МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА/ПРОЄКТ
на здобуття освітнього ступеня бакалавра

на тему: **«РОЗРОБКА AR ЗАСТОСУНКУ НА ІГРОВОМУ РУШІЮ UNITY ДЛЯ ПОШУКУ АУДИТОРІЙ В НАУОА»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Антонюк Євгеній Михайлович

Керівник: кандидат технічних наук, доцент кафедри ЕММІТ,
Шевченко Галина Володимирівна

Рецензент: Front-end Developer "DOODLE", LLC,
Місай Володимир Віталійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)

Протокол № 11 від «18» травня 2023 р.

Острог, 2023

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*
Кафедра: *економіко-математичного моделювання та інформаційних технологій*
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри економіко-математичного моделювання
та інформаційних технологій

_____ Ольга КРИВИЦЬКА
«____» _____20__ р.

ЗАВДАННЯ
на кваліфікаційну роботу/проєкт студента
Антонюка Євгенія Михайловича
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка AR застосунку на ігровому рушію Unity для пошуку аудиторій в НаУОА.

Керівник роботи: Шевченко Галина Володимирівна, кандидат технічних наук, доцент кафедри ЕММІТ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджено наказом ректора НаУОА від "31" жовтня 2022 року №77 та *"28" квітня 2023 року №39.*

2. Термін здачі студентом закінченої роботи/проєкту: "31" травня 2023 року.

3. Вихідні дані до роботи/проєкту: Unity3D, ARCore, ARFoundation, Android SDK, 3D моделювання та візуалізація.

4. Перелік завдань, які належить виконати: налаштування робочого середовища, створення сцени та об'єктів, взаємодія з ARFoundation та ARCore, реалізація візуальних ефектів та інтерфейсу, інтеграція додаткового контенту.

5. Перелік графічного матеріалу: рисунки, таблиці.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Шевченко Г.В.	01.12.2022	01.12.2022
2	Шевченко Г.В.	01.12.2022	01.12.2022
3	Шевченко Г.В.	01.12.2022	01.12.2022

7. Дата видачі завдання: 01.12.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Затвердження теми роботи/проєкту.	до 31.10.22	
2	Постанова технічного завдання.	до 01.12.22	
3	Налаштування проєкту та середовища розробки, створення сцени та об’єктів.	до 25.02.23	
4	Створення 3D моделі приміщення, взаємодія з ARFoundation та ARCore.	до 14.04.23	
5	Розробка інтерфейсу користувача.	до 01.05.23	
6	Реалізація візуальних ефектів та додаткових можливостей додатку.	до 08.05.23	
7	Вичитування останньої версії додатку, тестування, внесення правок.	до 12.05.23	
8	Тестування додатку та його оптимізація.	до 13.05.23	
9	Попередній захист кваліфікаційної роботи/проєкту.	до 18.05.23	
10	Здача кваліфікаційної роботи/проєкту на кафедру.	до 31.05.23	

Студент: _____ Антонюк Є.М.

(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи: _____ Шевченко Г.В.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ
кваліфікаційної роботи/проєкту
на здобуття освітнього ступеня бакалавра

Тема: Розробка AR застосунку на ігровому рушію Unity для пошуку аудиторій в НаУОА
Автор: Антонюк Євгеній Михайлович
Науковий керівник: Шевченко Галина Володимирівна

Захищена «_____»_____ 20__ року.

Пояснювальна записка до кваліфікаційної роботи: 75 *(кількість сторінок роботи) с.,* 25 *(кількість рисунків) рис.,* 11 *(кількість таблиць) табл.,* 0 *(кількість додатків) додатків,* 16 *(кількість джерел) джерел.*
Ключові слова: AR застосунок, Unity, 3D.
Короткий зміст праці: Кваліфікаційна робота на тему: «Розробка AR застосунку на ігровому рушію Unity для пошуку аудиторій в НаУОА» присвячена розробці AR застосунку для пошуку кабінетів з можливістю користування на платформі Android. Проблемний пошук кабінетів в академії актуальний для багатьох студентів та викладачів, тому розробка AR додатку може значно полегшити процес орієнтування в просторі.

У процесі розробки були використані сучасні засоби, такі як мова програмування C#, інтегроване середовище розробки Microsoft Visual Studio, графічний редактор Figma та середовище для створення 3D моделей Blender. Ігровий рушій Unity був обраний як оптимальне рішення для розробки додатку через його високу функціональність та можливості створення AR додатків.

Результатом роботи є функціональний AR додаток, який дозволяє швидко та легко знайти потрібну аудиторію, використовуючи функцію камери на мобільному пристрої.

The qualification work on the topic: "Development of an AR application on the Unity game engine for finding classrooms in the National University of Applied Sciences" is devoted to the development of an AR application for finding classrooms with the ability to use the Android platform. The problematic search for classrooms in the academy is relevant for many students and teachers, so the development of an AR application can greatly facilitate the process of orientation in space.

In the development process, modern tools were used, such as the C# programming language, the Microsoft Visual Studio integrated development environment, the Figma graphic editor, and the Blender 3D modeling environment. The Unity game engine was chosen as the optimal solution for developing the application due to its high functionality and the ability to create AR applications.

The result is a functional AR application that allows you to quickly and easily find the right audience using the camera function on your mobile device.

(підпис автора)

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

APK – це формат файлів, який використовується для розповсюдження та встановлення мобільних додатків на операційній системі Android. Він містить всі компоненти програми, включаючи код, ресурси, мультимедіа, конфігураційні файли та інші необхідні елементи. Файли APK можна завантажити з різних джерел, включаючи Google Play Store, альтернативні маркетплейси, веб-сайти та інші джерела, але користувачі повинні дозволяти встановлення додатків з невідомих джерел, щоб уникнути загроз безпеці.

SDK – це набір програмних інструментів, які дозволяють розробникам створювати додатки для певної платформи або середовища розробки. Наприклад, Android SDK надає інструменти для створення додатків для пристроїв Android. SDK може бути доступним для різних платформ і мов програмування і дозволяє розробникам створювати програми за допомогою стандартних інструментів і функцій, що надаються платформою або середовищем розробки, роблячи розробку більш ефективною і швидкою.

Android – це операційна система для мобільних пристроїв, розроблена компанією Google. Вона зазвичай встановлюється на смартфонах, планшетах, годинниках, телевізорах та інших пристроях.

Scene – це контейнер, який містить об'єкти, створені та розміщені у 3D або 2D віртуальному ігровому просторі. Сцена визначає середовище, в якому гравець взаємодіє з грою, і може містити різні елементи, такі як об'єкти, камери, світло, сценарії, звуки та інші компоненти. В Unity можна створити декілька сцен, кожна з яких містить різні ігрові елементи. Сцени є важливим елементом для розробки віртуальних ігор та інших програм з 3D або 2D графікою.

QR-Code – матричний код (двовимірний штрих-код), розроблений і представлений японською компанією «Denso-Wave» в 1994 році. Основна перевага QR-коду — це легке розпізнавання сканувальним обладнанням (в тому числі й фотокамерою мобільного телефона), що дає можливість використання в торгівлі, на виробництві, в логістиці.

AR – є скороченням від "Augmented Reality" (доповнена реальність) і є технологією, що дозволяє додавати віртуальний контент (такий як об'єкти, анімацію, звуки та ін.) до реального світу. У AR використовуються різні пристрої, такі як смартфони, планшети, AR-окуляри та інші, для того, щоб відображати віртуальні об'єкти у реальному часі в реальному оточенні. AR дозволяє користувачам взаємодіяти з віртуальними об'єктами у реальному світі, що дозволяє створювати нові можливості для навчання, розваг та інших сфер життя.

Debug – це процес виявлення, аналізу та виправлення помилок у програмному коді. Налагодження є важливою частиною розробки програмного забезпечення,

оскільки дозволяє розробникам виявляти та виправляти проблеми в коді. Зазвичай налагодження передбачає використання спеціалізованих інструментів і методів для відстеження виконання програми, відображення значень змінних, моніторингу викликів функцій та інших дій, які допомагають виявити і виправити помилки.

Assets – це будь-який файл, який може бути використаний в проекті Unity, такий як зображення, звук, 3D-моделі, скрипти та інші елементи. Asset може бути створений в Unity або імпортований з інших джерел, таких як зовнішні програми чи інтернет. Assets відіграють важливу роль в розробці ігор та інших програм, розроблених на Unity. Наприклад, 3D-моделі використовуються для створення об'єктів та персонажів в грі, зображення використовуються для створення текстур та інтерфейсу користувача, а звук використовується для створення атмосфери та ефектів звуку.

Prefab – це збережений елемент, який містить один або більше готових до використання об'єктів, які можуть бути повторно використані в різних частинах проекту. Префаб вміщує в себе всі компоненти та налаштування об'єкта, що дозволяє швидко створювати та редагувати об'єкти в проекті. Префаб є важливою складовою розробки ігор та інших програм в Unity, оскільки дозволяє розробникам швидко створювати та редагувати об'єкти в проекті, що використовуються у багатьох частинах гри. Наприклад, префаб може містити повторювані елементи гри, такі як перешкоди, декорації, інтерактивні об'єкти та інші, які можуть бути повторно використані в різних рівнях гри.

Rendering – це процес відображення 3D-сцени на екрані комп'ютера або іншого візуального пристрою. Рендеринг відображає 3D-сцену згідно з налаштуваннями освітлення, матеріалів та інших параметрів, що дозволяє створювати реалістичні та деталізовані зображення. Unity має вбудований рендеринговий двигун, який дозволяє створювати візуалізації 3D-сцен з високою якістю зображення. Рендеринг в Unity включає в себе такі елементи, як обробка освітлення, текстур, матеріалів, тіней та інших ефектів.

Скрипт – це програмний код, написаний на мові програмування, який додає функціональність та поведінку до об'єктів в 3D-сцені. Скрипти можуть бути додані до різних об'єктів, таких як персонажі, предмети, камери та інші, що дозволяє розробникам створювати складні інтерактивні елементи гри. Скрипти в Unity написані на мовах програмування, таких як C# або JavaScript, і виконуються в середовищі Unity. Скрипти можуть виконувати різні дії, такі як переміщення об'єктів, зміна їх властивостей, розрахунок фізики та інші.

Sprite – це 2D-зображення, яке може бути використане для створення графічних елементів в проекті Unity, таких як персонажі, об'єкти та інтерфейс користувача. Спрайти можуть бути створені в Unity або імпортовані з інших джерел, таких як зовнішні програми чи інтернет. Спрайти використовуються для створення

2D-графіки в Unity. Вони можуть бути розміщені на 2D-площинах, які називаються Sprite Renderer, або використовуватися для створення інтерфейсу користувача, який відображається на екрані.

ПЗ – це скорочення від "програмне забезпечення". Це загальний термін, який використовується для опису комп'ютерних програм, які призначені для виконання конкретних завдань або функцій на комп'ютері чи інших електронних пристроях. Програмне забезпечення може бути розроблено для різних цілей, включаючи бізнес, науку, освіту, розваги та інші сфери діяльності. Відомі приклади програмного забезпечення включають операційні системи, текстові редактори, антивірусні програми, графічні редактори, браузери та багато іншого.

UI – означає "User Interface" (користувацький інтерфейс) і відноситься до візуальних елементів, які використовуються для взаємодії з грою або додатком. Це можуть бути кнопки, текстові поля, списки, вікна, меню та інші елементи, які дозволяють користувачу взаємодіяти з програмою та виконувати різноманітні дії. У Unity можна створювати UI елементи за допомогою інструментів, які дозволяють редагувати та налаштовувати їх вигляд та поведінку.

Push – повідомлення – це коротке повідомлення, яке надсилається на мобільний пристрій користувача безпосередньо з сервера застосунку або платформи. Ці повідомлення можуть містити різноманітну інформацію, наприклад, новини, рекламу, сповіщення про події, нагадування про дедлайни, повідомлення від соціальних мереж та інше.

ВСТУП

Актуальність – дослідження полягає в розв'язанні проблеми навігації в приміщеннях та наданні відвідувачам послуг на основі їх місцезнаходження (Location-based service, LBS). В сучасному світі, де будівлі, торгові центри та розважальні комплекси стають все більш об'ємними та складними, орієнтуватися в них можливо лише для тих, хто постійно відвідує такі місця. Це стає викликом для непідготовленої людини, яка може втратитися в приміщенні. Рішення, що використовуються в indoor-навігації, також можуть допомогти в орієнтуванні на відкритих просторах, де використання систем супутникової навігації працює некоректно.

Використання QR-кодів дозволяє досягти достатньої точності за прийнятним рівнем фінансових витрат і є перспективною технологією в розробці indoor-навігації. Більшість сучасних на ринку навігаційних технологій для приміщень засновані на QR-кодах і використовують AR для візуалізації маршрутів.

Поширення додатків для внутрішньої навігації на базі AR в різних споживчих секторах в майбутньому буде зростати, оскільки користувачі все більше розбираються в цифрових технологіях і готові застосовувати нові технології в своєму повсякденному житті. Це створює великі комерційні перспективи для розробки indoor-навігації та AR-навігації, який вже привернув увагу таких великих гравців на ринку, як Google, Apple, Qualcomm, Broadcom, Sony і т.д.

Метою розробки мобільного додатку є реалізація Indoor-навігації та навігації з використанням доповненої реальності AR всередині академій для платформи Android. Для досягнення цієї мети використовуються сучасні алгоритмічні та програмні засоби, а також інтегроване середовище розробки та ігровий рушій Unity, який містить всі необхідні інструменти та бібліотеки. Розроблений додаток дозволить користувачам швидко та ефективно орієнтуватися в приміщенні та отримувати інформацію про місця, які їх цікавлять, засновану на їх місцезнаходженні.

Мета дослідження – полягає в розробці ігрового додатку на базі рушію Unity, який дозволить розширити функціональні можливості навігації всередині Національного університету «Острозька академія» з використанням Indoor-навігації та навігації з використанням доповненої реальності AR. Розробка такого додатку має на меті вирішення проблеми навігації усередині приміщення та зменшення часу, який витрачається студентами на пошук необхідної локації в академії.

Додаток буде містити детальну карту всієї території академії, що дозволить користувачам швидко знайти необхідні локації та отримати інформацію про розташування аудиторій, бібліотеки, студентської їдальні та інших корисних місць.

Крім того, застосунок буде забезпечувати можливість використання AR-технологій для візуалізації маршрутів та відображення необхідної інформації в режимі реального часу.

Розробка такого додатку не тільки допоможе студентам та викладачам швидше та ефективніше орієнтуватися в академії, але і стане важливим кроком у розширенні функціональних можливостей навігації всередині будівель та інших приміщень. Такий застосунок можна буде використовувати не тільки в академічному середовищі, але й у торгових центрах, аеропортах, музеях та інших об'єктах з великою кількістю локацій.

Задачі дослідження:

- Аналіз предметної області;
- Проектування інформаційної системи;
- Вибір інструментарію, методів реалізації та тестування ПП.

Об’єктом дослідження є процес розробки мобільного додатку з Indoor-навігацією та навігацією з використанням доповненої реальності AR всередині академій для покращення і пришвидшення навігації всередині приміщень.

Предметом дослідження є технології: ігровий рушій Unity, тривимірний графічний редактор Blender, мова програмування C#, двовимірний графічний редактор Figma.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

- проаналізувати існуючі методи, підходи, а також інструменти для оцінки ефективності використання навігації всередині приміщень;
- розробити модель оцінки ефективності використання навігації всередині приміщень, а також критерії покращення процесу розробки та вихідного коду;
- розробити модель візуального відображення Indoor-навігації та навігації з використанням доповненої реальності AR;
- розробити програмний модуль взаємодії з QR-кодами;
- перевірити розроблені теоретичні положення, методи, алгоритми та інструменти на практиці.

Методи дослідження. В процесі дослідження застосовуються методи indoor-навігації, методи доповненої реальності AR, методи навігації з використанням QR-кодів, методи візуального представлення отриманих результатів.

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Визначення Indoor-навігації

Оскільки, GPS навігація не працює в будівлях, тут використовуються інші технології позиціонування, коли потрібно автоматичне позиціонування. Wi-Fi, маяки або QR-коди часто використовуються в цьому випадку для створення так званого «внутрішнього GPS».

Однак, на відміну від GPS, вони також дозволяють визначити фактичний рівень поверху. Для більшості програм потрібна функція "маршрутизації в приміщенні", яка точно веде людей по будівлі за допомогою програми навігації в приміщенні, і таким чином автоматично визначає їхнє положення - дуже схоже на навігаційні системи, які ми використовуємо в наших автомобілях.

Навігація в приміщенні з автоматичним позиціонуванням зазвичай використовується як клієнтська програма. Це означає, що позиція визначається безпосередньо на смартфоні користувача і, отже, необхідно розробити додаток, який буде виконувати ці функції. Місцезнаходження визначається, як правило, за допомогою Wi-Fi, маяків або Qr-кодів. Також доступний канал зворотного зв'язку, наприклад, для надсилання push-повідомлень. Серверний підхід також можливий, можна зіткнутись з технічними проблемами.

1.2. Опис предметного середовища (функціональної моделі, процесу діяльності)

Для реалізації застосунку «Indoor-OA-DEMO» необхідно було вибрати технології, які будуть використовуватись для його створення. Було обрано наступні технології для роботи: мова програмування C#, ігровий рушій Unity, середовище розробки Visual Studio, графічний редактор Blender та Figma.

Платформа Android

Наявність операційної системи (ОС) - головна особливість, яка відрізняє смартфон від звичайного мобільного телефону. При виборі конкретної моделі телефону або пристрою, операційна система часто є визначальним фактором.

Android – операційна система і платформа для мобільних телефонів, планшетів, електронних книг, навігаторів, смарт годинників, фітнес-браслетів, фотоапаратів, ігрових приставок, холодильників, ноутбуків, окулярів Google Glass, телевізорів та інших пристроїв.

Платформа заснована на власній реалізації віртуальної машини Java, а також ядрі Linux для мобільних пристроїв. Операційною системою володіє Google, яка купила її в 2005 році в компанії Android.Inc., яку потім купила Google. Android

відкриває можливість створенню Java-додатків, які керують пристроєм через бібліотеки, розроблені Google.

Крім того, можна писати програми на C та іншими мовами програмування за допомогою Android Native Development Kit 1.5 (Cupcake), випущеної 30 квітня 2009 р. Серед основних поліпшень з'явилася підтримка запису і перегляду відео в режимі камери; підтримка 19 Bluetooth A2DP; можливість автоматичного підключення до Bluetooth гарнітури.

Розробка мобільних ігрових додатків має низку послідовних етапів. Але щоб створити мобільний додаток потрібен певний інструмент, особливо, якщо цей додаток ігровий. На практиці використовуються рушії, де вже є реалізована велика кількість потрібних функцій для створення ігрових додатків.

Плюси Android:

- Відкритість та безкоштовність операційної системи, що дозволяє розробникам створювати додатки безкоштовно та без обмежень.
- Широкий вибір мобільних пристроїв у різних цінових категоріях та з різними характеристиками, що дозволяє користувачам знайти оптимальне рішення для своїх потреб.
- Багатофункціональність та можливості налаштувань, що дозволяє користувачам налаштовувати операційну систему на свій смак та використовувати її для різних задач.
- Велика кількість додатків у Google Play Market, що дозволяє користувачам знайти необхідний додаток для своїх потреб.

Мінуси Android:

- Недостатня оптимізація операційної системи на деяких мобільних пристроях, що може призводити до поганої продуктивності та витрати батареї.
- Безпека операційної системи може стати проблемою через велику кількість різноманітних пристроїв, які використовують Android, та різні рівні оновлень безпеки.
- Низька стійкість до зламів та вірусів порівняно з іншими операційними системами.
- Надмірна відкритість може призвести до проблем з приватністю та безпекою користувачів, зокрема через можливість встановлення додатків з ненадійних джерел.

Мова програмування C#

C# – об'єктно-орієнтована мова програмування, створена спеціально для роботи у середовищі Microsoft .NET Framework. Мова C# була розроблена з урахуванням сильних і слабких особливостей інших мов, зокрема Java і C++. Специфікація мови C# була написана Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде. Андерс Хейлсберг відомий у світі програмування як автор компілятора Turbo Pascal і лідер команди, яка створила Delphi.

Ключові особливості мови C#:

- Компонентна орієнтованість
- Код зібраний воєдино (декларації і реалізації об'єднані разом)
- Уніфікована система типів і їх безпечність
- Автоматична і мануальна робота за пам'яттю
- Використання єдиної бібліотеки класів - CLR

Специфікація C# визначає мінімальний набір бібліотек типів і класів, на який має розраховувати компілятор. На практиці, C# найчастіше використовується з якоюсь реалізацією Common Language Infrastructure. Задля зручності розробки на проекті був використаний редактор коду Visual Studio Code.

Плюси C#:

- Потужна та ефективна мова програмування з високою продуктивністю та швидкодією виконання.
- Легко вивчити та користуватися, оскільки має чіткий синтаксис та вбудовані бібліотеки, що дозволяють розробникам писати програми швидко та ефективно.
- Мова програмування C# має широкі можливості для розробки додатків для різних платформ, включаючи Windows, Android та iOS.
- Має вбудовані функції для роботи з базами даних, що дозволяє легко створювати програми, які працюють з даними.

Мінуси C#:

- Менш популярна в порівнянні з іншими мовами програмування, такими як Java або Python.
- Вимагає від розробників знання платформи .NET Framework, що може бути складнішим для початківців.
- Має меншу підтримку спільноти порівняно з іншими мовами програмування, що може призвести до меншої кількості матеріалів для навчання та меншої підтримки.

- Обмежені можливості для розробки веб-додатків порівняно з іншими мовами програмування, такими як JavaScript або PHP.

Visual Studio

Visual Studio Code - це інтегроване середовище розробки (IDE) для створення програмного забезпечення на платформі Microsoft Windows та інших платформах, таких як Android, iOS та веб-платформи. Visual Studio містить різноманітні інструменти для розробки, тестування та налагодження програмного забезпечення. Це середовище надає розробникам доступ до різних мов програмування, таких як C#, C++, Visual Basic та інших, а також до бібліотек та інструментів для створення різноманітних додатків, включаючи додатки для роботи на різних платформах та пристроях. Visual Studio дозволяє розробникам працювати з кодом, візуальною графікою та іншими компонентами додатка у зручному та інтуїтивно зрозумілому інтерфейсі.

Редактори коду дозволяють зручно та легко розробляти інтерфейси програм, чи веб-сайтів, мають настроюваний зовнішній вигляд, підсвічування синтаксису, можливість додавання сторонніх плагінів для покращення взаємодії з користувачем, що дає високі переваги над типовими текстовими процесорами.

Переваги Visual Studio:

- Безкоштовний.
- Велика кількість плагінів.
- Динамічний набір тексту.
- Підтримка Git.
- Невелика вага – редактор не потребує значних ресурсів операційної системи.

Unity

Unity - це інтегроване середовище розробки (IDE) для створення ігрових додатків та інтерактивних візуальних додатків. Unity дозволяє розробникам створювати додатки для різних платформ, таких як Windows, macOS, Linux, Android, iOS, PlayStation, Xbox та інших. Unity має потужний двигун гри, який дозволяє розробникам створювати 2D та 3D ігри з використанням різних мов програмування, таких як C#, C++, JavaScript та інших. У Unity є велика кількість готових компонентів та ресурсів, таких як зображення, звуки та інші ефекти, які допомагають розробникам створювати додатки швидко та ефективно. Unity також має вбудовані інструменти для розробки додатків з використанням доповненої реальності (AR) та віртуальної реальності (VR). Unity дозволяє розробникам створювати AR-та VR-додатки для різних пристроїв, таких як мобільні пристрої, розумні окуляри та інші.

