { self.pointColor = }

            self.mapView?.addAnnotation(point)
            self.mapView.addOverlay(MKCircle(center: point.coordinate, radius: 0.2))
        }
    }
}
```

Додаток М. Код перетворення кроків маршруту у леги.

```
for (index, step) in self.routeSteps.enumerated() {
    self.createRouteLegFrom(step, index)
}
```

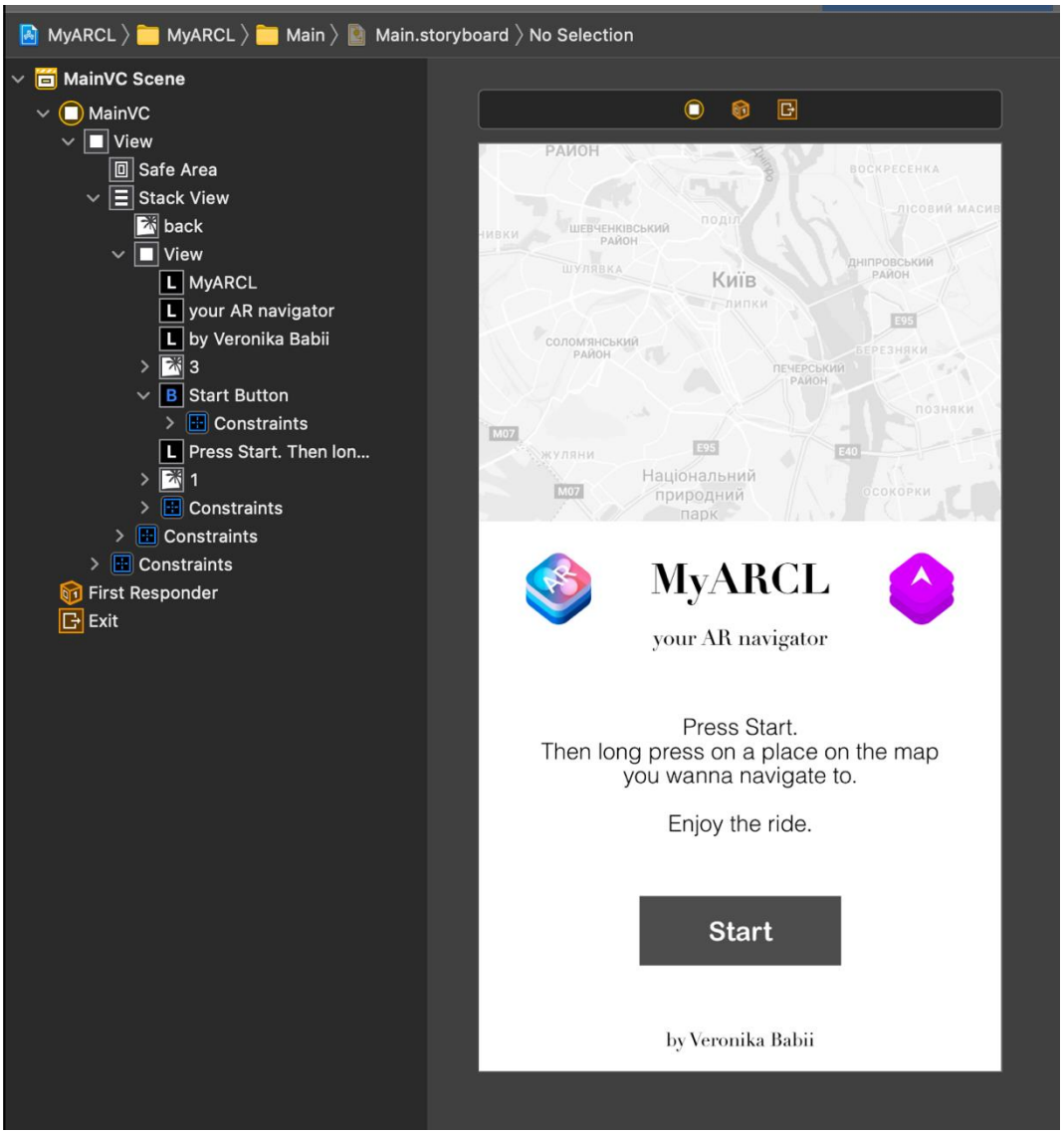
```
func createRouteLegFrom(_ routeStep: MKRoute.Step, _ index: Int) {