На веб-сайті Unity представлено кілька тарифних планів рушія, відрізняються вартістю і рівнем підтримки (рис. 1.1). Кожна з версій цього програмного забезпечення відкриває для користувача новий функціонал, який відсутній в базовій версії. Відмінності версій один від одного представлені на рисунку 1.1.

	Personal Бесплатно Начните творить с бесплатной версией Unity Начать Вы студент? Оформите бесплатную студенческую подписку!	**Plus** 399 $ в год за место Больше возможностей и ресурсов для развития ваших проектов Выбрать тариф	**Pro** 2 040 $ в год за место Готовое профессиональное решение для создания и поддержки контента Выбрать тариф	**Enterprise** Масштабный успех для крупных организаций с амбициозными целями Свяжитесь с нами Для больших команд
Создать				
ⓘ Платформа Unity для разработки в реальном времени	✓	✓	✓	✓
ⓘ Визуальный скриптинг Unity	✓	✓	✓	✓
ⓘ Система контроля версий Unity Plastic SCM (3 пользователя и 5 ГБ хранилища)	✓	✓	✓	✓
ⓘ Управление заставочным экраном	—	✓	✓	✓
ⓘ Развертывание на игровых консолях	—	—	✓	✓

Рис. 1.1. Вартість передплати рушія Unity

Так само можна побачити основні відмінності версій та їх вартість. У нашому випадку буде використана версія Personal, вона є безкоштовною, не має обмежень для проекту, а найголовніше, має практично все необхідне для старту розробки з нуля.

Це потужне, але в той самий час просте в роботі ПЗ, що дозволяє створювати і випускати 2D і 3D-додатки. Розробка мобільних додатків на Unity відкриває перед розробниками безліч можливостей платформи для підтримки та монетизації створених ігор.

У 2008 році Unity почали співпрацювати з iOS, з Android — у 2010, а далі розробники змогли створювати свої проекти і для ігрових консолей Xbox і Playstation.

Понад 50% всіх мобільних додатків розроблено саме на Unity. Спочатку Unity призначався для розробки на комп'ютерах Mac, пізніше з'явилося оновлення, що дозволяє працювати з Windows.

Плюси рушія Unity:

- Зрозумілий редактор та інструментарій: за декілька днів основні речі може освоїти навіть той, хто вперше стикається з розробкою застосунку. А якщо питання залишаються — відповіді є на одному з багатьох ресурсів, форумів, та

у відеоуроках на YouTube. Створення додатку на Unity буде під силу навіть школяреві.

- Сучасний рівень графіки, здатний конкурувати з більш дорогими рушіями. Unity, безумовно, програє UnrealEngine за можливостями, але може працювати із освітленням, стандартним набором постпроцессингових ефектів, SSAO.
- Ігровий рушій Unity поширюється умовно безкоштовно. Платити потрібно буде тільки за розширення пакетів підписки. На ліцензії кілька разів на рік бувають знижки, зазвичай -20%.
- Велика спільнота розробників.
- Безліч опублікованих додатків.
- Внутрішній магазин готових рішень, де можна купити готові фрагменти коду, текстури та звуки.
- Можливість створення фотореалістичної графіки.
- Розробка на Unity дозволяє легко імпортувати створені програмні застосунки між ОС Windows, Linux, OS X, Android, iOS, на консолі PlayStation, Xbox, Nintendo, на VR- і AR-пристрої.

Мінуси середовища розробки Unity:

- Розробка додатку на Unity вимагає навичок програмування.
- Безліч вбудованих компонентів роблять продукт об’ємним. Це може стати проблемою, адже користувачі не люблять завантажувати великі додатки, а в деяких країнах (наприклад, Індія та Бразилія) люди користуються недорогими, слабкими гаджетами, які не потягнуть ваш додаток.
- У розробників немає доступу до вихідного коду власного додатку. Вам доведеться чекати, поки інженери Unity самі це зроблять. Початкових кодів вам не дадуть навіть по платній ліцензії.
- Немає інтеграції із зовнішніми сервісами та бібліотеками (наприклад, Facebook), розробники змушені налаштовувати це вручну.
- Неможливість додати до рушія стороннюю фізику.

ARKit і ARCore

Зростаючу популярність AR в розробці додатків можна віднести до двох великих технологічних гігантів, таких як Apple і Google, які розробили власні набори інструментів розробки AR з фреймворками і SDK, що включають ARKit і ARCore відповідно.

ARKit і ARCore є одними з найпопулярніших наборів інструментів, які необхідні для створення програми доповненої реальності. Вони спростили розробку додатків з використанням AR. Хоча всі вони можуть запропонувати хороші

можливості, є невеликі відмінності, які розробники повинні знати і враховувати при виборі правильних інструментів, фреймворка і SDK для розробки додатків AR.

Основні пропозиції ARKit - це точки, виявлення площини, карта світу AR, оцінка освітленості, прив'язки, відстеження особи, захоплення руху, визначення людей і сеанси спільної роботи.

ARKit підтримує виявлення і відстеження зображень, а також допомагає вбудовувати віртуальні об'єкти в додатки AR або на поверхні.

ARCore - це набір інструментів Google (фреймворк і SDK) для створення додатків доповненої реальності. Перевага ARCore - це те, що він підтримує розробку як для платформ Android (7.0 і вище), так і для iOS (11 і вище). Крім того, цей набір інструментів для розробки доповненої реальності доступний безкоштовно.

ARCore пропонує точки, визначення площини, пози, оцінку освітленості, прив'язки, відстеження зображень, відстеження осіб, перекриття об'єктів і прив'язки до хмари. Ці можливості і результати аналогічні ARKit.

Завдяки можливостям відстеження руху ARCore розробники можуть відстежувати стан телефону щодо навколишнього середовища. Іншими важливими можливостями ARCore є розуміння навколишнього середовища, включаючи визначення розміру і місця розташування поверхонь і оцінку освітленості, включаючи реальні умови освітлення.

Завдяки можливостям відстежування місцеположення в реальному часі і інтеграції віртуальних і реальних об'єктів, ARCore є відмінним інструментом для доповненої реальності в додатках електронної комерції. Він дозволяє розміщувати об'єкти і текст у фізичному оточенні.

ARKit і ARCore - популярні набори інструментів для розробки доповненої реальності. ARKit підтримує тільки платформу iOS, в той час як ARCore підтримує як iOS, так і Android.

Blender

Blender - це вільний та відкритий 3D-графічний пакет, який використовується для створення візуальних ефектів, анімації, комп'ютерних ігор, моделювання, візуалізації та інших задач в області комп'ютерної графіки. Blender є доступним для використання на різних платформах, таких як Windows, macOS та Linux.

Blender має потужний набір інструментів для створення 3D-об'єктів, які включають у себе різноманітні методи моделювання, анімації, текстурування, освітлення та рендерингу. Blender також має вбудовані інструменти для створення

спеціальних ефектів, таких як частинки, симуляція фізики та інші. Також є потужним інструментом для створення анімацій та візуальних ефектів в кіноіндустрії, виробництві відеоігор, а також в інших галузях, таких як архітектура, дизайн та інженерія. Blender також може використовуватися для створення візуалізацій наукових досліджень та інших наукових проектів.

Особливостями пакету є малий розмір, висока швидкість рендерінгу, наявність версій для багатьох операційних систем — FreeBSD, GNU/Linux, Mac OS X, SGI Irix 6.5, Sun Solaris 2.8 (sparc), Microsoft Windows, SkyOS, MorphOS та Pocket PC. Пакет має такі функції, як симуляція динаміки твердих тіл (Rigid Body), рідин (Liquid simulation) та м'яких тіл (Soft body), редагування матеріалів і геометрії за принципом вузлів (Nodes), велику кількість легко доступних розширень, написаних мовою Python.

Плюси Blender:

- Вільне та відкрите програмне забезпечення, що дозволяє використовувати та змінювати програму безкоштовно.
- Інтуїтивний та доступний інтерфейс, що дозволяє швидко засвоювати програму та працювати з нею ефективно.
- Потужний набір інструментів для створення 3D-об'єктів, який включає у себе різноманітні методи моделювання, анімації, текстурування, освітлення та рендерингу.
- Вбудовані інструменти для створення спеціальних ефектів, таких як частинки, симуляція фізики та інші.
- Широкі можливості використання для створення анімацій та візуальних ефектів в кіноіндустрії, виробництві відеоігор та в інших галузях, таких як архітектура, дизайн та інженерія.

Мінуси Blender:

- Велика складність деяких інструментів, що може призвести до складнощів у роботі з програмою для початківців.
- Деякі функції можуть бути менш доступними, ніж у комерційних аналогів, особливо у галузі візуалізації та рендерингу.
- Не завжди досконала сумісність з іншими програмними засобами, що може створювати проблеми з обміном даними між програмами.
- Обмежена підтримка від виробників апаратного забезпечення, що може призвести до проблем зі сумісністю апаратного забезпечення з програмою.

Figma

Figma - це веб-сервіс та програмне забезпечення для розробки веб-дизайну та інтерфейсів користувача. Figma дозволяє дизайнерам та розробникам працювати разом у реальному часі над проектами та спільно редагувати дизайн, що зменшує час та зусилля, необхідні для розробки веб-дизайну та інтерфейсів користувача. Має простий та інтуїтивно зрозумілий інтерфейс та набір інструментів для створення векторних зображень, прототипів та інтерактивних дизайнів. Figma також дозволяє дизайнерам та розробникам створювати компоненти та бібліотеки, які можуть бути використані для швидкої розробки та збереження консистентного дизайну.

Figma дозволяє розробникам та дизайнерам створювати спільні командні облікові записи, що дозволяє зберігати та керувати проектами в одному місці. Додаткові можливості Figma включають підтримку зберігання та експорту дизайну у різних форматах, таких як PNG, SVG та PDF.

Плюси Figma:

- Віддалений доступ до програми, що дозволяє працювати з будь-якого місця та на будь-якому пристрої з Інтернет-підключенням.
- Можливість спільної роботи та редагування проектів в реальному часі, що дозволяє команді працювати над проектом одночасно та швидко вносити зміни.
- Багатофункціональний та інтуїтивний інтерфейс, що дозволяє легко створювати векторні зображення, прототипи та інтерактивні дизайни.
- Вбудовані бібліотеки компонентів та шаблонів, що дозволяють створювати швидкі та зручні дизайни з високою консистентністю.
- Легкий експорт дизайну у різних форматах, таких як PNG, SVG та PDF, що дозволяє легко ділитися дизайном з іншими.

Мінуси Figma:

- Обмежена підтримка для роботи з растровими зображеннями, що може призвести до меншої функціональності для деяких проектів.
- Відсутність деяких інструментів та функцій, які можуть бути доступні в інших програмах для розробки дизайну.
- Потребує Інтернет-підключення, що може створювати проблеми у випадку поганої Інтернет-зв'язку.
- Вартість підписки може бути високою для окремих користувачів або невеликих команд.
- Недостатня сумісність з деякими іншими програмними засобами, що може призвести до проблем з обміном даними між програмами.

1.3. Огляд існуючих додатків за даною тематикою

Path Guide - це мобільний додаток для Android, розроблений компанією Microsoft, який дозволяє користувачам створювати та зберігати маршрути та навігаційні карти для використання в закритих просторах, таких як торгові центри, музеї, аеропорти та інші громадські будівлі. Додаток використовує сенсори мобільних пристроїв, такі як акселерометр, гіроскоп та компас, для визначення розташування користувача та надає інформацію про маршрути та розташування об'єктів в закритих просторах, що дозволяє користувачам легко знаходити шлях до певних місць та об'єктів, таких як магазини, ресторани або входи.

Крім того, Path Guide дозволяє користувачам створювати власні карти та маршрути, які можна зберігати та використовувати в майбутньому, а також ділитися ними з іншими користувачами. Додаток має простий та зручний інтерфейс, що дозволяє легко користуватися ним як для навігації, так і для створення власних карт та маршрутів (Рис. 1.2).

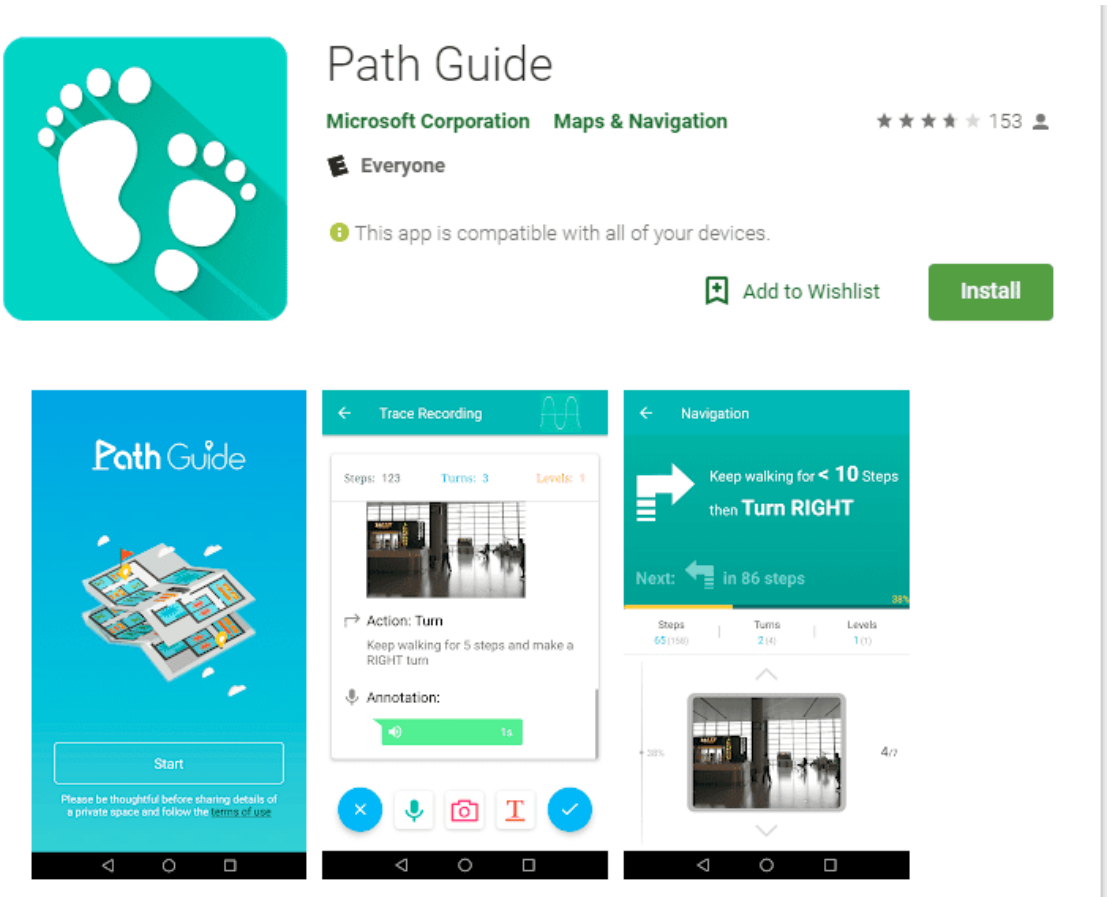


Рис. 1.2. Додаток Path Guide

Google Indoor Maps - це функція в Google Maps, яка дозволяє користувачам переглядати та навігуватися в закритих приміщеннях, таких як торгові центри, аеропорти, музеї та інші громадські будівлі. Ця функція використовує технологію

Google Street View, щоб створити віртуальні тури закритих приміщень та надати користувачам можливість переглядати та навігуватися в цих місцях. Крім того, Google Indoor Maps використовує Bluetooth, Wi-Fi та інші сенсори для визначення розташування користувача всередині будівлі та надає інформацію про розташування магазинів, ресторанів, туалетів та інших об'єктів всередині будівлі.

Використання Google Indoor Maps дозволяє користувачам швидко та легко знаходити необхідні об'єкти всередині будівлі, зменшує час, необхідний для пошуку та допомагає уникнути загублення. Крім того, Google Indoor Maps дозволяє власникам будівель додавати свої закриті приміщення до Google Maps, що може допомогти привернути нових клієнтів та покращити їх взаємодію зі своїми відвідувачами (Рис. 1.3).

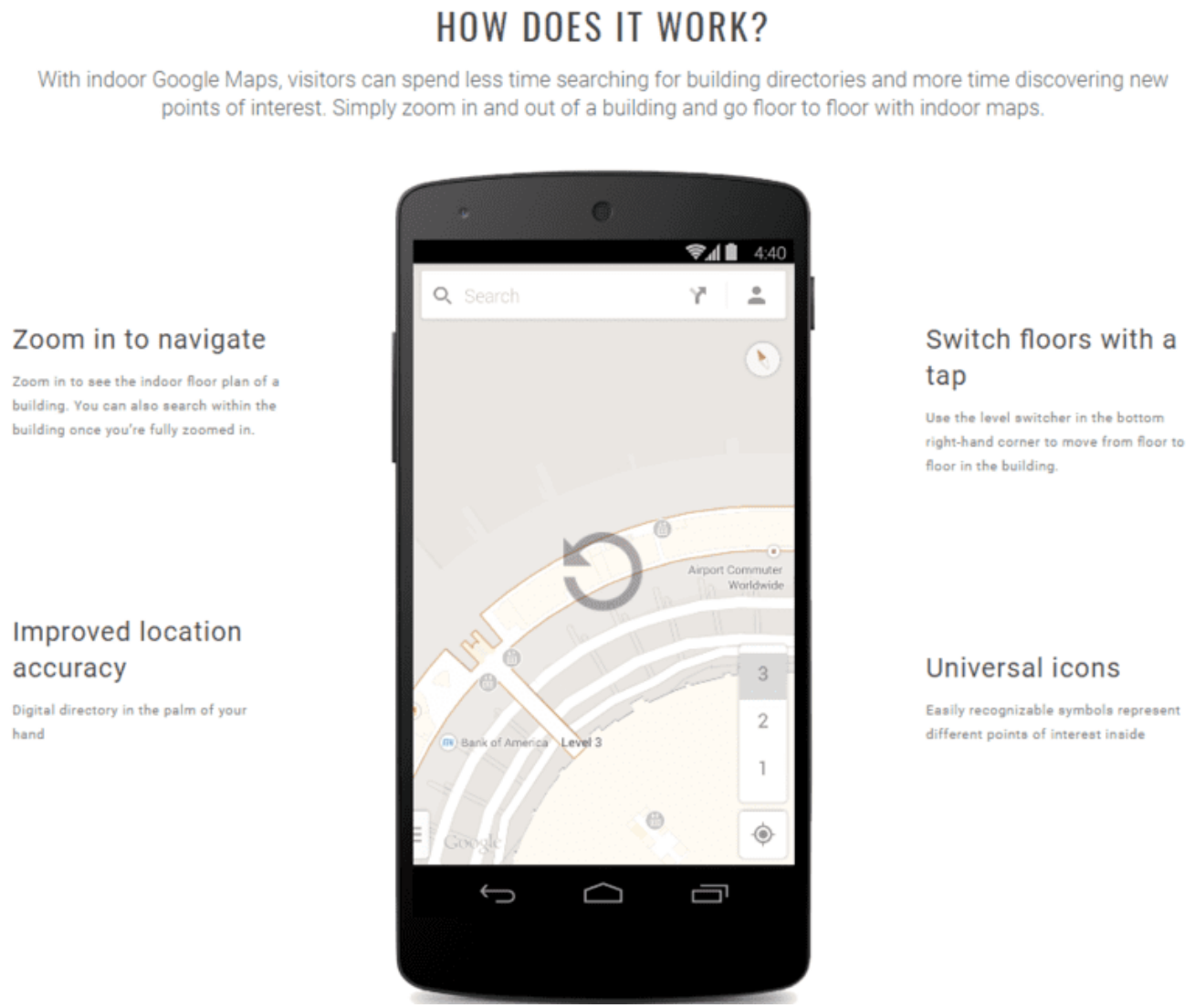


Рис. 1.3. Додаток Google Indoor Maps

Situm Mapping Tool - це програмне забезпечення для створення та керування картами в закритих приміщеннях з використанням технології внутрішнього позиціонування (Indoor Positioning System, IPS). Це програмне забезпечення дозволяє

користувачам створювати власні карти закритих приміщень, таких як торгові центри, аеропорти, музеї та інші громадські будівлі, з використанням технології IPS. Програмне забезпечення Situm Mapping Tool забезпечує точне визначення місця розташування об'єктів всередині будівлі та надає користувачам можливість створювати плани та маршрути для навігації всередині будівлі. Крім того, Situm Mapping Tool дозволяє користувачам імпортувати та експортувати картографічні дані, що дозволяє легко обмінюватися даними між різними користувачами. Програмне забезпечення також має функції для керування правами доступу користувачів та забезпечення безпеки даних. Результатом використання Situm Mapping Tool є точні та детальні карти закритих приміщень, які можуть бути використані для навігації та управління рухом всередині будівлі. Це може бути корисним для власників закритих приміщень, таких як торгові центри та аеропорти, для поліпшення досвіду відвідувачів та забезпечення їх безпеки (Рис. 1.4).

Рис. 1.4. Додаток Situm Mapping Tool

1.4. Постановка задачі

Даний AR застосунок буде корисним для студентів, що шукають аудиторії всередині академії, оскільки він дозволить користувачам відобразити маршрут до потрібної аудиторії на екрані свого смартфону, використовуючи камеру та доповнену реальність. Крім того, додаток буде містити карту академії з відміченими аудиторіями, що спростить пошук потрібної локації.

Для створення AR застосунку використовується мова програмування C# та технологія доповненої реальності. Віртуальні об'єкти будуть проектуватися у реальному світі, що дозволить користувачам бачити розташування аудиторії та маршрут до неї. Назва застосунку «Indoor-OA-DEMO» може змінитися з часом в залежності від потреб користувачів та розвитку проекту. Оскільки додаток буде розроблений для платформи Android версії 5.0 і вище, він буде доступний для широкого кола користувачів.

Основна мета проекту полягає в полегшенні пошуку аудиторій всередині академії для студентів. Використання технології доповненої реальності дозволить створити зручний та легкий у використанні додаток, який спростить навігацію студентів всередині академії.

Додаток повинен містити наступний функціонал:

- Вибір кабінету з можливістю побудови маршруту до нього.
- Можливість створення нових міток та видалення створених.
- Підтримка сканування QR-коду для знаходження місця, де знаходиться користувач.
- Швидкий пошук кабінету у списку.
- Відкриття мапи на весь екран, з можливістю гортання та зміни її розміру.
- Налаштування додатка, такі як зміна мови, теми та вимкнення звуків.
- Можливість зберігання даних.

Додаток повинен бути простим та зручним у використанні. Користувачі повинні мати можливість швидко знайти потрібний кабінет та побудувати маршрут до нього. Функція створення міток дозволить користувачам позначати важливі місця всередині будівлі, що допоможе в подальшому будувати до них маршрути. Для зручності користувачів, додаток повинен мати можливість зміни мови та теми, а також вимкнення звуків. Підтримка сканування QR-коду дозволить користувачам швидко знайти місце, де вони знаходяться. Окрім цього, додаток повинен забезпечувати безпеку та захист даних користувачів, а також мати можливість зберігати дані для подальшого використання.

Також повинні бути присутні елементи меню з кнопками, які мають наступний функціонал:

- Перемикання між вікнами.
- Очищення поля вводу.
- Розкриття списку.

Перемикання між вікнами дозволить користувачам переходити між різними екранами додатку, що спростить його використання та зробить його більш зручним для користувачів. Очищення поля вводу дозволить користувачам легко видалити введені дані, що особливо корисно при пошуку кабінету або власної мітки всередині будівлі. Розкриття списку дозволить користувачам швидко знайти необхідний кабінет, що спростить його використання та зробить його більш зручним для користувачів.

Висновок до розділу 1

У першому розділі детально описано постановку задачі та вимоги до кінцевого продукту, з яким повинні бути сумісні всі елементи мобільного додатку. Також наведений приклад мобільного додатку та середовища розробки, які були обрані для виконання проекту. Для кожного з обраних середовищ та додатків проаналізовано їх плюси та мінуси, що дозволить вибрати найбільш оптимальний варіант для розробки додатку.

У наступному розділі буде розглянуто вибір інструментів для розробки додатку. Будуть розглянуті різні варіанти програмних мов та інструментів розробки, які можуть бути використані для розробки мобільного додатку. При виборі інструментів будуть враховані вимоги до кінцевого продукту та можливості середовища розробки. Для забезпечення ефективності та якості розробки, важливо обрати найбільш оптимальний варіант інструментів та програмних мов. Вибір мови програмування та інструментів розробки повинен бути зроблений з урахуванням вимог до кінцевого продукту, можливостей розробника та середовища розробки. Крім того, важливо забезпечити безпеку та захист даних користувачів, а також забезпечити швидкість та стабільність роботи додатку. Для цього важливо правильно обрати програмні мови та інструменти розробки, які забезпечать оптимальну продуктивність та безпеку додатку.

Отже, важливо ретельно проаналізувати всі можливі варіанти програмних мов та інструментів розробки, щоб вибрати найбільш оптимальний варіант для розробки мобільного додатку, який буде зручним та безпечним для користувачів та матиме високу продуктивність.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз предметної області

Аналіз предметної області показує, що Indoor-навігація може бути корисною для відвідувачів будівель та власників бізнесу. Вона допомагає знайти шлях у великих будівлях, покращує навігацію та зменшує витрати часу на пошук предметів. Крім того, вона надає власникам бізнесу можливість надавати контент та рекламу на основі місцезнаходження.

Індор-навігація може бути використана в різних галузях, таких як аеропорти, торгові центри, музеї та інші. У аеропортах вона допомагає користувачам легко пересуватись, відстежувати багаж та інші предмети у реальному часі. У торгових центрах та підприємствах вона може бути використана для надсилання повідомлень про купони на знижки та спеціальні пропозиції, коли клієнти підходять до продуктів, що рекламуються. У музеях вона може надавати користувачам карти та аудіокерівництво, що полегшує взаємодію з відвідувачами. Однак, важливо забезпечити безпеку та конфіденційність даних користувачів, зокрема, їхнього місцезнаходження. Крім того, необхідно забезпечити якість та швидкість роботи системи Indoor-навігації, адже це безпосередньо впливає на користувачів та їхню задоволеність.

Отже, Indoor-навігація може бути дуже корисною для відвідувачів будівель та власників бізнесу, але важливо забезпечити безпеку та якість роботи системи. Для цього необхідно вибирати надійні технології та розробники повинні враховувати пріоритети користувачів та їхні потреби.

2.2. Проектування системи

Архітектура мобільних додатків - це набір методів та шаблонів, яких потрібно дотримуватися, щоб створити повністю структуровану мобільну програму.

Система керування додатком представлена у вигляді коду мовою програмування C#. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

Правильна архітектура повинна бути настільки абстрактною, щоб її можна було застосовувати до таких платформ, як iOS або Android. Однією з найважливіших особливостей адекватної архітектури є - поділ рівня відповідальності.

Важливі фактори, які слід враховувати при розробці архітектури мобільних додатків:

Визначення типу пристрою. Смартфони бувають різних категорій, і це те, про що потрібно пам’ятати, розробляючи архітектуру мобільних додатків. Тип смартфона зазвичай визначається його операційними системами, на яких вони працюють. Як ви вже знали, що смартфони Android абсолютно відрізняються від iPhone, і ці два абсолютно різні категорії. Це також є ключовими вирішальними факторами перед вибором архітектури мобільного додатка. Іншими важливими характеристиками пристрою, які слід врахувати, є:

- Розмір екрана та роздільна здатність;
- Характеристики процесора;
- Пам'ять;
- Ємність зберігання;
- Наявність інструментів розробки.

Отже, все, про що потрібно пам’ятати, це визначити тип пристрою перед тим, як вибрати архітектуру мобільних додатків. Найважливішим фактором є метод навігації додатком. Однак потреби та пріоритети клієнтів можна задовольнити, вибравши оптимальний метод навігації. Навігація мобільних додатків має великий вплив на досвід користувачів. Важливо вибрати оптимальний спосіб навігації для програми. Щоб отримати оптимальний, ви можете вибрати зі списку методів навігації:

- Одномісний вид;
- Складена панель навігації;
- Вид прокрутки;
- Модульний контролер;
- Навігація на основі жестів;
- Навігація, керована пошуком;
- Контролер вкладки.

Щоб зрозуміти вимоги користувача, необхідно розглядати різні сценарії. Заплутаний інтерфейс призводить до відмови програми. Користувачі мають мати можливість безперешкодно взаємодіяти з додатком.

Висновок до розділу 2

У другому розділі було проведено аналіз предметної області та підготовлено постановку задачі, виконано розробку архітектури ігрового додатку, визначено основні пакети сценаріїв гри.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Засоби розробки включають технічну документацію Unity для роботи з інтерфейсом і бібліотеками, які використовуються при створенні продукту, публікації на відеохостингу YouTube, а також статті в мережі Інтернеті.

Візуалізатор Unity - це потужний інструмент розробки. У Unity реалізовано велику кількість функцій для створення додатків та додавання спеціальних плагінів.

Крім того, в Unity є простий редактор інтерфейсу методом перетягування, який легко налаштовується. Його функціонал включає велику кількість різних комбінацій екранів, за допомогою яких можна виконувати тонкі налаштування всього проекту відразу всередині самого редактора. Unity підтримує такі мови програмування як C#.

Розробка кожного проекту в Unity починається з розробки незалежних сцен та окремих файлів. Також у вбудованому магазині Unity можна знайти багато різних об'єктів, анімацій, ефектів для цих анімацій, а також ще велику кількість налаштувань для проекту.

Unity дозволяє легко імпортувати моделі з різних 3D-редакторів, таких як Blender, Maya, 3ds Max тощо. Він має вбудовані інструменти для налаштування фізики, анімації та освітлення сцени. За допомогою Unity можна створювати якісні 3D-ігри, інтерактивні додатки, візуалізації та інші проекти (рис 3.1).

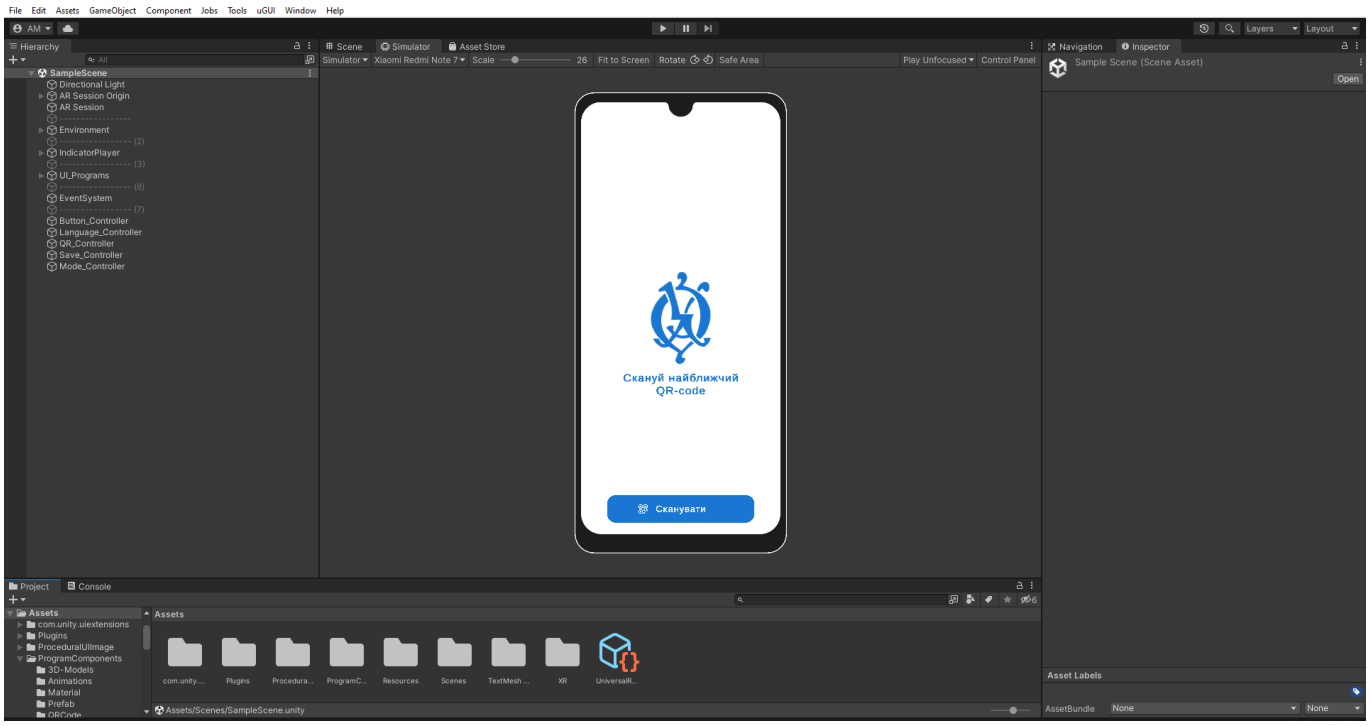


Рис. 3.1. Вікно створення фізичних можливостей

В кожній сцені міститься багато об'єктів (моделей), а порожні об'єкти – це об'єкти, що не мають жодної ігрової моделі. Кожен об'єкт складається з набору компонентів, з якими будуть взаємодіяти сценарії. Кожен такий елемент має свою назву (у Unity є можливість називати різні об'єкти схожими назвами). Вони мають свої теги та шари, на яких вони будуть відображатися. Логіка гри реалізується за допомогою написання сценаріїв для кожного елемента та додавання цих сценаріїв до потрібного об’єкта. Кожен сценарій має дві початкові функції: Start() та Update(). У них описуються основні дії та події, які будуть виконуватися з періодичним викликом. Сценарії дозволяють реалізувати взаємодію між об'єктами, фізику, штучний інтелект та іншу логіку гри. В Unity є декілька видів сценаріїв: два з них це C# and JavaScript, але всі більш серйозні ігри розробляються за допомогою C#. В C# доступні усі можливості для розробки ігрових продуктів високої якості.

3.2. Вимоги до технічного та програмного забезпечення

Додаток «Indoor-OA-DEMO» має бути написаним на мові C# з використанням ігрового рушію Unity та має працювати на платформі Android версії 5.0 і вище.

Для створення ігрового застосунку використовуються наступні засоби:

- Ігровий рушій – Unity 2020.3.6f1. Unity - це безкоштовний рушій для розробки ігор та інтерактивних додатків.
- Мова програмування – C#. C# - об'єктно-орієнтована мова програмування від Microsoft, яка використовується для розробки на .NET.
- Середовище розробки – Visual Studio Community 2019. Безкоштовна версія середовища розробки від Microsoft для .NET.
- Бібліотека моделей та плагінів – Unity Asset Store. Магазин активів Unity містить готові моделі, текстури, звуки та інструменти для розробки.
- Створення 3D-графіки – Blender 2.92. Безкоштовний редактор для створення та редагування 3D-моделей, текстур та анімації.
- Проєктування інтерфейсу – Figma. Безкоштовний редактор для створення прототипів інтерфейсу та спільної розробки.

Вимоги до технічного забезпечення пристрою для розробки:

- Процесор: Intel Core i5-3470;
- Відеокарта: NVIDIA® GeForce® GTX 660 2 ГБ чи подібні;
- Оперативна пам'ять: 8 ГБ;
- Операційна система: Windows 10/11 64-розрядна.

Вимоги до технічного забезпечення мобільного пристрою:

- Процесор: 4-ядерний;

- Оперативна пам'ять: 2 ГБ;
- Місце на диску: 30 МБ;
- Операційна система: Android 5.0.

3.3. Опис програмної реалізації

3.3.1. Прототипування додатку в Figma

Перед тим як приступити до розробки будь-якого програмного продукту, в тому числі й мобільного додатку, необхідно відпрацювати дизайн-концепцію. Один з найбільш популярних інструментів для цього - Figma. Він дозволяє створювати прототипи додатків, визначати їх функціонал та інтерфейс, та проводити тестування ще до початку розробки.

Створення прототипу - важлива стадія проектування, оскільки вона дозволяє визначити потенційні проблеми в роботі додатку та запропонувати шляхи їх вирішення. Крім того, прототип дає змогу зрозуміти, як користувачі будуть взаємодіяти з додатком, та показує, яким має бути кінцевий результат. Після створення прототипу, я зміг перейти до фази розробки, використовуючи прототип як основу для створення функціональності та дизайну додатку. Такий підхід допомагає зберегти час та зусилля, а також зменшити кількість помилок під час розробки.

Отже, створення прототипу за допомогою Figma - важлива складова успішної розробки мобільного додатку, яка дозволяє визначити концепцію та вигляд додатку, перш ніж приступати до його розробки (рис 3.2).

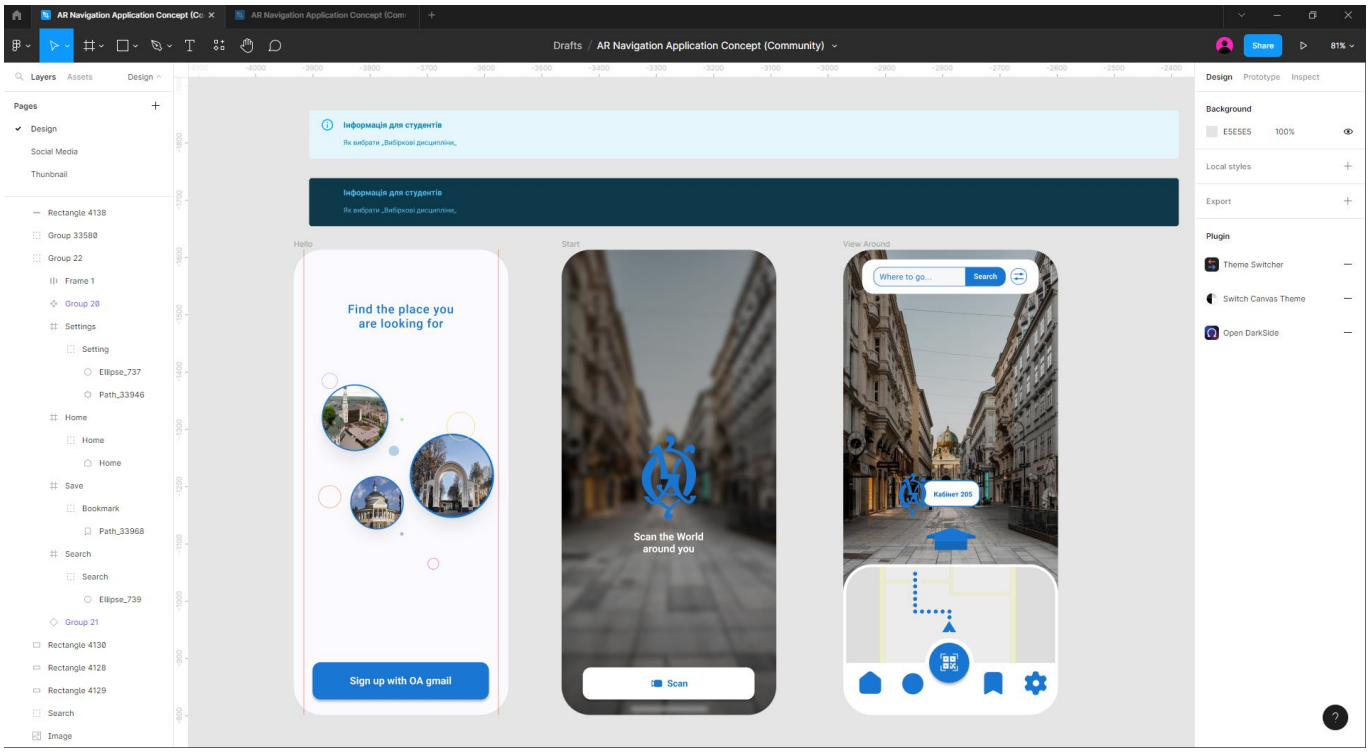


Рис. 3.2. Створення прототипу для майбутнього проекту

3.3.2. Встановлення необхідних бібліотек для роботи з додатком

Для початку роботи над додатком необхідно встановити такі пакети та плагіни з Package Manager:

- AR Foundation – це пакет для Unity, який надає можливість розробки додатків з використанням доповненої реальності (AR) на різних платформах, таких як iOS та Android. Цей пакет забезпечує спільний набір інструментів та функцій для створення AR додатків, що працюють на різних платформах. AR Foundation дозволяє розробникам легко створювати AR додатки, які можуть використовувати різні технології AR, такі як ARKit, ARCore, HoloLens та інші.
- ARCore XR Plugin – це плагін для Unity, який дозволяє створювати додатки з використанням доповненої реальності (AR) на платформі Android з використанням ARCore - фреймворку для розробки AR додатків від Google. ARCore XR Plugin надає розробникам доступ до ARCore API у Unity, що дозволяє легко використовувати функції ARCore для створення додатків з AR. З його допомогою можна відстежувати рухи пристрою, розпізнавати поверхні та об'єкти в реальному світі, що дозволяє розробникам створювати різноманітні AR додатки, від ігор до додатків для візуалізації та навчання. ARCore XR Plugin підтримує такі функції, як фіксація об'єктів на реальній поверхні, відстеження рухів камери та інші функції, необхідні для створення AR додатків на платформі Android.
- Universal RP – це пакет для Unity, який дозволяє розробникам створювати високоякісні графічні ефекти та візуальні елементи у своїх проектах. Universal RP є наступником старішої версії Render Pipeline, що дозволяє забезпечити кращу продуктивність та підтримку більшої кількості платформ.
- XR Plugin Management – це система управління плагінами (пакетами) для віртуальної та доповненої реальності (VR/AR) в Unity. Ця система дозволяє розробникам легко додавати та використовувати різні плагіни для різних платформ та пристроїв, що підтримують VR/AR. XR Plugin Management забезпечує спільний інтерфейс для роботи з різними плагінами, що дозволяє легко переключатися між ними та налаштовувати їх параметри.
- ZXing – це вільний та відкритий код бібліотеки для розпізнавання штрих-кодів та QR-кодів. Ця бібліотека може бути використана в різних програмах та додатках, що потребують розпізнавання штрих-кодів та QR-кодів. ZXing може розпізнавати різні типи штрих-кодів, включаючи Code 39, Code 128, EAN-8, EAN-13, UPC-A, UPC-E та інші. Вона також підтримує розпізнавання QR-кодів, що дозволяє використовувати їх для швидкого доступу до інформації в мобільних додатках та веб-сайтах. ZXing є популярним інструментом у багатьох програмах та додатках для розпізнавання штрих-кодів та QR-кодів.

- DOTween – це додатковий пакет для Unity, який дозволяє легко анімувати об'єкти та властивості в грі. Цей пакет надає більш розширені можливості анімації, ніж стандартні інструменти Unity, дозволяючи розробникам створювати більш реалістичні та складні анімації з меншими зусиллями.
- ProceduralUIImage – це додатковий пакет для Unity, який дозволяє створювати текстури та зображення програмно, без використання зовнішніх файлів. Цей пакет дозволяє розробникам створювати складні та цікаві текстури з використанням коду, що може бути використаний для різних ефектів та стилів. ProceduralUIImage надає можливість створювати текстури з різними параметрами, такими як розмір, форма, кольори та шаблони. Цей пакет також підтримує створення текстур з використанням шуму та інших математичних функцій, що дозволяє створювати більш складні та реалістичні текстури.

3.3.3. Робота з мапою

Для створення точної та реалістичної 3D мапи будинку було використано потужний редактор Blender. При розробці мапи використовувався фактичний план будівлі, що дозволило забезпечити точність та відтворення всіх деталей.

Головне завдання при створенні мапи полягало в тому, щоб з невеликими похибками повторити розміри будівлі та забезпечити співвідношення між AR мапою та реальним будинком. Точність мапи є важливим фактором для забезпечення точної та ефективної навігації (рис 3.3).

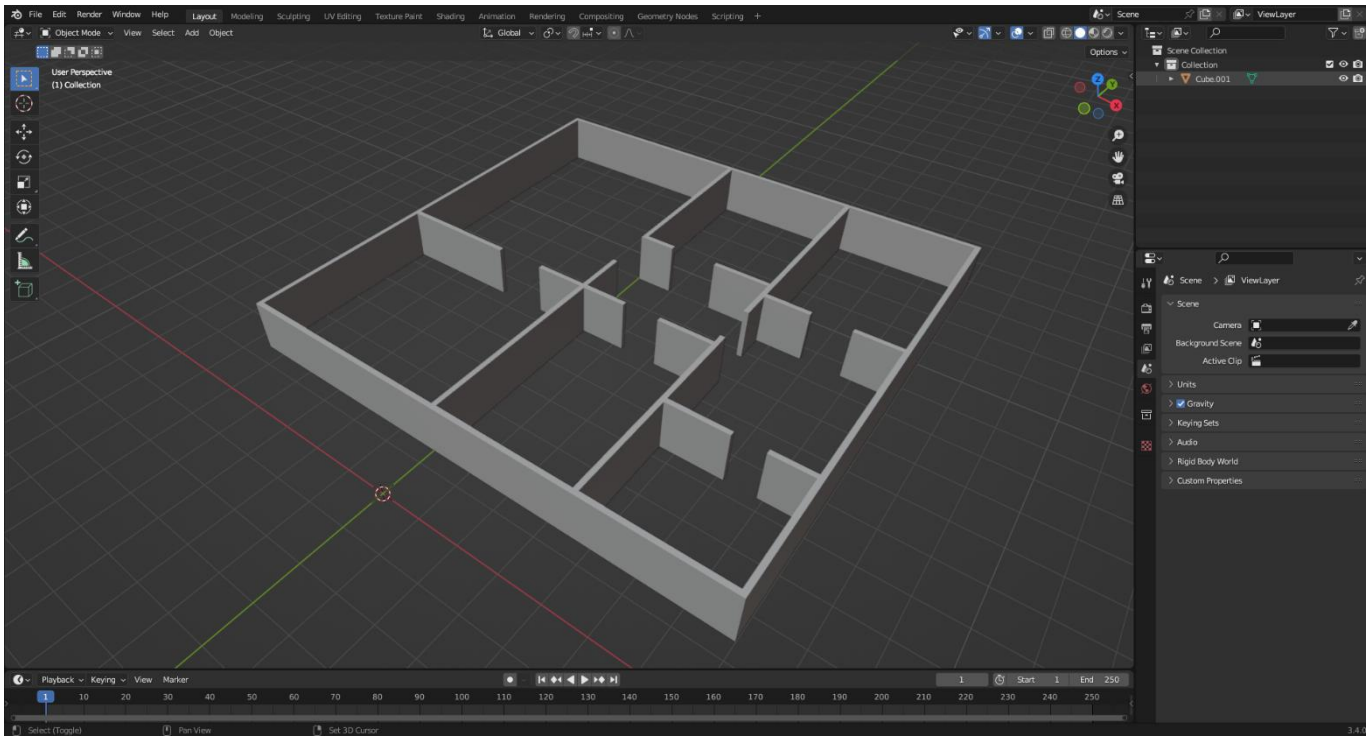


Рис. 3.3. Створення мапи для майбутнього проекту

Після створення 3D мапи будинку, наступним кроком є її інтеграція в ігровий рушій Unity. Для забезпечення ефективної навігації користувачів у просторі, використовується функція AI Navigation.