    // first leg
    if index == 0 {
        let nextLocation = CLLocation(latitude: routeStep.polyline.coordinate.latitude,
                                      longitude: routeStep.polyline.coordinate.longitude)

        let interSteps = CLLocationCoordinate2D.getInterLocs(currLocation: startLocation,
                                                             destLocation: nextLocation)
        currRouteLegs.append(interSteps)

        let routeLeg = RouteLeg(directions: routeStep.instructions, coordinates: interSteps)

        if !routeLegs.contains(routeLeg) { routeLegs.append(routeLeg) }
    }
    // another legs
    else {
        let prevStep = routeSteps[index - 1]
        let prevLocation = CLLocation(latitude: prevStep.polyline.coordinate.latitude,
                                      longitude: prevStep.polyline.coordinate.longitude)

        let nextLocation = CLLocation(latitude: routeStep.polyline.coordinate.latitude,
                                      longitude: routeStep.polyline.coordinate.longitude)

        let interSteps = CLLocationCoordinate2D.getInterLocs(currLocation: prevLocation,
                                                             destLocation: nextLocation)
        currRouteLegs.append(interSteps)

        let routeLeg = RouteLeg(directions: routeStep.instructions, coordinates: interSteps)

        if !routeLegs.contains(routeLeg) { routeLegs.append(routeLeg) }
```

Додаток Н. Код трансформації 3D координат.

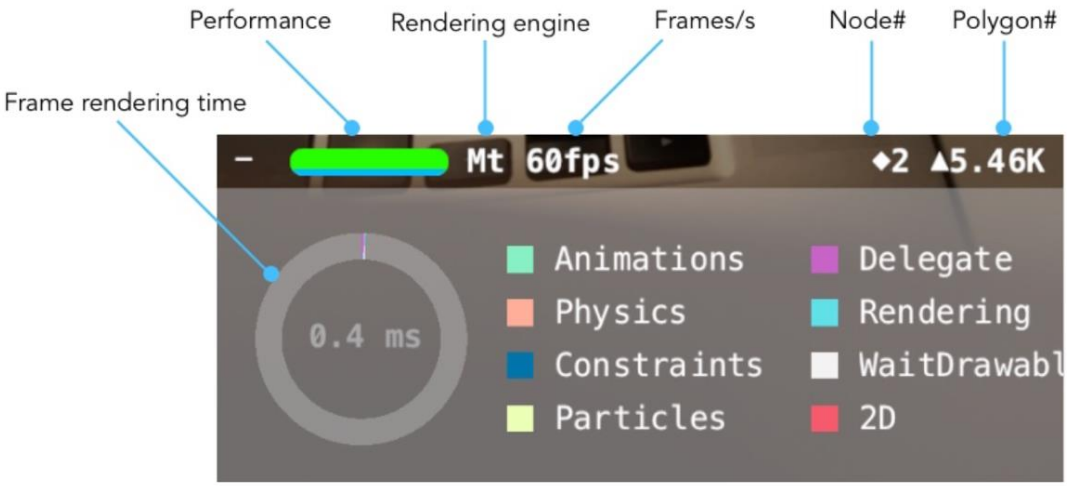
```
let locTrans = TransMatrix.transformCoordinates(startLoc: startLocation,
                                                location: stepLoc)
```

```
static func transformCoordinates(startLoc: CLLocation, location: CLLocation) -> simd_float4x4 {

    let matrix: simd_float4x4 = matrix_identity_float4x4

    let distanceToStart = Float(location.distance(from: startLoc))

    let bearing = startLoc.angleToLoc(location)

    let pos = vector_float4(0.0, 0.0, -distanceToStart, 0.0)

    let translationMatrix = self.translationMatrix(matrix: matrix_identity_float4x4, translation: pos)
    let rotationMatrix = self.rotateYCoord(matrix: matrix_identity_float4x4, degrees: Float(bearing))

    let transformMatrix = simd_mul(rotationMatrix, translationMatrix)