У випадку з AR додатком, AI Navigation дозволяє користувачам точно та ефективно переміщатися по простору будинку, ігноруючи перешкоди та використовуючи оптимальний шлях. Це забезпечує точність та ефективність навігації, що є критичними факторами для успішної розробки додатків з AR навігацією (рис 3.4).

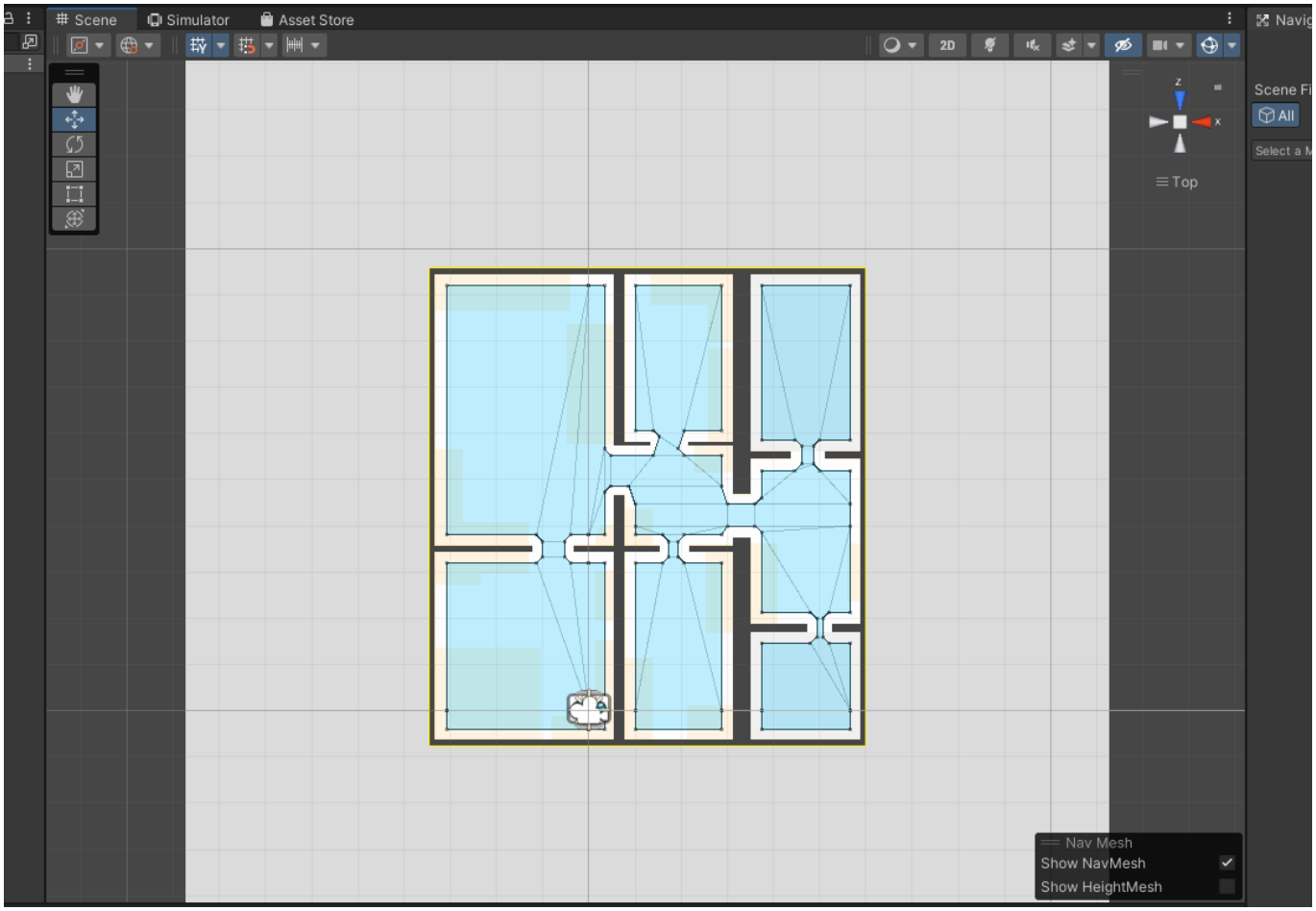


Рис. 3.4. Реалізація пошуку шляхів за допомогою NavMesh

3.3.4. Робота з основними компонентами додатку

Після створення 3D мапи будинку та використання функції AI Navigation, наступним кроком було створення індикатора, який допомагатиме користувачам зорієнтуватися в просторі та повторюватиме їх безпосереднє переміщення. Для цього була створена модель, яка була прив'язана до AR камери та рухалася разом з користувачем. Ця модель слугувала індикатором переміщення, що дозволяло користувачам більш точно зорієнтуватися в просторі та визначати своє місцезнаходження.

Індикатор містить наступні компоненти:

Position Constraint – це функціонал, який дозволяє обмежувати рух об'єкта в 3D просторі залежно від руху іншого об'єкта. В ігровому рушії Unity, Position Constraint може бути використаний для створення різноманітних ефектів та механік, таких як замокання об'єктів до інших об'єктів, створення систем руху та багато іншого.

Rotation Constraint – це функціонал, який дозволяє обмежувати обертання об'єкта в 3D просторі залежно від руху іншого об'єкта. В ігровому рушії Unity, Rotation Constraint може бути використаний для створення різноманітних ефектів та механік, таких як обмеження обертання камери, замокання об'єктів до інших об'єктів та багато іншого.

Line Renderer – це компонент в ігровому рушії Unity, який дозволяє створювати лінії та криві в 3D просторі. Line Renderer в моєму випадку був використаний для створення різноманітних ефектів та механік, таких як побудова маршруту до кінцевої точки.

Audio Source – це компонент в ігровому рушії Unity, який дозволяє відтворювати звукові ефекти в іграх та додатках. Audio Source я використовував для відтворення звукового сповіщення коли до кінечного маркеру залишається менше двох метрів.

Скрипт SetNavigationTarget – Даний скрипт відповідає за навігацію гравця в ігровому рушії Unity. Він містить компоненти NavMeshPath, LineRenderer та SearchableDropDown. Основні елементи цього скрипту наступні:

- *List<Target> navigationTargetObjects:* список об'єктів, які можуть бути використані як цілі навігації.
- *NavMeshPath path:* змінна, яка представляє шлях, розрахований за допомогою NavMesh.
- *LineRenderer line:* змінна, яка представляє компонент Line Renderer, що використовується для відображення шляху навігації.
- *SearchableDropDown navigationTargetDropDown:* випадаючий список, що містить список цілей навігації.
- *Vector3 targetPosition:* позиція поточної цілі навігації.
- *bool isSoundPlayed:* змінна, яка відстежує, чи відтворювався звуковий ефект для поточної цілі навігації.

Метод *Start()* ініціалізує змінні *path, line* та *navigationTargetDropDown*. Метод *Update()* викликається кожен кадр та обраховує шлях навігації, відображає його за

допомогою *Line Renderer* та відтворює звуковий ефект, якщо гравець наближається до цілі навігації. Метод *OnConnectedToServer(string selectedText)* викликається, коли гравець обирає нову ціль навігації зі списку випадаючого списку. В цьому методі об'єкти цілей навігації активуються або деактивуються в залежності від того, яку ціль навігації вибрав гравець. Крім того, метод оновлює змінні *targetPosition* та *isSoundPlayed*.

Отже, цей скрипт дозволяє гравцеві вибирати цілі навігації зі списку випадаючого списку та навігувати до них, відображаючи шлях навігації за допомогою *Line Renderer* та відтворюючи звуковий ефект, коли гравець наближається до цілі навігації (рис 3.5).

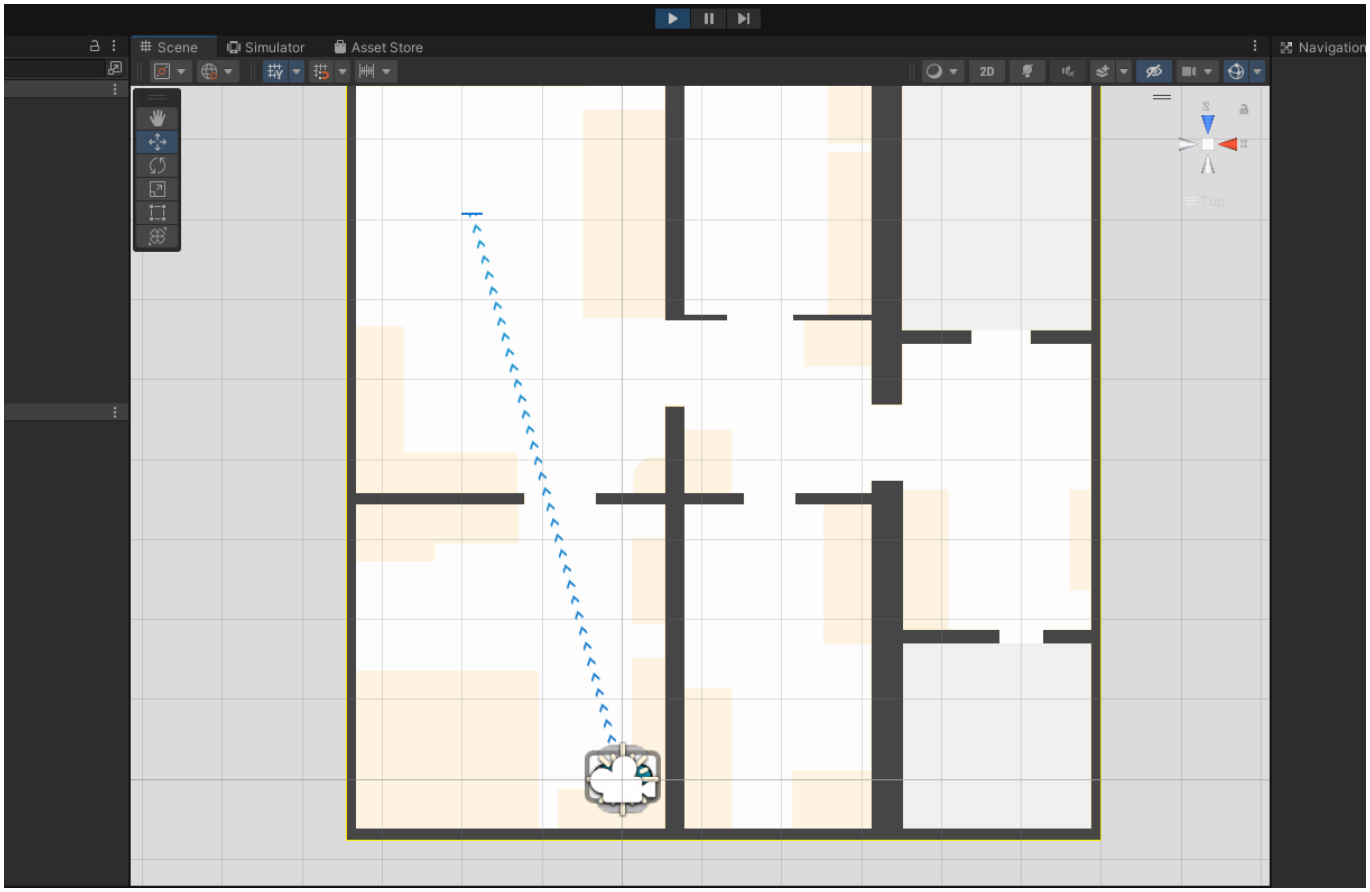


Рис. 3.5. Робота скрипту SetNavigationTarget

Лістинг коду 3.1. SetNavigationTarget

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class SetNavigationTarget : MonoBehaviour
{
    [SerializeField]
    private SearchableDropDown navigationTargetDropDown;
    [SerializeField]
    public List<Target> navigationTargetObjects = new List<Target>();
    private NavMeshPath path;
```

```
    private LineRenderer line;
    private Vector3 targetPosition = Vector3.zero;
    private bool isSoundPlayed = false;
    private void Start()
    {
        path = new NavMeshPath();
        line = transform.GetComponent<LineRenderer>();
        navigationTargetDropDown.OnValueChangedEvt += OnConnectedToServer;
    }

    private void Update()
    {
        if (targetPosition != Vector3.zero)
        {
            NavMesh.CalculatePath(transform.position, targetPosition, NavMesh.AllAreas,
path);
            line.positionCount = path.corners.Length;
            line.SetPositions(path.corners);

            if (!isSoundPlayed && Vector3.Distance(transform.position, targetPosition) <=
2f)
            {
                AudioSource audioSource = GetComponent<AudioSource>();
                audioSource.Play();
                isSoundPlayed = true;
            }
        }
    }

    public void OnConnectedToServer(string selectedText)
    {
        targetPosition = Vector3.zero;
        isSoundPlayed = false;

        foreach (var target in navigationTargetObjects)
        {
            if (target.Name.Equals(selectedText))
            {
                target.SetActive(true);
                targetPosition = target.PositionObject.transform.position;
            }
            else
            {
                target.SetActive(false);
            }
        }

        if (targetPosition == Vector3.zero)
        {
            line.positionCount = 0;
            line.SetPositions(new Vector3[0]);
        }
    }
}
```

Примітка. Саме цей код реалізує побудову маршруту від індикатора до кінечної мітки.

Були встановлені мітки на кімнати, які стали кінцевими точками маршруту, який вказувала навігаційна лінія. Ці мітки були створені у вигляді логотипу Острозької академії за допомогою програмного забезпечення Blender та мали анімацію похитування. Це дозволило створити чіткий та привабливий візуальний

елемент для кожної кінцевої точки маршруту, який допоміг користувачам зорієнтуватися в просторі та знайти потрібну кімнату без зайвих зусиль. Крім того, використання Blender дозволило створити мітки високої якості, що додало професіоналізму та естетичної привабливості до проекту (рис 3.6).

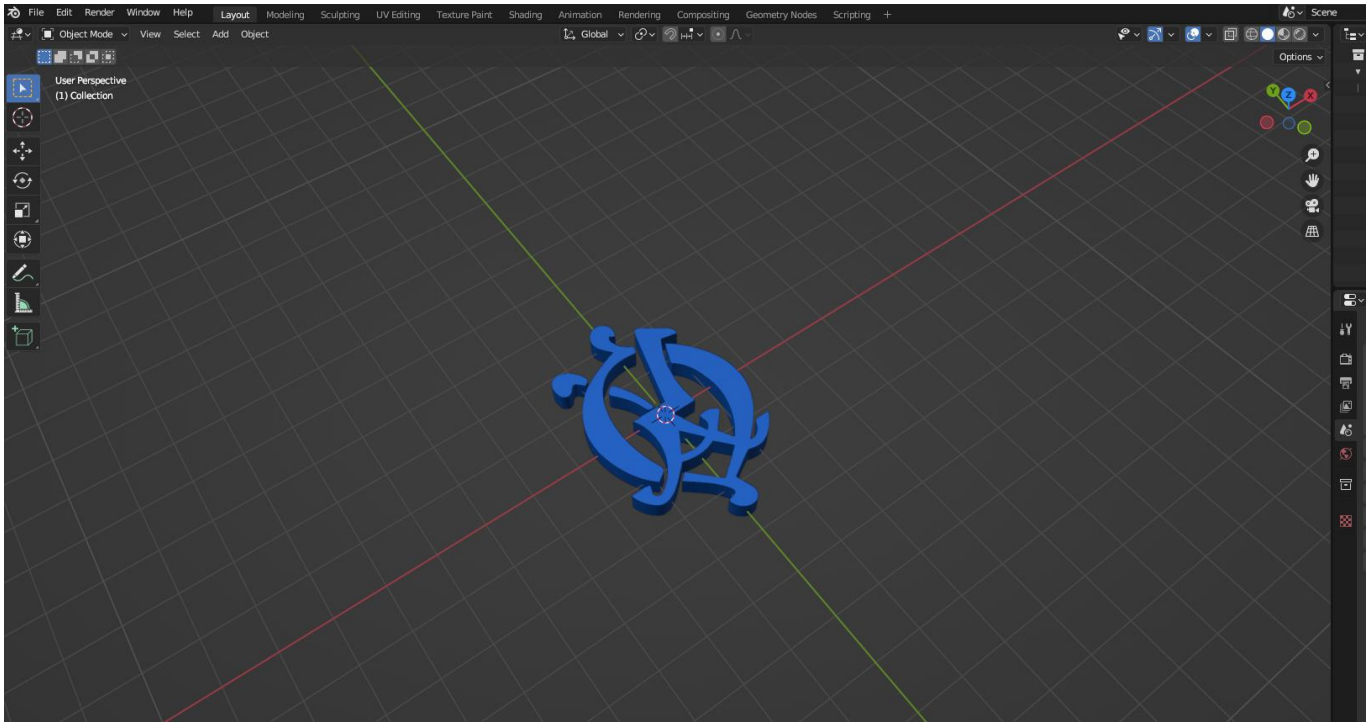


Рис. 3.6. 3D модель логотипу академії

У процесі розробки системи навігації було вирішено додати таргети з QR-кодами, щоб забезпечити користувачам зручний та швидкий спосіб визначити своє місцезнаходження в будівлі. При скануванні QR-коду з допомогою мобільного додатку, користувач міг отримати точну інформацію про своє місцезнаходження в будівлі та відобразити його на карті. Крім того, цей елемент системи навігації додав додатковий рівень зручності, ефективності для користувачів та дозволило забезпечити точність та швидкість визначення місцезнаходження користувача в приміщенні (рис 3.7).

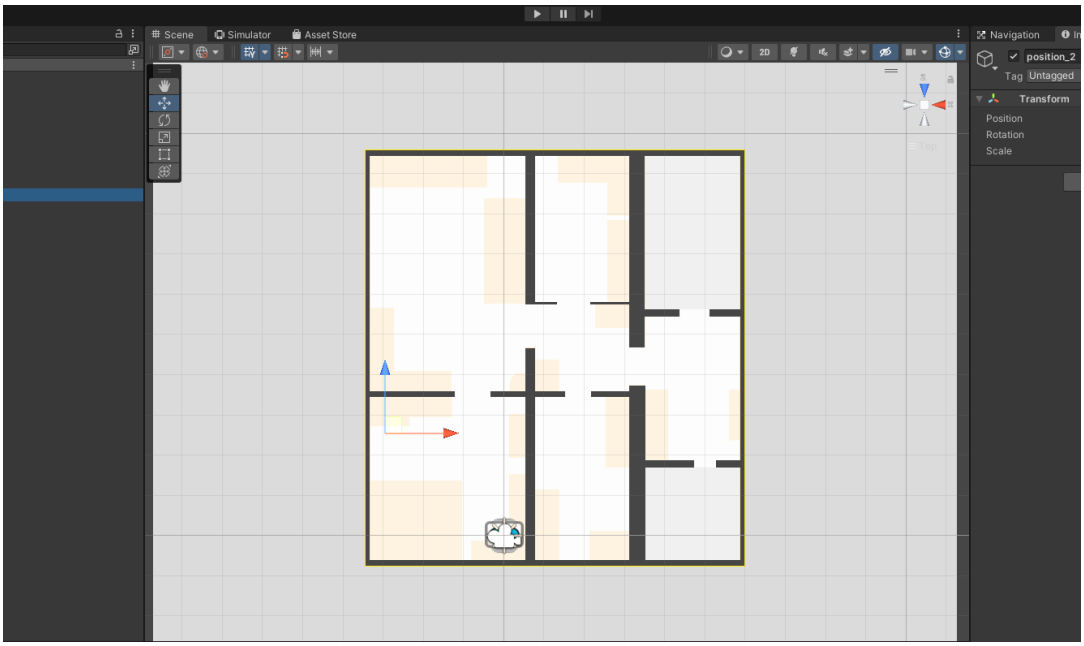


Рис. 3.7. Розміщення таргету QR-кода

3.3.5. Інтерфейс та органи керування додатком

Після налаштування мапи, було створено інтерфейс навігаційного додатку, та додано функціонал, щоб забезпечити його коректну роботу. На початковому етапі розробки інтерфейсу навігаційного додатку було створено домашню сторінку, яка містила випадаючий список з можливістю пошуку потрібного кабінету, функціональну міні-мапу, кнопку додавання нової мітки та навігаційні кнопки (рис 3.8).

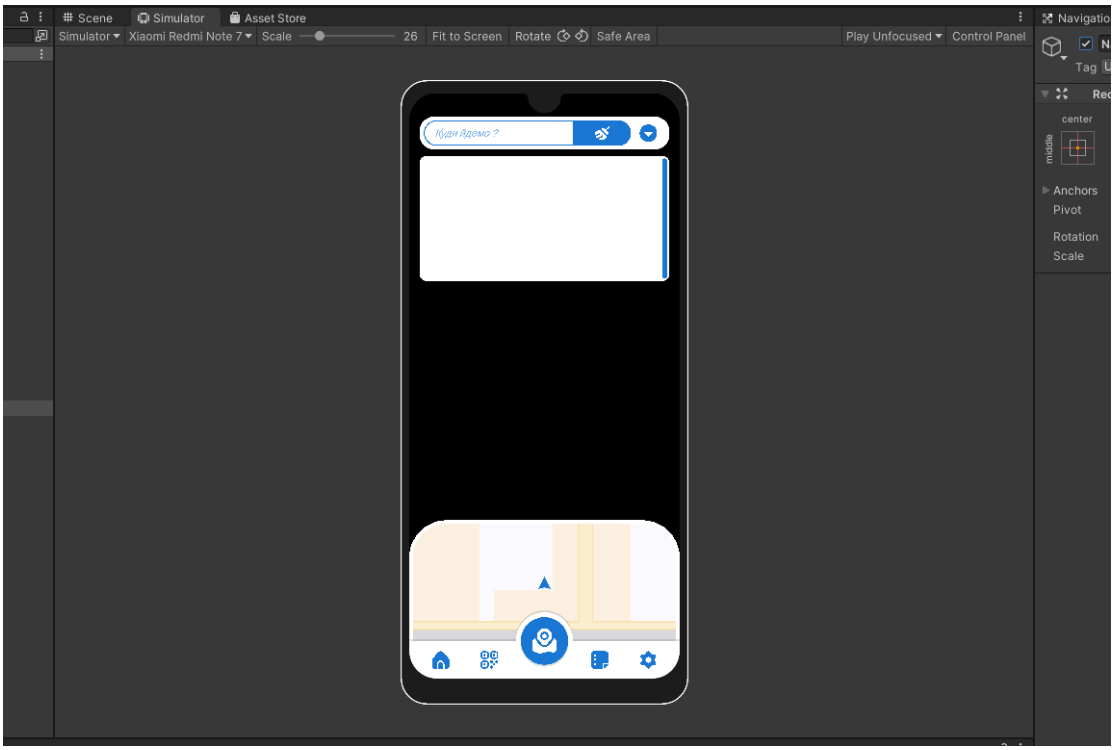


Рис. 3.8. Домашня сторінка додатку

Що ж розпочнемо з випадаючого списку, це повністю кастомний UI елемент, який було розроблено з метою покращення його функціональності, адже стандартний елемент не має можливості пошуку та має обмежену функціональність. Для того, щоб реалізувати задуману функціональність, було створено скрипт, що включає в себе методи для додавання елементів до розкривного списку, фільтрації елементів за введеним користувачем запитом, відображення списку, вибору елемента зі списку, очищення значення розкривного списку та зміни розміру списку в залежності від кількості елементів. Окрім цього, у скрипті було визначено публічні змінні: blockerButton (кнопка блокування), buttonsPrefab (префаб кнопок), maxScrollRectSize (максимальний розмір списку) та avlOptions (список доступних опцій). Було також реалізовано різні події, такі як OnValueChangedEvt, onValueChanged, onEndEdit та onClick, які спрацьовують при зміні значення введеного користувачем запиту, вибору елемента зі списку, введення користувачем нового значення та натискання на кнопки. Основні методи скрипту включають Init(), який ініціалізує розкривний список та додає до нього елементи; OnEndEditing(), який перевіряє, чи введене користувачем значення є допустимим; FilterDropdown(), який фільтрує елементи списку за введеним користувачем запитом; OnItemSelected(), який встановлює вибраний елемент; OnDDButtonClick(), який відображає або приховує список; та SetScrollActive(), який встановлює статус відображення списку.

Отже, випадаючий список є функціональним та має можливість пошуку, завдяки розробленому скрипту. Це дозволяє зручно та швидко вибирати необхідні елементи зі списку та забезпечує більш зручну роботу з додатком (рис 3.9).

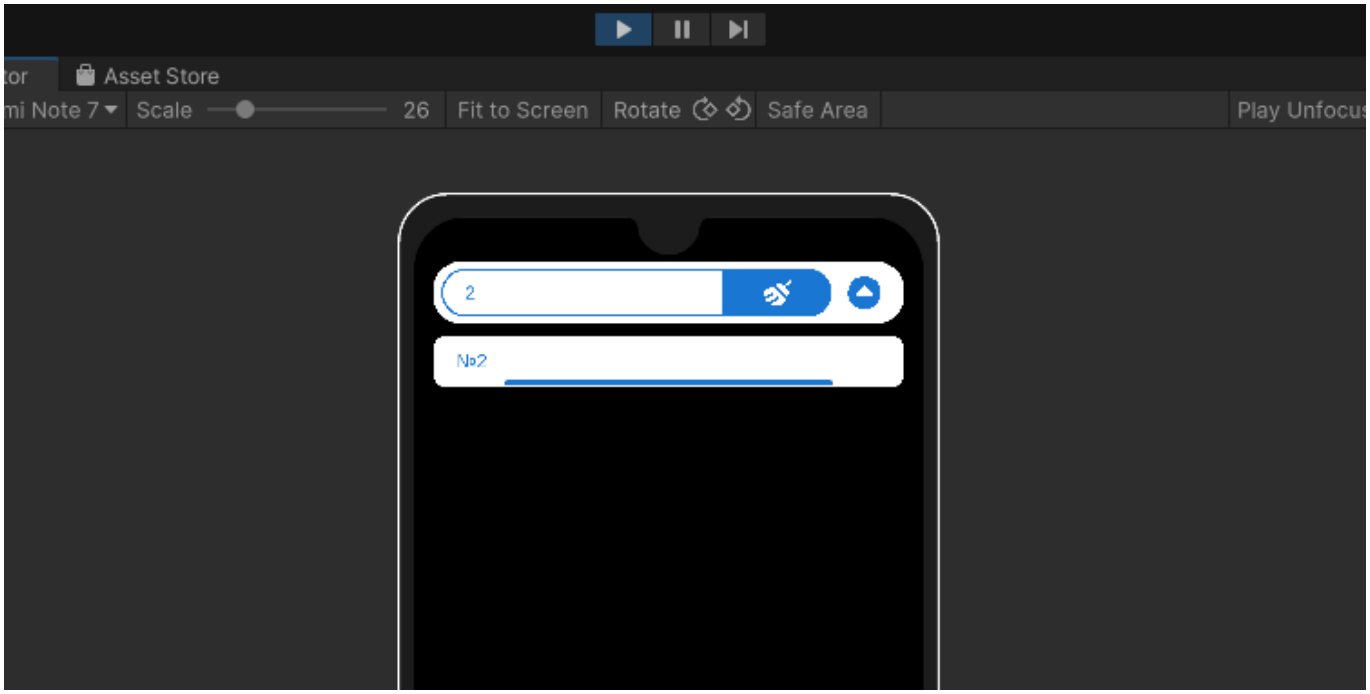


Рис. 3.9. Демонстрація роботи пошуку у випадаючому списку

Лістинг коду 3.2. SearchableDropDown

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using Button = UnityEngine.UI.Button;
using UnityEngine.EventSystems;


public class SearchableDropDown : MonoBehaviour
{
    [SerializeField] private Button blockerButton;
    [SerializeField] private GameObject buttonsPrefab = null;
    [SerializeField] private int maxScrollRectSize = 180;
    [SerializeField] public List<string> avlOptions = new List<string>();
    [SerializeField] private Button clearButton;



    private Button ddButton = null;
    private TMP_InputField inputField = null;
    private ScrollRect scrollRect = null;
    private Transform content = null;
    private RectTransform scrollRectTrans;
    private bool isContentHidden = true;
    private List<Button> initializedButtons = new List<Button>();

    public delegate void OnValueChangedDel(string val);
    public OnValueChangedDel OnValueChangedEvt;

    void Start()
    {
        Init();
    }

    void Update()
    {
        UpdateIfNewOptionAdded();
        ResizeScrollRect();
    }

    private void Init()
    {
        ddButton = this.GetComponentInChildren<Button>();
        scrollRect = this.GetComponentInChildren<ScrollRect>();
        inputField = this.GetComponentInChildren<TMP_InputField>();
        scrollRectTrans = scrollRect.GetComponent<RectTransform>();
        content = scrollRect.content;

        blockerButton.GetComponent<RectTransform>().sizeDelta = new Vector2(Screen.width,
Screen.height);
        blockerButton.gameObject.SetActive(false);
        blockerButton.transform.SetParent(this.GetComponentInParent<Canvas>().transform);

        blockerButton.onClick.AddListener(OnBlockerButtClick);
        ddButton.onClick.AddListener(OnDDButtonClick);
        scrollRect.onValueChanged.AddListener(OnScrollRectvalueChange);
        inputField.onValueChanged.AddListener(OnInputvalueChange);
        inputField.onEndEdit.AddListener(OnEndEditing);
```

```
        AddItemToScrollRect(avlOptions);

        clearButton.onClick.AddListener(ResetDropDown);

    }

    public string GetValue()
    {
        return inputField.text;
    }

    public void ResetDropDown()
    {
        inputField.text = string.Empty;
        OnValueChangedEvt?.Invoke(string.Empty);

    }
    public void AddItemToScrollRect(List<string> options)
    {
        foreach (var option in options)
        {
            var buttObj = Instantiate(buttonsPrefab, content);
            buttObj.GetComponentInChildren<TMP_Text>().text = option;
            buttObj.name = option;
            buttObj.SetActive(true);
            var butt = buttObj.GetComponent<Button>();
            butt.onClick.AddListener(delegate { OnItemSelected(buttObj); });
            initializedButtons.Add(butt);
        }
        ResizeScrollRect();
        scrollRect.gameObject.SetActive(false);
    }
    private void OnEndEditing(string arg)
    {
        if (string.IsNullOrEmpty(arg))
        {
            Debug.Log("no value entered ");
            return;
        }
        StartCoroutine(CheckIfValidInput(arg));
    }

    IEnumerator CheckIfValidInput(string arg)
    {
        yield return new WaitForSeconds(0.1f);
        if (!avlOptions.Any(option => option.Contains(arg)))

        {

            inputField.text = String.Empty;
        }
        OnValueChangedEvt?.Invoke(inputField.text);
    }

    private void ResizeScrollRect()
    {
        LayoutRebuilder.ForceRebuildLayoutImmediate((RectTransform)content.transform);
        var length = content.GetComponent<RectTransform>().sizeDelta.y;

        scrollRectTrans.sizeDelta = length > maxScrollRectSize ? new
Vector2(scrollRectTrans.sizeDelta.x,
            maxScrollRectSize) : new Vector2(scrollRectTrans.sizeDelta.x, length + 5);
    }

    private void OnInputvalueChange(string arg0)
```

```
    {
        if (!avlOptions.Contains(arg0))
        {
            FilterDropdown(arg0);
        }
    }
    public void FilterDropdown(string input)
    {
        if (string.IsNullOrEmpty(input))
        {
            foreach (var button in initializedButtons)
                button.gameObject.SetActive(true);
            ResizeScrollRect();
            scrollRect.gameObject.SetActive(false);
            return;
        }

        var count = 0;
        foreach (var button in initializedButtons)
        {
            if (!button.name.ToLower().Contains(input.ToLower()))
            {
                button.gameObject.SetActive(false);
            }
            else
            {
                button.gameObject.SetActive(true);
                count++;
            }
        }

        SetScrollActive(count > 0);
        ResizeScrollRect();
    }
    private void OnScrollRectvalueChange(Vector2 arg0)
    {

    }

    private void OnItemSelected(GameObject obj)
    {
        inputField.text = obj.name;
        foreach (var button in initializedButtons)
            button.gameObject.SetActive(true);
        isContentHidden = false;
        OnDDButtonClick();
        StopAllCoroutines();
        StartCoroutine(CheckIfValidInput(obj.name));
    }
    private void OnDDButtonClick()
    {
        if(GetActiveButtons()<=0)
            return;
        ResizeScrollRect();
        SetScrollActive(isContentHidden);
    }
    private void OnBlockerButtClick()
    {
        SetScrollActive(false);
    }
    private void SetScrollActive(bool status)
    {
        scrollRect.gameObject.SetActive(status);
        blockerButton.gameObject.SetActive(status);
        isContentHidden = !status;
```

```
        ddButton.transform.localScale = status ? new Vector3(1, -1, 1) : new Vector3(1,
1, 1);
    }
    private float GetActiveButtons()
    {
        var count = content.transform.Cast<Transform>().Count(child =>
child.gameObject.activeSelf);
        var length = buttonsPrefab.GetComponent<RectTransform>().sizeDelta.y * count;
        return length;
    }

    private void UpdateIfNewOptionAdded()
    {
        int currentCount = initializedButtons.Count;
        int optionsCount = avlOptions.Count;

        if (currentCount < optionsCount)
        {
            List<string> newOptions = avlOptions.GetRange(currentCount, optionsCount -
currentCount);
            AddItemToScrollRect(newOptions);
        }
    }
}
```