    return simd_mul(matrix, transformMatrix)
}
```

Додаток П. Скріншот дизайну додатку.

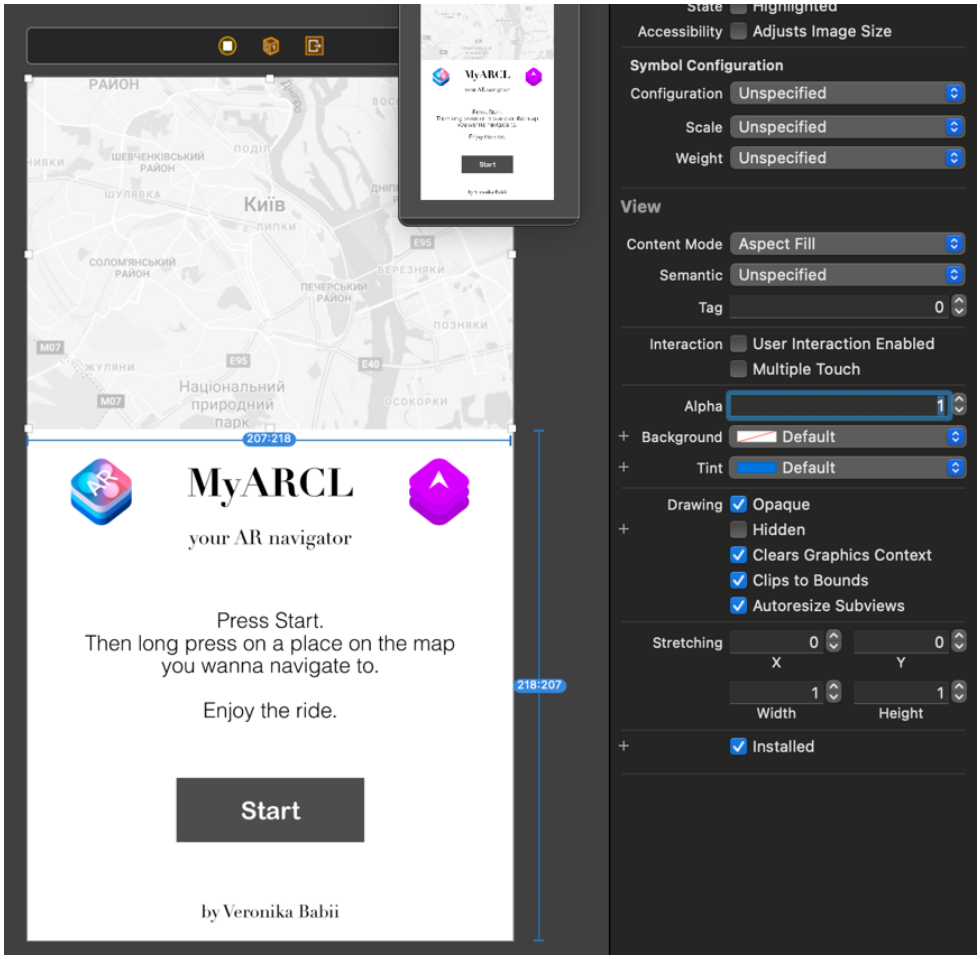


Додаток Р. ARKit Performance Statistics.

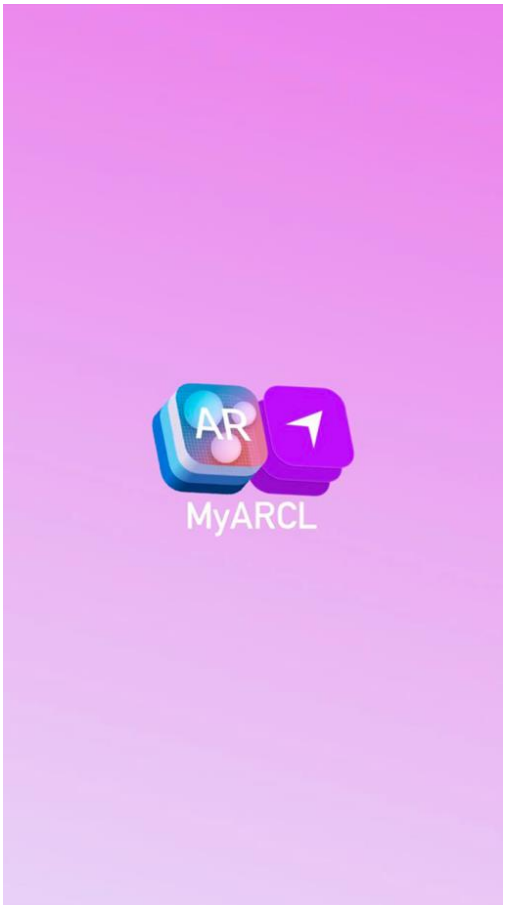


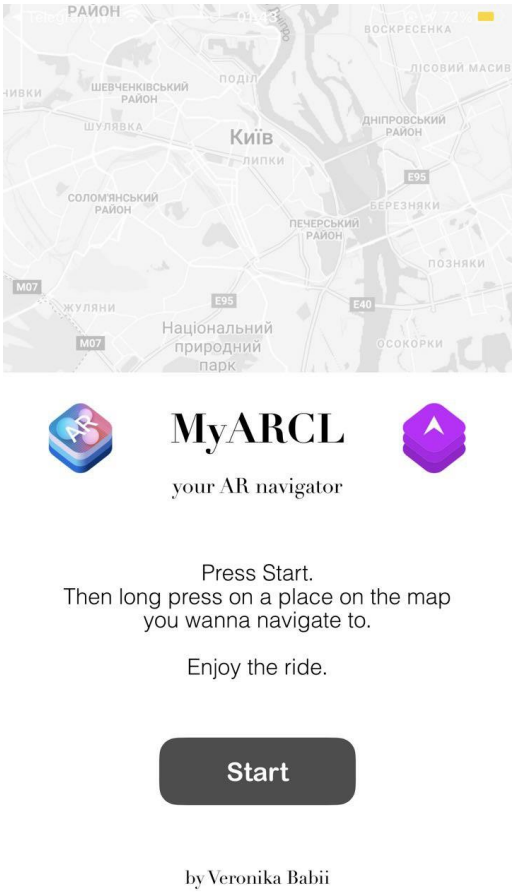
Додаток С. Скріншот вигляду елементу сфера у додатку.