Примітка. Саме цей код реалізує весь функціонал випадаючого списку.

За допомогою Renderer Texture була створена міні мапа, яка відображає поточне місце знаходження гравця та дозволяє відслідкувати його рух на карті. Крім цього, міні-мапу можна відкривати на весь екран, переміщати, зумувати та подвійним кліком повертати у початкове положення.

У скрипті CameraMovement використовується змінна cam, яка вказує на об'єкт Camera, який необхідно переміщувати та зумувати. Ця змінна має атрибут [SerializeField], що дозволяє встановлювати значення для неї з інспектора Unity. Крім того, скрипт містить змінні zoomStep, minCamSize та maxCamSize, які відповідають за крок зумування камери, мінімальний та максимальний розмір камери відповідно. Метод PanCamera() відповідає за переміщення камери. Якщо targetTexture камери не встановлено, то при натисканні на ліву кнопку миші виконується переміщення камери. При зажатій кнопці миші камера переміщується в напрямку руху миші. Методи ZoomIn() та ZoomOut() відповідають за зумування камери. Вони викликаються при натисканні відповідних кнопок на інтерфейсі Unity-проекту та зменшують або збільшують розмір камери на величину zoomStep, але не дозволяють камері вийти за межі minCamSize та maxCamSize. У методі PanCamera() також реалізовано можливість швидкого переміщення камери до попередньо визначеного об'єкту з тегом "Arrow". Якщо користувач двічі клікає на ліву кнопку миші з інтервалом менше catchTime, то виконується пошук об'єкту з тегом "Arrow". Якщо об'єкт знайдено, камера переміщується до цього об'єкту. Якщо об'єкт не знайдено, то виводиться повідомлення про помилку (рис 3.10).

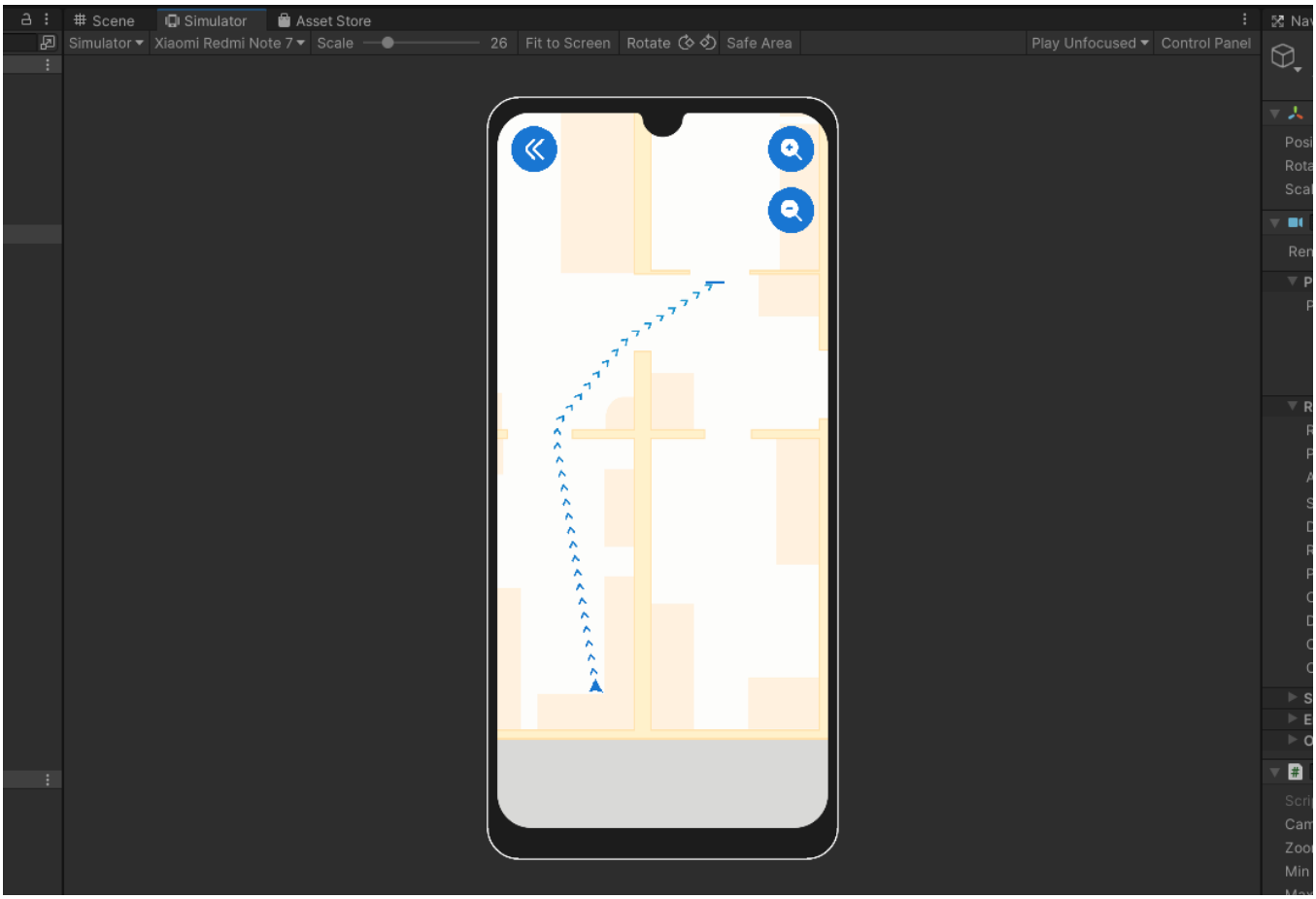


Рис. 3.10. Демонстрація розкриття міні-мапи на весь екран

Лістинг коду 3.3. CameraMovement

```
using UnityEngine;

public class CameraMovement : MonoBehaviour
{
    [SerializeField]
    private Camera cam;
    [SerializeField]
    private float zoomStep, minCamSize, maxCamSize;
    private Vector3 dragOrigin;
    private float lastClickTime;
    private float catchTime = 0.25f;

    private void Update()
    {
        PanCamera();
    }

    private void PanCamera()
    {
        // Перевірка, чи targetTexture не встановлено
        if (cam.targetTexture == null)
        {
            if (Input.GetMouseButtonDown(0))
            {
                if (Time.time - lastClickTime < catchTime)
                {
                    // Знайти об'єкт з тегом "Camni"
```

```
                    GameObject camniObject = GameObject.FindWithTag("Arrow");

                    if (camniObject != null)
                    {
                        // Встановити координати камери відносно об'єкта "Camni"
                        cam.transform.position = new
Vector3(camniObject.transform.position.x, cam.transform.position.y,
camniObject.transform.position.z);
                        Debug.Log("Координати камери встановлено на " +
cam.transform.position.ToString() + " відносно об'єкта з тегом 'Arrow'");
                    }
                    else
                    {
                        Debug.LogWarning("Об'єкт з тегом 'Camni' не знайдено!");
                    }
                }
                lastClickTime = Time.time;
            }

            if (Input.GetMouseButtonDown(0))
                dragOrigin = cam.ScreenToWorldPoint(Input.mousePosition);

            if (Input.GetMouseButton(0))
            {
                Vector3 difference = dragOrigin -
cam.ScreenToWorldPoint(Input.mousePosition);
                print("origin" + dragOrigin + "newPosition" +
cam.ScreenToWorldPoint(Input.mousePosition) + "difference" + difference);
                cam.transform.position += difference;
            }
        }
    }

    public void ZoomIn()
    {
        float newSize = cam.orthographicSize - zoomStep;
        cam.orthographicSize = Mathf.Clamp(newSize, minCamSize, maxCamSize);
    }

    public void ZoomOut()
    {
        float newSize = cam.orthographicSize + zoomStep;
        cam.orthographicSize = Mathf.Clamp(newSize, minCamSize, maxCamSize);
    }
}
```

Примітка. Саме цей код реалізує основну роботу міні-мапи.

Для того, щоб забезпечити користувачу комфортне користування додатком, важливо не дозволяти йому переміщувати камеру за межі встановленої області. Для цього я використовував скрипт CameraBounds, який дозволяє обмежити рух камери в межах певної області.

У скрипті використовуються змінні cam, яка вказує на камеру, що необхідно обмежити, та xBounds, zBounds, які вказують межі для руху камери по осях X та Z відповідно. Значення цих змінних можна встановити через інспектор Unity-редактора завдяки атрибуту [SerializeField]. У методі OnDrawGizmos() використовуються методи класу Gizmos, щоб намалювати на сцені прямокутник з вказаними межами xBounds та zBounds. Це дозволяє візуально показати зону, в межах якої дозволено

переміщувати камеру. У методі LateUpdate() виконується обмеження руху камери в межах вказаних значень. Змінна clampedPosition містить поточну позицію камери, яку необхідно обмежити. За допомогою методу Mathf.Clamp() обмежується позиція камери по осях X та Z. Результат обмеження записується назад у позицію камери (рис 3.11).

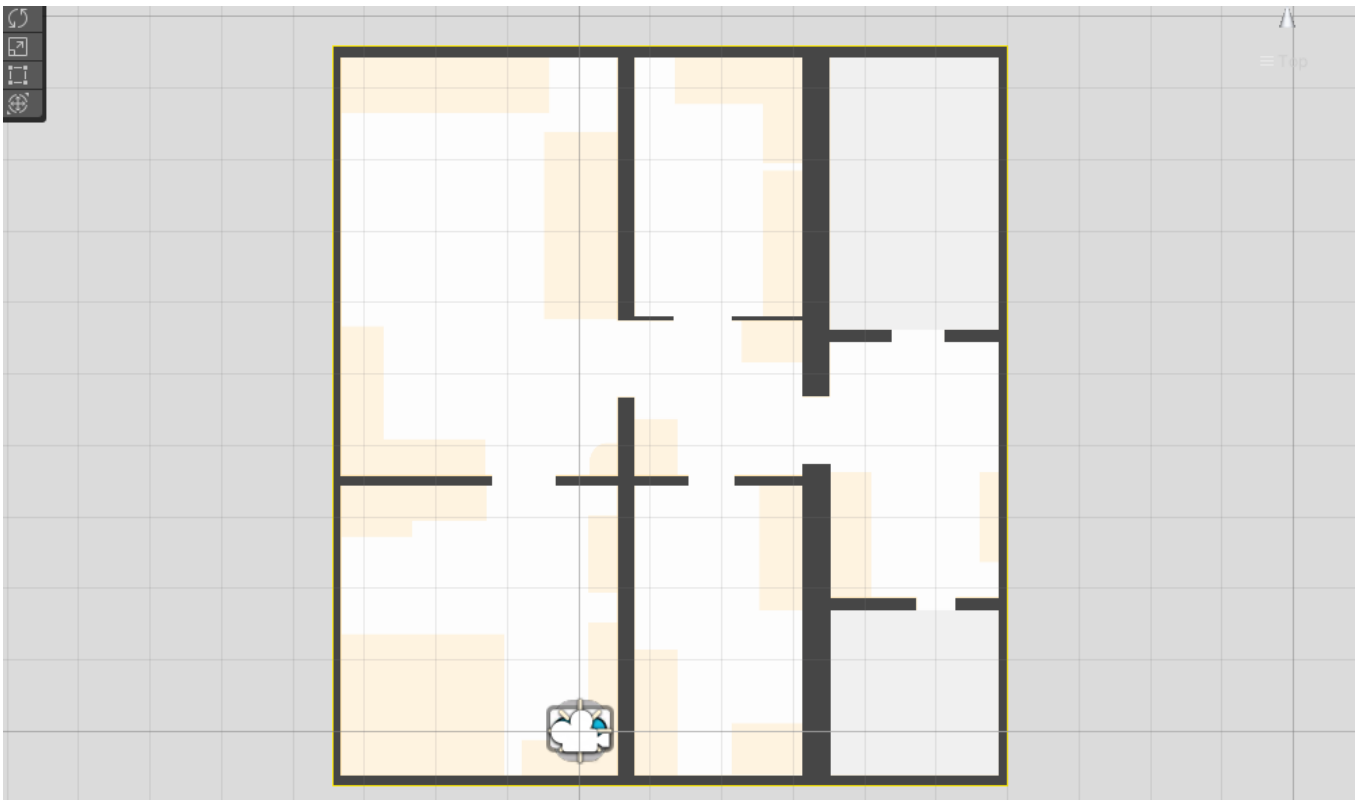


Рис. 3.11. Демонстрація обмеженої зони руху камери (жовта лінія)

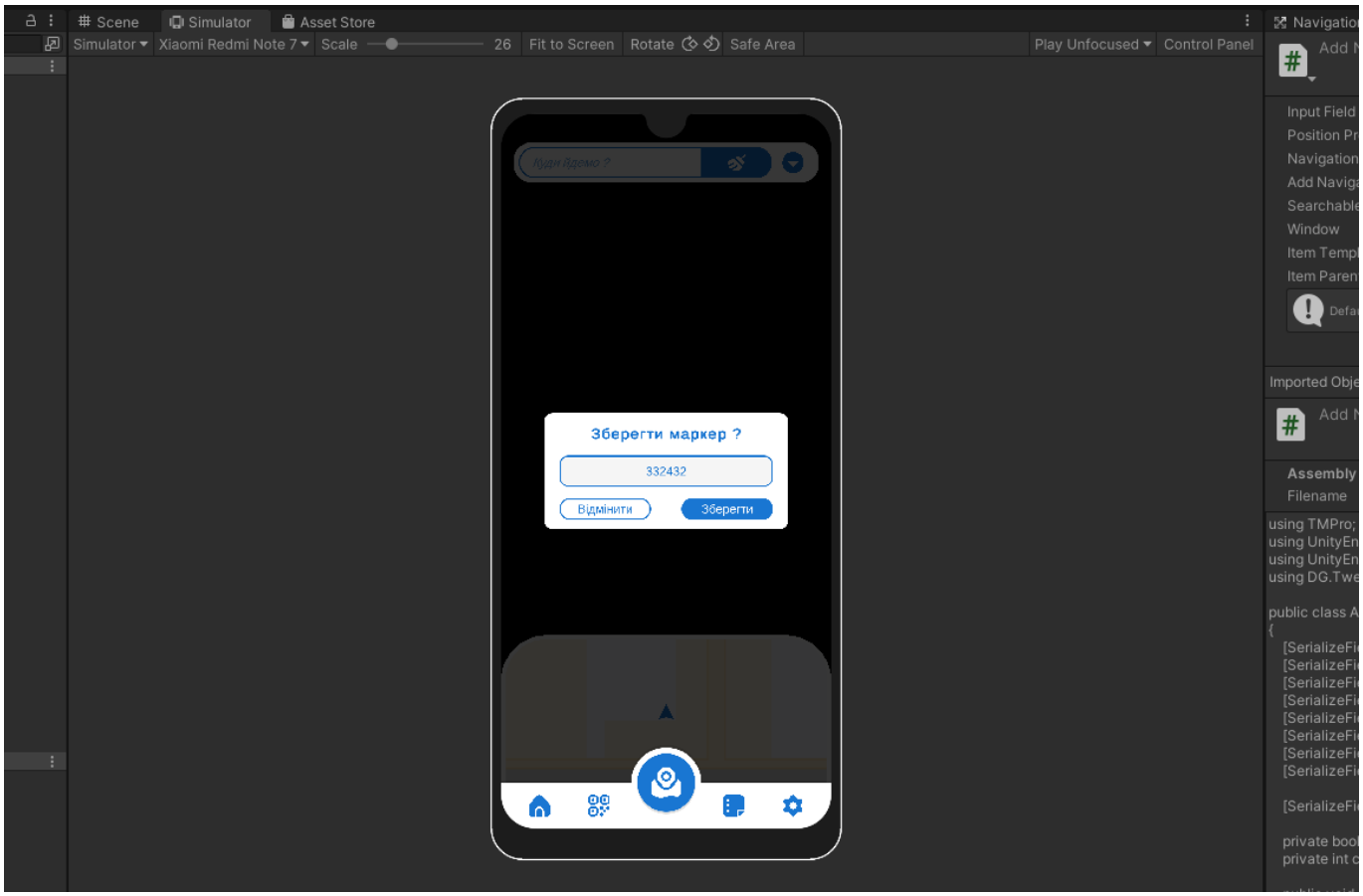
Лістинг коду 3.4. CameraBounds

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraBounds : MonoBehaviour
{
    [SerializeField]
    private Camera cam;
    [SerializeField]
    private Vector2 xBounds, zBounds;

    private void OnDrawGizmos()
    {
        Gizmos.color = Color.yellow;
        Gizmos.DrawLine(new Vector3(xBounds.x, 0, zBounds.x), new Vector3(xBounds.y, 0,
zBounds.x));
        Gizmos.DrawLine(new Vector3(xBounds.x, 0, zBounds.y), new Vector3(xBounds.y, 0,
zBounds.y));
        Gizmos.DrawLine(new Vector3(xBounds.x, 0, zBounds.x), new Vector3(xBounds.x, 0,
zBounds.y));
        Gizmos.DrawLine(new Vector3(xBounds.y, 0, zBounds.x), new Vector3(xBounds.y, 0,
zBounds.y));
```

```
    }

    private void LateUpdate()
    {
        Vector3 clampedPosition = cam.transform.position;
        clampedPosition.x = Mathf.Clamp(clampedPosition.x, xBounds.x, xBounds.y);
        clampedPosition.z = Mathf.Clamp(clampedPosition.z, zBounds.x, zBounds.y);
        cam.transform.position = clampedPosition;
    }
}
```

Примітка. Саме цей код реалізує обмеження руху камери.

Надалі функція додавання нових маркерів та видалення. При натисканні кнопки додавання нового маркера з'являється модальне вікно з полем вводу тексту та кнопками "Зберегти" та "Відмінити". Анімація вікна реалізована за допомогою бібліотеки DG.Tweening, яка дозволяє створювати плавні переходи між значеннями, змінювати властивості об'єктів на сцені та створювати складні анімаційні послідовності зі зручним інтерфейсом програмування. Після введення назви нового маркера та натискання кнопки "Зберегти", створюється новий об'єкт positionMarker на сцені з відповідною назвою та початковими координатами. Крім того, додатково з'являється push-повідомлення, яке повідомляє користувачів про те, що маркер успішно створений (рис 3.12).



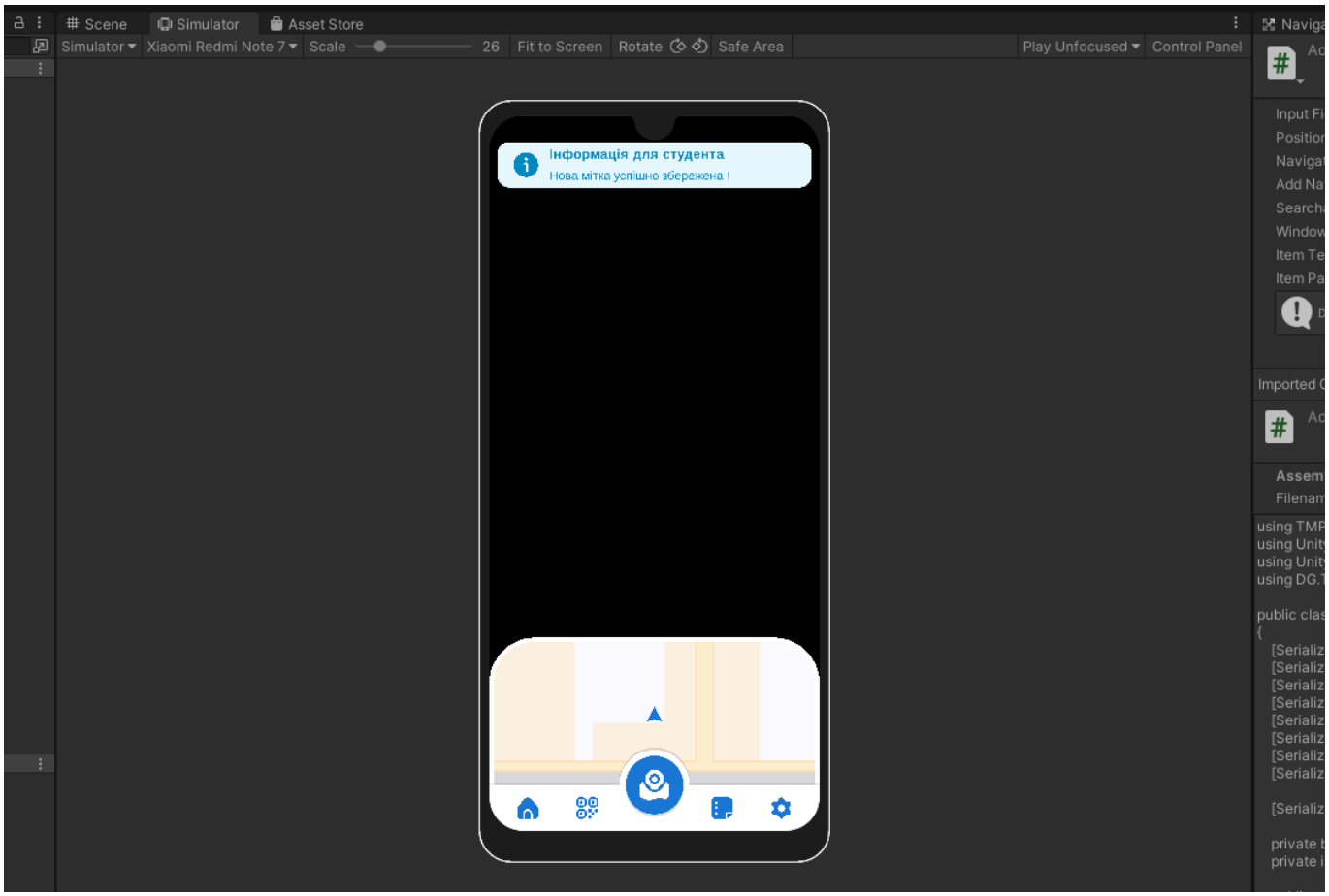


Рис. 3.12. Демонстрація додавання нової мітки та push-повідомлення

Також передбачена можливість видалення створених маркерів. При створенні маркерів у вікні з’являється список всіх створених маркерів, який автоматично оновлюється при додаванні нових маркерів. Кожен елемент списку містить назву маркера та його координати на сцені. Для видалення маркерів зі списку передбачено чек-поінт, що дозволяє вибрати потрібні елементи та видалити їх за допомогою кнопки «Видалити». Після видалення маркера зі списку він повністю видаляється зі сцени (рис 3.13).

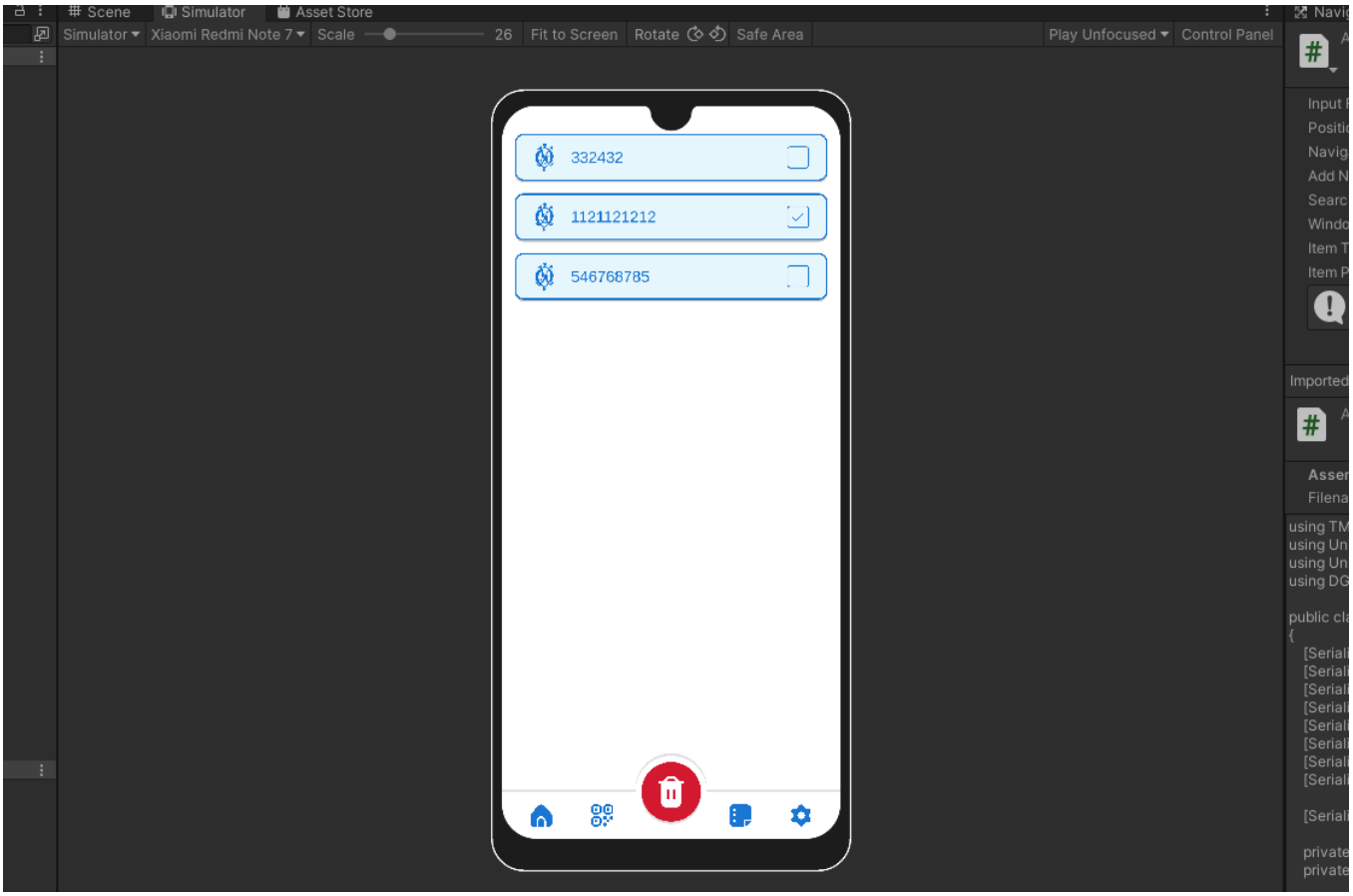


Рис. 3.13. Демонстрація видалення міток

Усі ці функції реалізує скрипт AddNavigationTarget Він містить методи, які викликаються при додаванні нової цілі навігації, підтвердженні додавання, видаленні цілей навігації зі списку та скасуванні операції додавання.

У скрипті використовуються наступні змінні:

- inputField: поле вводу, в якому вводиться назва нової цілі навігації;
- positionPrefab: префаб гри, який використовується для відображення позиції цілі на сцені;
- navigationTarget: скрипт, який містить список цілей навігації;
- addNavigationTargetWindow: графічний елемент, який містить форму для додавання нової цілі навігації;
- searchableDropDown: графічний елемент, який містить випадаючий список всіх цілей навігації;
- window: графічний елемент, який містить форму додавання нової цілі навігації;
- itemTemplate: префаб графічного елементу, який відображає ім'я цілі навігації в списку;
- itemParent: графічний елемент, який містить всі графічні елементи списку цілей навігації;

- changeTimeScale: час, за який форма додавання нової цілі навігації з'являється на екрані;
- isActive: змінна, яка вказує, чи є форма додавання нової цілі навігації активною;
- cloneCount: лічильник кількості клонів префабу гри positionPrefab.

Скрипт містить такі методи:

- OnAddNavigationTarget(): метод, який викликається при натисканні кнопки додавання нової цілі навігації. Він змінює значення змінної isActive на true, відображає форму додавання нової цілі навігації на екрані та плавно збільшує її розмір за допомогою методу DOScale() з бібліотеки анімації DG.Tweening.
- OnOkButtonClick(): метод, який викликається при підтвердженні додавання нової цілі навігації. Він перевіряє, чи введено коректну назву цілі навігації, створює новий об'єкт positionObject з префабу positionPrefab, додає нову ціль навігації до списку цілей навігації, додає нову назву цілі до списку avlOptions графічного елементу searchableDropDown, додає новий елемент до списку цілей навігації, очищує поле вводу та закриває форму додавання нової цілі навігації.
- OnDeleteButtonClick(): метод, який викликається при натисканні кнопки видалення цілі навігації. Він переглядає всі елементи списку цілей навігації та видаляє ті, які відмічені галочкою. Він також видаляє відповідні елементи зі списку avlOptions графічного елементу searchableDropDown та зі списку цілей навігації.
- OnCancelButtonClick(): метод, який викликається при натисканні кнопки скасування операції додавання нової цілі навігації. Він закриває форму додавання нової цілі навігації та плавно зменшує її розмір за допомогою методу DOScale() з бібліотеки анімації DG.Tweening.

Лістинг коду 3.5. AddNavigationTarget

```
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using DG.Tweening;

public class AddNavigationTarget : MonoBehaviour
{
    [SerializeField] private TMP_InputField inputField;
    [SerializeField] private GameObject positionPrefab;
    [SerializeField] private SetNavigationTarget navigationTarget;
    [SerializeField] private GameObject addNavigationTargetWindow;
    [SerializeField] private SearchableDropDown searchableDropDown;
    [SerializeField] private GameObject window;
    [SerializeField] private GameObject itemTemplate;
    [SerializeField] private Transform itemParent;

    [SerializeField] private float changeTimeScale = 1.0f;

    private bool isActive = false;
```

```
    private int cloneCount = 0;

    public void OnAddNavigationTarget()
    {
        isActive = true;
        addNavigationTargetWindow.SetActive(true);
        window.SetActive(true);
        window.transform.localScale = Vector3.zero;
        window.transform.DOScale(1f, changeTimeScale).SetUpdate(true);
    }

    public void OnOkButtonClick()
    {
        string name = inputField.text;

        if (string.IsNullOrEmpty(name))
        {
            Debug.LogWarning("Navigation target name cannot be empty");
            return;
        }

        GameObject positionObject = Instantiate(positionPrefab, Vector3.zero,
Quaternion.identity);

        Target newTarget = new Target
        {
            Name = name,
            PositionObject = positionObject
        };

        navigationTarget.navigationTargetObjects.Add(newTarget);
        searchableDropDown.avlOptions.Add(name); // Додати новий елемент до avlOptions

        Debug.Log("Added new navigation target: " + name);

        // Додавання до ієрархії Environment/NavigationTarget/...
        Transform navigationTargetParent =
GameObject.Find("Environment/NavigationTarget").transform;
        positionObject.transform.SetParent(navigationTargetParent, false);

        // Збільшення лічильника клонів та додавання до імені клону
        cloneCount++;
        positionObject.name = positionPrefab.name + " Clone " + cloneCount;

        GameObject arrowObject = GameObject.FindGameObjectWithTag("Arrow");
        if (arrowObject != null)
        {
            Vector3 arrowPosition = arrowObject.transform.position;
            Quaternion arrowRotation = arrowObject.transform.rotation;
            Vector3 spawnPosition = new Vector3(arrowPosition.x, 0f, arrowPosition.z);
            positionObject.transform.position = spawnPosition;
            positionObject.transform.rotation = arrowRotation;
        }
        else
        {
            Debug.LogWarning("Cannot find object with tag 'Arrow'");
        }

        // Додавання нового елемента до списку
        GameObject newItem = Instantiate(itemTemplate, itemParent);
        newItem.transform.SetAsLastSibling();
        newItem.SetActive(true);

        // Копіювання тексту до TextTMP нового елемента
        TMP_Text itemText = newItem.GetComponentInChildren<TMP_Text>();
```

```
        itemText.text = name;

        // Очистити значення InputField
        inputField.text = "";

        OnCancelButtonClick();
    }

    public void OnDeleteButtonClick()
    {
        foreach (Transform child in itemParent)
        {
            GameObject item = child.gameObject;
            Toggle toggle = item.GetComponentInChildren<Toggle>();
            if (toggle != null && toggle.isOn == true)
            {
                TMP_Text itemText = item.GetComponentInChildren<TMP_Text>();
                string itemName = itemText.text;
                searchableDropDown.avlOptions.Remove(itemName);

                // Знайти об'єкт з ім'ям, яке співпадає з ім'ям видаленого елемента з
avlOptions
                Transform content = searchableDropDown.transform.Find("Scroll
View/Viewport/Content");
                foreach (Transform children in content)
                {
                    TMP_Text childText = children.GetComponentInChildren<TMP_Text>();
                    if (childText != null && childText.text == itemName)
                    {
                        // Знайдено відповідний об'єкт - видалити його
                        if (children.gameObject.name.Contains("Clone"))
                        {
                            Destroy(children.gameObject.transform.parent.gameObject);
                        }
                        else
                        {
                            Destroy(children.gameObject);
                        }
                        break;
                    }
                }

                // Видалити елемент з navigationTargetObjects
                Target targetToRemove = null;
                foreach (Target target in navigationTarget.navigationTargetObjects)
                {
                    if (target.Name == itemName)
                    {
                        targetToRemove = target;
                        break;
                    }
                }
                if (targetToRemove != null)
                {
                    navigationTarget.navigationTargetObjects.Remove(targetToRemove);

                    // Видалити префаб зі сцени, який пов'язаний з
navigationTargetObjects
                    Destroy(targetToRemove.PositionObject);
                }

                // Видалити елемент зі сцени
                Destroy(item);
            }
        }
```

```
    }

    public void OnCancelButtonClick()
    {
        isActive = false;
        window.transform.DOScale(0f, changeTimeScale).OnComplete(() =>
        {
            window.SetActive(false);
            addNavigationTargetWindow.SetActive(false);
        }).SetUpdate(true);
    }
}
```

Примітка. Саме цей код реалізує додавання та видалення макерів.

Наступним кроком було створення скрипту QRCodeRecenter для розпізнавання QR-кодів з використанням камери на пристрої з підтримкою доповненої реальності (ARFoundation) у середовищі Unity. Основна мета скрипта полягає в знаходженні QR-коду на зображенні, отриманому з камери, розпізнаванні його та використанні отриманих даних для перенесення AR-об'єкту до нової позиції в просторі. Скрипт містить приватні та публічні змінні, які використовуються для зберігання посилань на AR-об'єкти, такі як ARSession, ARSessionOrigin, ARCameraManager, та список AR-об'єктів, що пов'язані з QR-кодами, які потрібно розпізнати. Також вибірково задаються об'єкти UI, які відповідають за відображення екранів при запуску додатку.

Основний метод OnCameraFrameReceived () обробляє дані, отримані з камери, та знаходить QR-код на зображенні, використовуючи бібліотеку ZXing. Після розпізнавання QR-коду викликається метод SetQrCodeRecenterTarget (), який заново позиціонує AR-об'єкт відповідно до інформації, отриманої з QR-коду. Якщо QR-код розпізнаний успішно, скрипт активує деякі об'єкти UI та деактивує інші (рис 3.14).

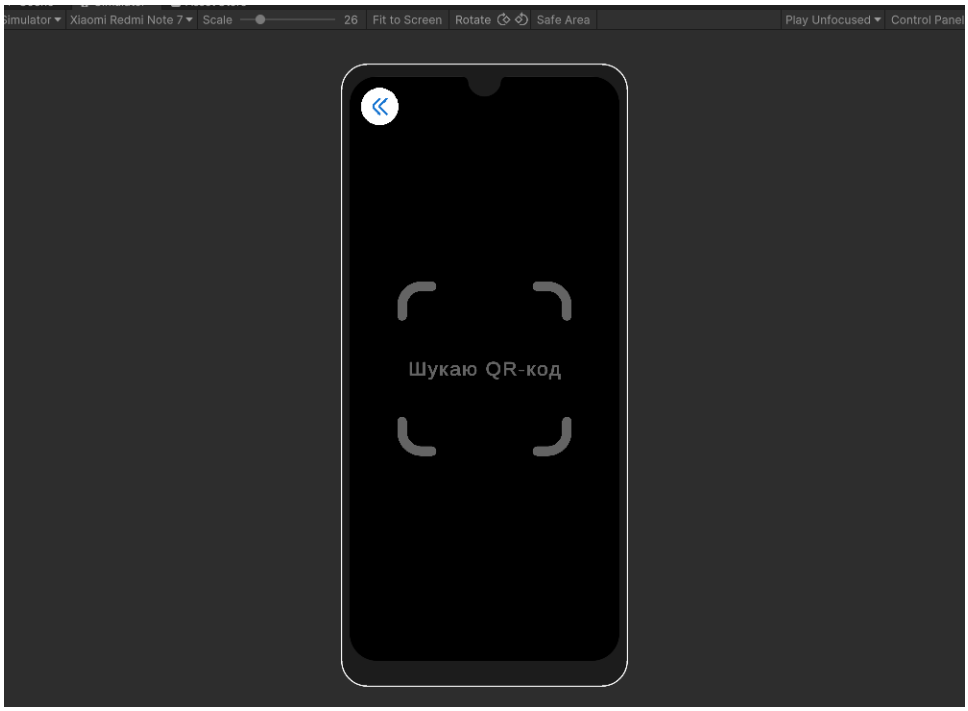


Рис. 3.14. Демонстрація вікна сканування QR-кодів

Лістинг коду 3.6. QRCodeRecenter