Додаток Т. Фон додатку.

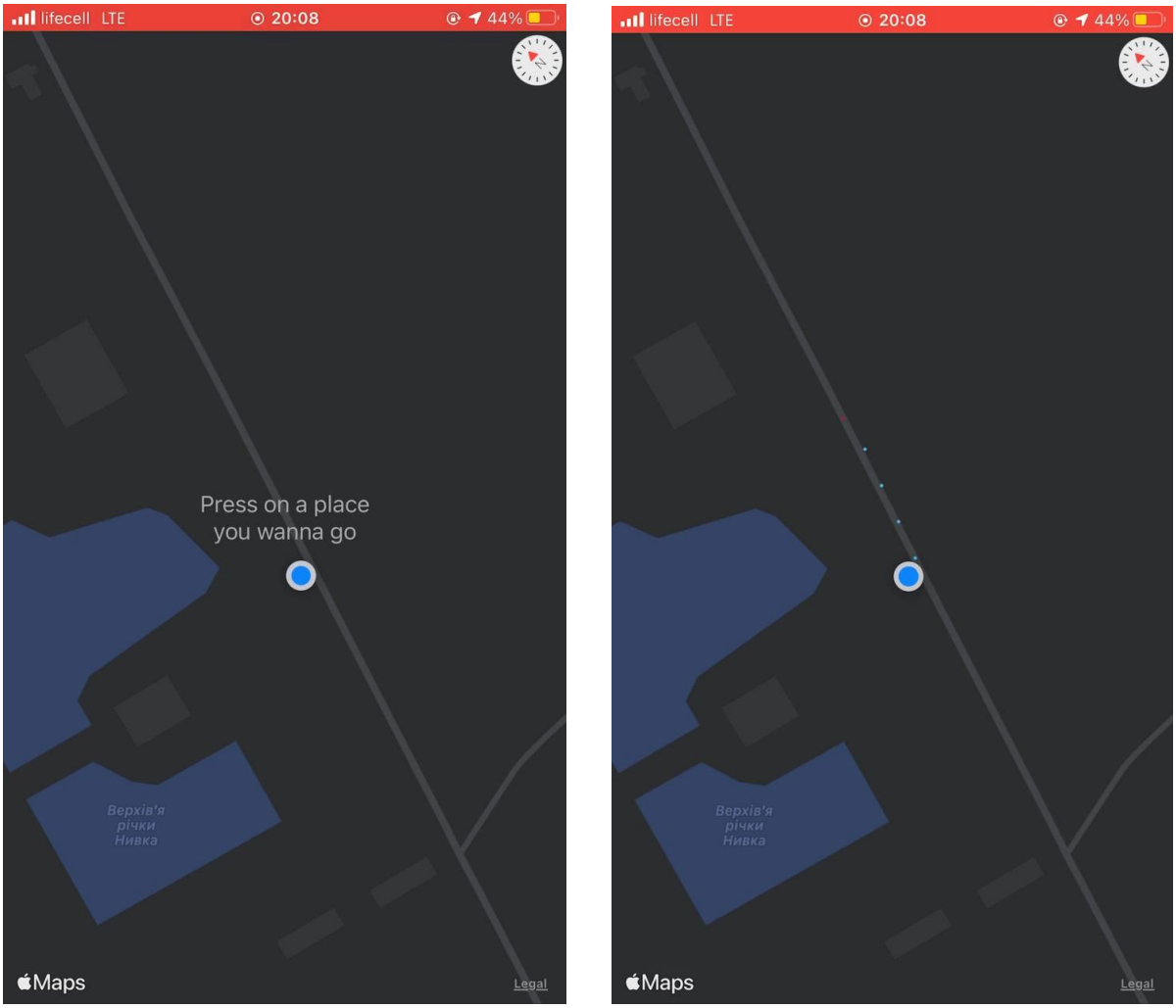


Додаток У. Скріншот екрану завантаження та головного меню додатку.





Додаток Ф. Обрання точки призначення на мапі.



Додаток Х. Екран навігації у доповненій реальності

Додаток Ц. Прокладання маршруту у доповненій реальності.