```
using System.Collections.Generic;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using UnityEngine;
using ZXing;
using Unity.Collections;

public class QRCodeRecenter : MonoBehaviour
{
    [SerializeField] private ARSession session;
    [SerializeField] private ARSessionOrigin origin;
    [SerializeField] private ARCameraManager cameraManager;
    [SerializeField] private List<Target> navigationTargetObjects = new List<Target>();
    [SerializeField] private GameObject ScanPage;
    [SerializeField] private GameObject HomePage;
    [SerializeField] private GameObject NavBar;
    [SerializeField] private GameObject ButtonOn;



    private Texture2D cameraImageTexture;
    private IBarcodeReader reader = new BarcodeReader();

    private void OnEnable()
    {
        cameraManager.frameReceived += OnCameraFrameReceived;
    }

    private void OnDisable()
    {
        cameraManager.frameReceived -= OnCameraFrameReceived;
    }

    private void OnCameraFrameReceived(ARCameraFrameEventArgs eventArgs)
    {
        if (!cameraManager.TryAcquireLatestCpuImage(out XRCpuImage image))
        {
            return;
        }

        var conversionParams = new XRCpuImage.ConversionParams
        {
            inputRect = new RectInt(0, 0, image.width, image.height),
            outputDimensions = new Vector2Int(image.width / 2, image.height / 2),
            outputFormat = TextureFormat.RGBA32,
            transformation = XRCpuImage.Transformation.MirrorY
        };

        int size = image.GetConvertedDataSize(conversionParams);

        var buffer = new NativeArray<byte>(size, Allocator.Temp);

        image.Convert(conversionParams, buffer);

        image.Dispose();

        cameraImageTexture = new Texture2D(
            conversionParams.outputDimensions.x,
            conversionParams.outputDimensions.y,
            conversionParams.outputFormat,
            false);
```

```
        cameraImageTexture.LoadRawTextureData(buffer);
        cameraImageTexture.Apply();

        buffer.Dispose();

        if (ScanPage.activeSelf)
        {
            var result = reader.Decode(cameraImageTexture.GetPixels32(),
cameraImageTexture.width, cameraImageTexture.height);

            if (result != null)
            {
                SetQrCodeRecenterTarget(result.Text);
                ScanPage.SetActive(false);
                HomePage.SetActive(true);
                NavBar.SetActive(true);
                ButtonOn.SetActive(true);

            }
        }
    }

    private void SetQrCodeRecenterTarget(string targetText)
    {
        Target currentTarget = navigationTargetObjects.Find(x =>
x.Name.ToLower().Equals(targetText.ToLower()));
        if (currentTarget != null)
        {
            session.Reset();

            origin.transform.position = currentTarget.PositionObject.transform.position;
            origin.transform.rotation = currentTarget.PositionObject.transform.rotation;
        }
    }
}
```

Примітка. Саме цей код реалізує сканування та знаходження місце положення користувача.

Для поліпшення користувацького досвіду та забезпечення зручності користувачів, я створив інтерфейс з деякими налаштуваннями. Ці налаштування дозволяють користувачам персоналізувати додаток під свої потреби та вимоги (рис 3.15).

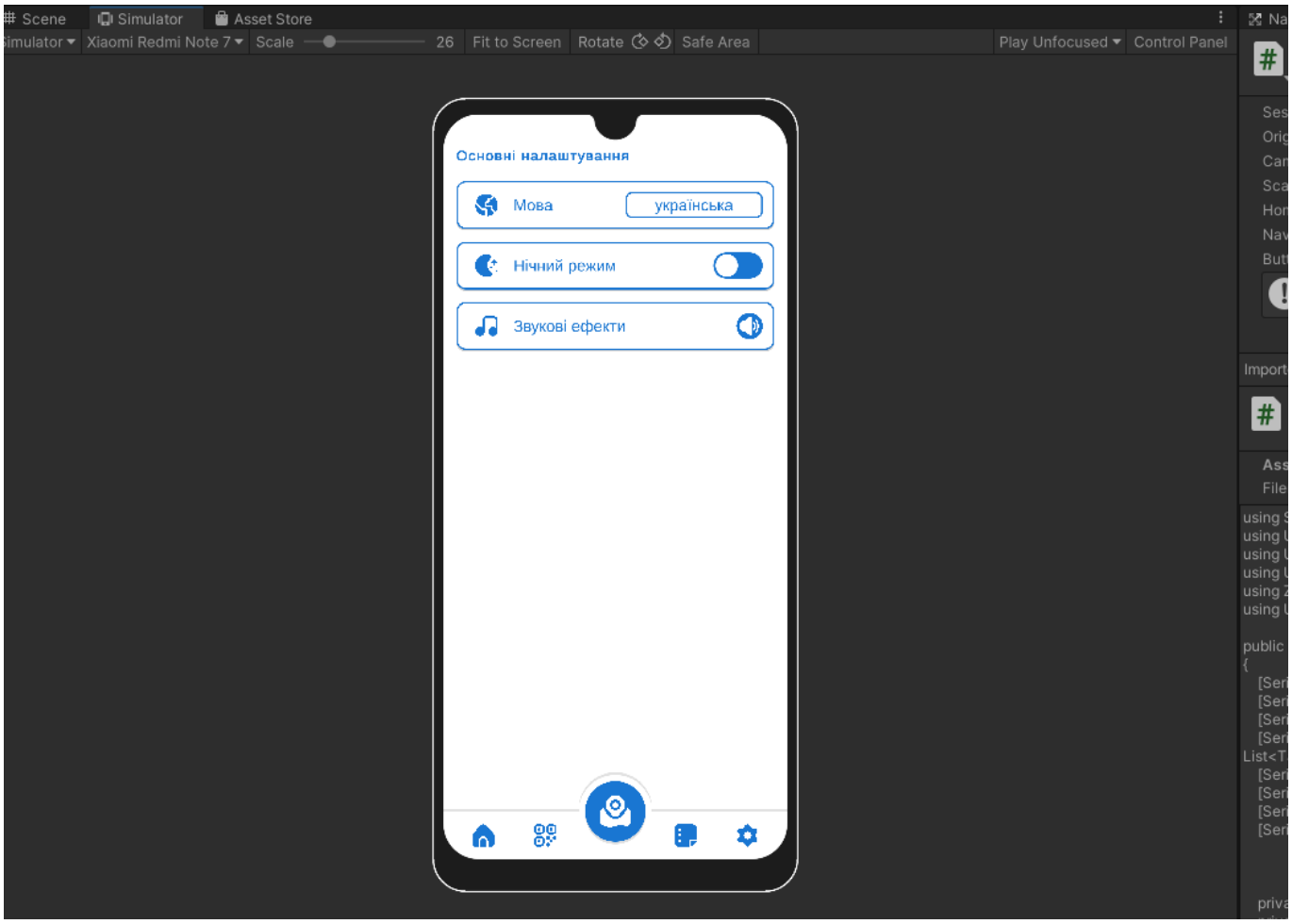
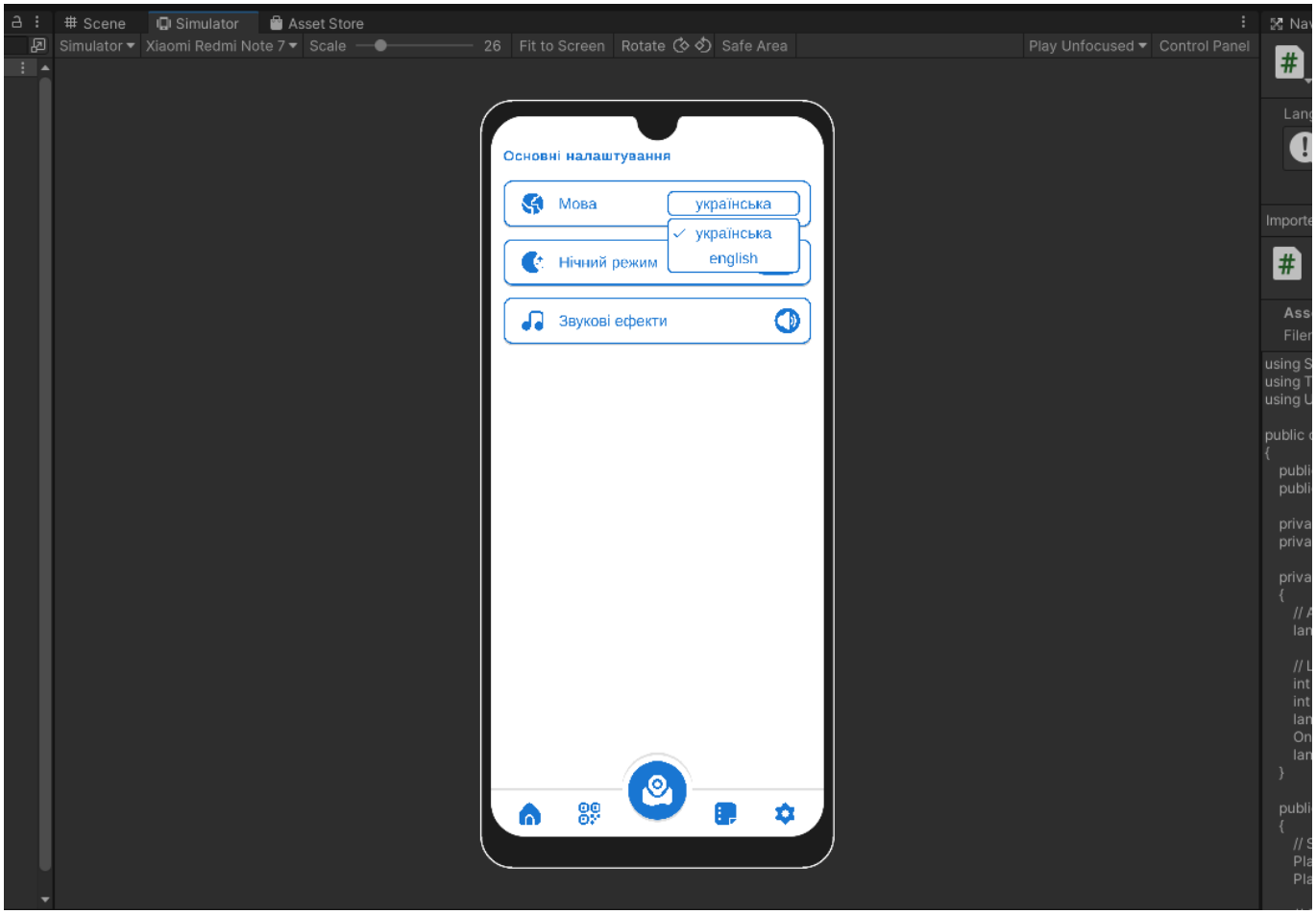


Рис. 3.15. Демонстрація вікна налаштувань

Однією з можливостей інтерфейсу є зміна мови. Ця функція дозволяє користувачам змінювати мову додатку на бажану. Це особливо корисно для міжнародних додатків, де користувачі з різних країн мають різні мовні налаштування. Для реалізації даної функцію було створено скрипт LanguageSwitcher, який містить TMP_Dropdown для вибору мови та список LanguageData, який містить переклади тексту для кожної мови.

При запуску гри скрипт додає варіанти мов до TMP_Dropdown на основі списку LanguageData та завантажує обрану мову та значення збережені в PlayerPrefs. Після вибору мови в TMP_Dropdown викликається метод OnLanguageChanged, який зберігає обрану мову та значення в PlayerPrefs та викликає метод SetLanguage для зміни мови на обрану. Метод SetLanguage знаходить LanguageData для вибраної мови та змінює текст для всіх елементів TMP_Text на екрані, використовуючи метод GetTranslation. Метод GetTranslation знаходить переклад тексту для обраної мови зі списку TranslationData відповідного LanguageData. Якщо переклад не знайдено, повертається оригінальний текст. Клас LanguageData містить інформацію про мову, що перекладається, та список TranslationData, який містить переклади тексту для

кожного ключа. Клас TranslationData містить пару ключ-значення для перекладу тексту (рис 3.16).



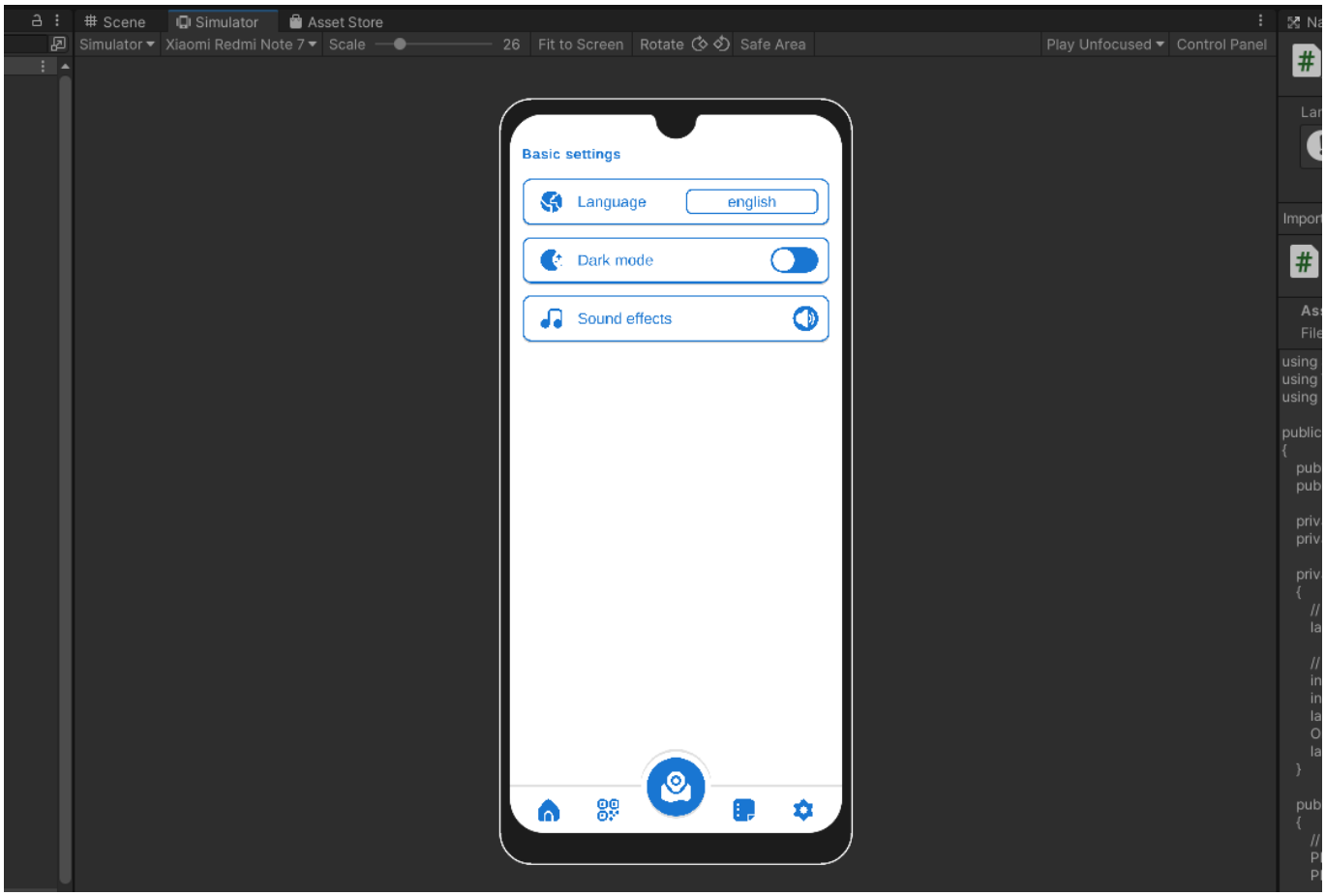


Рис. 3.16. Демонстрація зміни мови

Лістинг коду 3.6. LanguageSwitcher

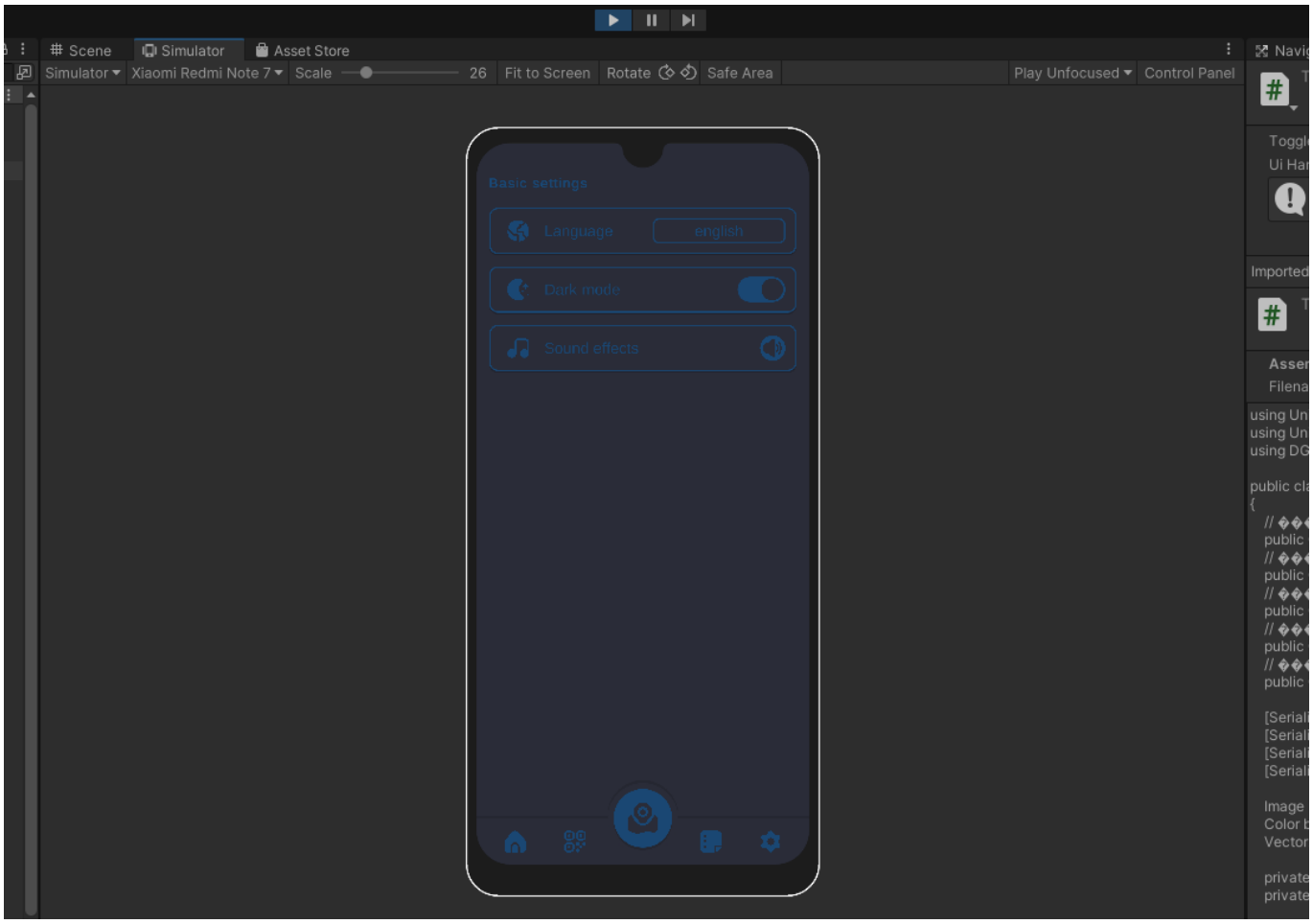
```
using System.Collections.Generic;
using TMPro;
using UnityEngine;

public class LanguageSwitcher : MonoBehaviour
{
    public TMP_Dropdown languageDropdown;
    public List<LanguageData> languageDataList;

    private const string LANGUAGE_KEY = "selected_language";
    private const string VALUE_KEY = "selected_value";

    private void Start()
    {
        languageDropdown.AddOptions(GetLanguageNames());

        int selectedLanguage = PlayerPrefs.GetInt(LANGUAGE_KEY, 0);
        int selectedValue = PlayerPrefs.GetInt(VALUE_KEY, 0);
        languageDropdown.SetValueWithoutNotify(selectedLanguage);
        OnLanguageChanged(selectedLanguage);
        languageDropdown.RefreshShownValue();
    }

    public void OnLanguageChanged(int index)
    {
        PlayerPrefs.SetInt(LANGUAGE_KEY, index);
        PlayerPrefs.SetInt(VALUE_KEY, languageDropdown.value);
```

```
        SetLanguage(index);
    }

    private void SetLanguage(int index)
    {
        LanguageData selectedLanguageData = languageDataList[index];

        foreach (var textMeshPro in Resources.FindObjectsOfTypeAll<TMP_Text>())
        {
            string translatedText =
selectedLanguageData.GetTranslation(textMeshPro.text);

            textMeshPro.text = translatedText;
        }
    }

    private List<string> GetLanguageNames()
    {
        List<string> languageNames = new List<string>();

        foreach (var languageData in languageDataList)
        {
            languageNames.Add(languageData.languageName);
        }

        return languageNames;
    }
}

[System.Serializable]
public class LanguageData
{
    public string languageName;
    public List<TranslationData> translationDataList;

    public string GetTranslation(string key)
    {
        foreach (var translationData in translationDataList)
        {
            if (translationData.key == key)
            {
                return translationData.value;
            }
        }

        return key;
    }
}

[System.Serializable]
public class TranslationData
{
    public string key;
    public string value;
}
```

Примітка. Саме цей код реалізує заміну мови.

Іншою можливістю інтерфейсу є зміна теми. Ця функція дозволяє користувачам змінювати вигляд додатку на бажаний. Наприклад, можна вибрати світлу тему для денного використання або темну тему для використання вночі. Це допомагає знизити втому очей та забезпечити комфортну роботу з додатком. Дана функція реалізована

за допомогою скрипту ToggleController. При старті гри скрипт завантажує збережені значення стану Toggle та вибрані кольори за допомогою PlayerPrefs. Якщо Toggle був увімкнений, то виконується метод SetToggleActive, який змінює кольори елементів UI відповідно до вибраних кольорів. Якщо Toggle був вимкнений, то виконується метод SetToggleInactive, який повертає кольори елементів UI до їхніх початкових значень. Колір кожного елементу UI визначається за його кольором за замовчуванням і списком нових кольорів, що передаються у скрипт. Колір замінюється за допомогою методу DOColor з бібліотеки DOTween, який забезпечує плавну анімацію зміни кольору (рис 3.17).



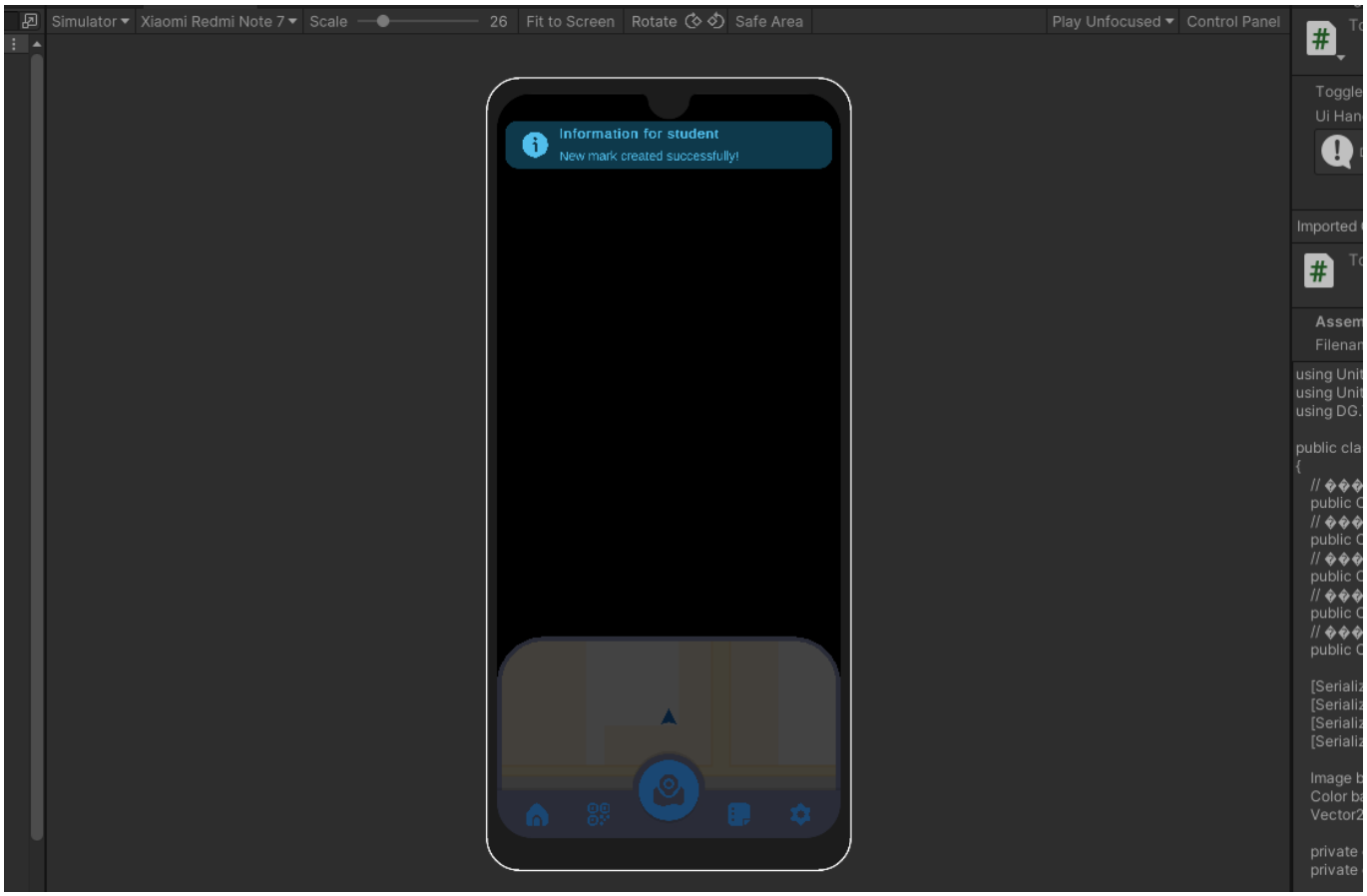


Рис. 3.17. Демонстрація зміни теми на темний

Лістинг коду 3.7. ToggleController

```
using UnityEngine;
using UnityEngine.UI;
using DG.Tweening;

public class ToggleController : MonoBehaviour
{
    // Колір, на який потрібно замінити #1976D2
    public Color newColor1 = new Color();
    // Колір, на який потрібно замінити #FFFFFF
    public Color newColor2 = new Color();
    // Колір, на який потрібно замінити #E5F6FD
    public Color newColor3 = new Color();
    // Колір, на який потрібно замінити #0088C0
    public Color newColor4 = new Color();
    // Колір, на який потрібно замінити #FBFBFF
    public Color newColor5 = new Color();

    [SerializeField] private Toggle toggle;
    [SerializeField] RectTransform uiHandleRectTransform;
    [SerializeField] Color backgroundActiveColor;
    [SerializeField] Color handleActiveColor;

    Image backgroundImage, handleImage;
    Color backgroundDefaultColor, handleDefaultColor;
    Vector2 handlePosition;

    private const string toggleKey = "toggleState";
    private const string colorKey = "selectedColor";
```

```
    private void Start()
    {
        toggle.onValueChanged.AddListener(OnToggleValueChanged);
        handlePosition = uiHandleRectTransform.anchoredPosition;
        backgroundImage = uiHandleRectTransform.parent.GetComponent<Image>();
        handleImage = uiHandleRectTransform.GetComponent<Image>();
        backgroundDefaultColor = backgroundImage.color;
        handleDefaultColor = handleImage.color;

        // Відновлення стану Toggle
        if (PlayerPrefs.HasKey(toggleKey))
        {
            bool isToggled = PlayerPrefs.GetInt(toggleKey) == 1;
            toggle.isOn = isToggled;
            if (isToggled)
                SetToggleActive();
        }

        // Відновлення вибраних кольорів
        if (PlayerPrefs.HasKey(colorKey))
        {
            string[] colorsJson = PlayerPrefs.GetString(colorKey).Split(',');
            if (colorsJson.Length == 5)
            {
                newColor1 = JsonUtility.FromJson<Color>(colorsJson[0]);
                newColor2 = JsonUtility.FromJson<Color>(colorsJson[1]);
                newColor3 = JsonUtility.FromJson<Color>(colorsJson[2]);
                newColor4 = JsonUtility.FromJson<Color>(colorsJson[3]);
                newColor5 = JsonUtility.FromJson<Color>(colorsJson[4]);
            }
        }
    }

    private void OnDestroy()
    {
        toggle.onValueChanged.RemoveListener(OnToggleValueChanged);
    }

    private void OnToggleValueChanged(bool isOn)
    {
        if (isOn)
            SetToggleActive();
        else
            SetToggleInactive();
    }

    private void SetToggleActive()
    {
        // Збереження стану Toggle
        PlayerPrefs.SetInt(toggleKey, 1);

        // Збереження вибраних кольорів
        PlayerPrefs.SetString(colorKey, JsonUtility.ToJson(newColor1) + "," +
                                JsonUtility.ToJson(newColor2) + "," +
                                JsonUtility.ToJson(newColor3) + "," +
                                JsonUtility.ToJson(newColor4) + "," +
                                JsonUtility.ToJson(newColor5));

        // Виконання інших дій
        uiHandleRectTransform.DOAnchorPos(handlePosition * -1,
.4f).SetEase(Ease.InOutBack);
        backgroundImage.DOColor(backgroundActiveColor, .6f);
        handleImage.DOColor(handleActiveColor, .4f);

        var uiElements = Resources.FindObjectsOfTypeAll<Graphic>();
```

```
        foreach (var uiElement in uiElements)
        {
            var color = uiElement.color;
            if (color.Equals(new Color(0x19 / 255f, 0x76 / 255f, 0xD2 / 255f)))
            {
                uiElement.color = newColor1;
            }
            else if (color.Equals(Color.white))
            {
                uiElement.color = newColor2;
            }
            else if (color.Equals(new Color(0xE5 / 255f, 0xF6 / 255f, 0xFD / 255f)))
            {
                uiElement.color = newColor3;
            }
            else if (color.Equals(new Color(0x00 / 255f, 0x88 / 255f, 0xC0 / 255f)))
            {
                uiElement.color = newColor4;
            }
            else if (color.Equals(new Color(0xFB / 255f, 0xFB / 255f, 0xFF / 255f)))
            {
                uiElement.color = new Color(0x43 / 255f, 0x43 / 255f, 0x43 / 255f);
            }
        }
    }

    private void SetToggleInactive()
    {
        // Збереження стану Toggle
        PlayerPrefs.SetInt(toggleKey, 0);

        // Виконання інших дій
        uiHandleRectTransform.DOAnchorPos(handlePosition, .4f).SetEase(Ease.InOutBack);
        backgroundImage.DOColor(backgroundDefaultColor, .6f);
        handleImage.DOColor(handleDefaultColor, .4f);

        var uiElements = Resources.FindObjectsOfTypeAll<Graphic>();
        foreach (var uiElement in uiElements)
        {
            var color = uiElement.color;
            if (color.Equals(newColor1))
            {
                uiElement.color = new Color(0x19 / 255f, 0x76 / 255f, 0xD2 / 255f);
            }
            else if (color.Equals(newColor2))
            {
                uiElement.color = Color.white;
            }
            else if (color.Equals(newColor3))
            {
                uiElement.color = new Color(0xE5 / 255f, 0xF6 / 255f, 0xFD / 255f);
            }
            else if (color.Equals(newColor4))
            {
                uiElement.color = new Color(0x00 / 255f, 0x88 / 255f, 0xC0 / 255f);
            }
            else if (color.Equals(new Color(0x43 / 255f, 0x43 / 255f, 0x43 / 255f)))
            {
                uiElement.color = new Color(0xFB / 255f, 0xFB / 255f, 0xFF / 255f);
            }
        }
    }
}
```

Примітка. Саме цей код реалізує заміну теми.

Третя функція - вимкнення звукових ефектів. Ця функція дозволяє користувачам вимкнути звуки, які супроводжують різні події додатку, такі як повідомлення про наближення до кабінету. Це може бути корисно для користувачів, які працюють у зоні, де потрібна тиша, або для тих, хто просто не хоче слухати звуки. Реалізовано за допомогою скрипту ChangeImageOnClick, що відповідає за зміну зображення (Sprite) та стану вимкнення звуку в грі при кліку на кнопку.

При запуску гри скрипт завантажує збережений стан вимкнення звуку та вибране зображення за допомогою PlayerPrefs. Якщо звук був вимкнений, то audioSource.mute буде встановлено в true, інакше - в false. Якщо було вибрано інше зображення, то image.sprite буде встановлено на alternateSprite, інакше - на originalSprite. Коли користувач клікає на кнопку, метод ToggleImageAndMute перевіряє поточне зображення. Якщо зображення - originalSprite, то зображення змінюється на alternateSprite, інакше - на originalSprite. Після цього метод перевіряє, чи був вимкнений звук, і змінює його на протилежне значення. Нарешті, скрипт зберігає нові значення стану вимкнення звуку та вибраного зображення за допомогою PlayerPrefs (рис 3.18).

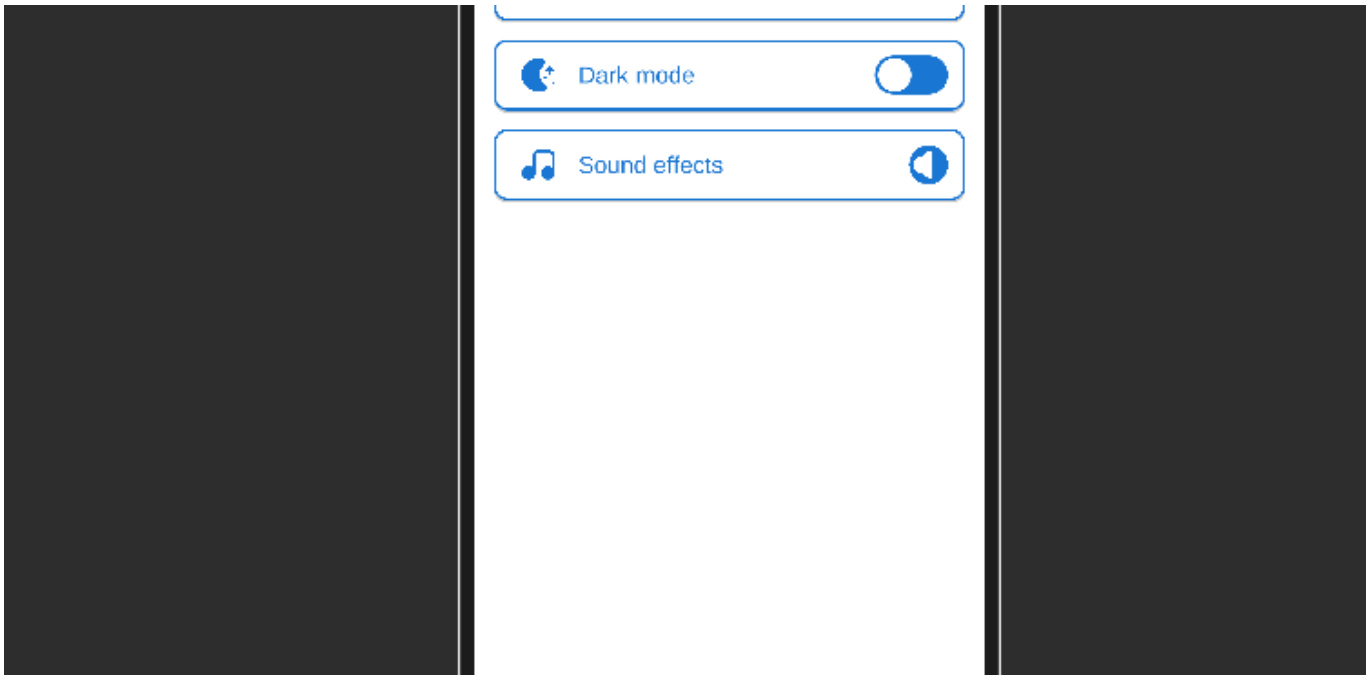


Рис. 3.18. Демонстрація вимкнення звуку

Лістинг коду 3.8. ChangeImageOnClick

```
using UnityEngine;
using UnityEngine.UI;

public class ChangeImageOnClick : MonoBehaviour
{
    [SerializeField] private Button buttonUI;
    [SerializeField] private Image image;
    [SerializeField] private Sprite alternateSprite;
```

```
    [SerializeField] private AudioSource audioSource;
    private Sprite originalSprite;
    private bool isMuted = false;
    private const string MUTE_KEY = "mute";
    private const string IMAGE_KEY = "image";

    private void Start()
    {
        originalSprite = image.sprite;
        buttonUI.onClick.AddListener(ToggleImageAndMute);

        if (PlayerPrefs.HasKey(MUTE_KEY))
        {
            isMuted = PlayerPrefs.GetInt(MUTE_KEY) == 1;
            audioSource.mute = isMuted;
        }

        if (PlayerPrefs.HasKey(IMAGE_KEY))
        {
            int imageIndex = PlayerPrefs.GetInt(IMAGE_KEY);
            if (imageIndex == 1)
            {
                image.sprite = alternateSprite;
            }
        }
    }

    private void ToggleImageAndMute()
    {
        if (image.sprite == originalSprite)
        {
            image.sprite = alternateSprite;
            PlayerPrefs.SetInt(IMAGE_KEY, 1);
        }
        else
        {
            image.sprite = originalSprite;
            PlayerPrefs.SetInt(IMAGE_KEY, 0);
        }

        isMuted = !isMuted;
        audioSource.mute = isMuted;
        PlayerPrefs.SetInt(MUTE_KEY, isMuted ? 1 : 0);
    }
}
```

Примітка. Саме цей код реалізує вимкнення звку.

Всі ці функції дозволяють користувачам налаштувати додаток під свої потреби та вимоги. Інтерфейс з цими налаштуваннями допомагає зробити додаток більш зручним та приємним у використанні, що позитивно впливає на задоволення та лояльність користувачів.

3.3.6. Збереження та завантаження даних

Для забезпечення повного функціоналу додатку було додано можливість збереження даних. Зокрема, було реалізовано збереження маркерів та зміну налаштувань. Для збереження даних було використано два способи:

- Збереження даних за допомогою XML – це мова розмітки, що використовується для обміну даними між різними системами. В Unity, XML може бути використаний для зберігання та обміну даними між різними скриптами, сценами або навіть між різними програмами. XML використовується для опису структури даних за допомогою тегів, які можуть мати атрибути та вкладені елементи. Кожен елемент має ім'я та значення, що дозволяє зберігати різноманітні дані, такі як текст, числа, дати, зображення та інші. У Unity, XML може бути використаний для зберігання налаштувань гри, стану гри, даних ігрових об'єктів, даних користувачів та багатьох інших речей. XML-документи можуть бути збережені в текстових файлах, що дозволяє легко зберігати та обмінювати дані між різними системами. Для обробки XML в Unity можна використовувати стандартний пакет .NET Framework, який містить класи для роботи з XML-документами, такі як XmlDocument та XmlReader. Крім того, є також багато сторонніх бібліотек, які дозволяють зручно працювати з XML в Unity.
- Збереження даних за допомогою PlayerPrefs – це компонент Unity, який дозволяє зберігати та отримувати дані на протязі сеансу гри або між сеансами гри. PlayerPrefs дозволяє зберігати значення ключ-значення, де ключ - це рядок, а значення - це ціле число, дійсне число або рядок. PlayerPrefs дуже корисний для зберігання налаштувань гри, рівнів, отриманих досягнень та інших даних, які повинні бути збережені між різними сеансами гри. Це дозволяє гравцям зберігати свій прогрес та налаштування, навіть коли вони закрили гру або вимкнули комп'ютер. PlayerPrefs зберігає дані в бінарному форматі в файлі конфігурації гри. Цей файл зберігається на жорсткому диску комп'ютера або на пристрої, на якому запущена гра. Для доступу до даних, збережених в PlayerPrefs, використовуються методи SetInt(), SetFloat(), SetString(), GetInt(), GetFloat(), GetString() та інші. Важливо мати на увазі, що PlayerPrefs не призначений для зберігання великих обсягів даних, так як це може призвести до зниження продуктивності гри. Також PlayerPrefs не є надійним засобом збереження даних, тому не слід використовувати його для зберігання конфіденційних даних, таких як паролі або кредитні картки.

Стан маркерів зберігався за допомогою XML використовуючи скрипт NavigationTargetsData. Для збереження даних скрипт використовує XML-серіалізацію, тобто перетворення об'єктів в XML-формат для зберігання на диск. Для цього використовується клас XmlSerializer, який дозволяє зберігати дані в зрозумілому форматі. Скрипт містить два класи: NavigationTargetsData та TargetData, які використовуються для зберігання даних про навігаційні цілі. Клас NavigationTargetsData містить список об'єктів класу TargetData, який містить ім'я

навігаційної цілі та її позицію. Скрипт містить також методи SaveData() та LoadData(), які використовуються для збереження та завантаження даних про навігаційні цілі з файлу на диск. Після завантаження даних скрипт використовує метод PopulateNavigationTargets(), який створює об'єкти навігаційних цілей на основі завантажених даних та додає їх до списку навігаційних цілей в грі. Крім того, метод PopulateNavigationTargets() також додає назви навігаційних цілей до випадаючого списку та до списку на екрані.

Лістинг коду 3.9. NavigationTargetsData

```
using System.Collections.Generic;
using System.IO;
using System.Xml.Serialization;
using TMPro;
using UnityEngine;

public class NavigationTargetsData
{
    public List<TargetData> Targets;
}

public class TargetData
{
    public string Name;
    public float[] Position;
}

public class NavigationTargetsManager : MonoBehaviour
{
    [SerializeField] private GameObject positionPrefab;
    [SerializeField] private SetNavigationTarget navigationTarget;
    [SerializeField] private SearchableDropDown searchableDropDown;
    [SerializeField] private GameObject itemTemplate;
    [SerializeField] private Transform itemParent;

    private NavigationTargetsData data;

    private void Start()
    {
        LoadData();
        PopulateNavigationTargets();
    }

    public void SaveData()
    {
        data = new NavigationTargetsData();
        data.Targets = new List<TargetData>();
        foreach (Target target in navigationTarget.navigationTargetObjects)
        {
            GameObject positionObject = target.PositionObject;
            if (!positionObject.CompareTag("NoSave"))
            {
                TargetData targetData = new TargetData();
                targetData.Name = target.Name;
                targetData.Position = new float[3];
                targetData.Position[0] = positionObject.transform.position.x;
                targetData.Position[1] = positionObject.transform.position.y;
                targetData.Position[2] = positionObject.transform.position.z;
                data.Targets.Add(targetData);
            }
```

```
        }

        XmlSerializer serializer = new XmlSerializer(typeof(NavigationTargetsData));
        using (StreamWriter streamWriter = new
StreamWriter(Application.persistentDataPath + "/navigationTargets.xml"))
        {
            serializer.Serialize(streamWriter, data);
        }

        Debug.Log("Navigation targets data saved");
    }
    public void LoadData()
    {
        XmlSerializer serializer = new XmlSerializer(typeof(NavigationTargetsData));
        try
        {
            using (StreamReader streamReader = new
StreamReader(Application.persistentDataPath + "/navigationTargets.xml"))
            {
                data = (NavigationTargetsData)serializer.Deserialize(streamReader);
            }
            Debug.Log("Navigation targets data loaded");
        }
        catch (FileNotFoundException)
        {
            Debug.LogWarning("Navigation targets data not found");
            data = new NavigationTargetsData();
            data.Targets = new List<TargetData>();
        }
    }
    private void PopulateNavigationTargets()
    {
        foreach (TargetData targetData in data.Targets)
        {
            string name = targetData.Name;
            GameObject positionObject = Instantiate(positionPrefab, Vector3.zero,
Quaternion.identity);
            positionObject.transform.position = new Vector3(targetData.Position[0],
targetData.Position[1], targetData.Position[2]);

            Target newTarget = new Target
            {
                Name = name,
                PositionObject = positionObject
            };
            navigationTarget.navigationTargetObjects.Add(newTarget);
            searchableDropDown.avlOptions.Add(name);
            Transform navigationTargetParent =
GameObject.Find("Environment/NavigationTarget").transform;
            positionObject.transform.SetParent(navigationTargetParent, false);

            int cloneCount = navigationTarget.navigationTargetObjects.Count;
            positionObject.name = positionPrefab.name + " Clone " + cloneCount;

            GameObject newItem = Instantiate(itemTemplate, itemParent);
            newItem.transform.SetAsLastSibling();
            newItem.SetActive(true);
            TMP_Text itemText = newItem.GetComponentInChildren<TMP_Text>();
            itemText.text = name;
        }
    }
}
```

Примітка. Саме цей код реалізує збереження та завантаження маркерів.

Стан усіх налаштувань зберігається за допомогою стандартного компонента Unity, PlayerPrefs. Збереження даних дозволяє користувачам зберігати свій прогрес та налаштування, навіть коли вони закрили додаток або вимкнули пристрій. Це покращує користувацький досвід та забезпечує більш зручний та ефективний роботу з додатком.

Висновок до розділу 3

Під час розробки кваліфікаційного проекту було затверджено комплексний план засобів розробки, які будуть використовуватись під час розробки проекту. Були визначені вимоги до технічного та програмного забезпечення, що дозволить забезпечити ефективну та стабільну роботу indoor застосунку. Детально описано процес реалізації indoor застосунку, включаючи використання засобів розробки, які дозволяють ефективно створювати, тестувати та налагоджувати програмний продукт. Були розглянуті критерії оцінки якості програмного продукту та процесу розробки, що дозволяє забезпечити високу якість готового продукту та ефективність розробки.

РОЗДІЛ 4. ТЕСТУВАННЯ

4.1. Функціональне тестування

Функціональне тестування є одним з найважливіших етапів тестування програмного забезпечення. Його основне завдання полягає у перевірці відповідності функціональних вимог ПЗ його реальним характеристикам. Під час функціонального тестування проводяться тестові сценарії, спрямовані на перевірку правильності роботи програмного продукту та його функціональності.

В результаті функціонального тестування формується таблиця з результатами, яка дозволяє оцінити стан програмного продукту та виявити можливі проблеми, які потрібно виправити перед випуском продукту на ринок. У таблиці з результатами можуть бути вказані такі параметри, як кількість виявлених помилок, час, необхідний для проведення тестування, показники продуктивності та якості роботи програмного продукту. Важливо пам'ятати, що функціональне тестування - це лише один з етапів тестування програмного забезпечення та не включає в себе інші види тестування, такі як тестування продуктивності, безпеки та інші.

Таблиця 4.1

Результати функціонального тестування

№	Назва тесту	Очікуваний результат	Отриманий результат	Тест пройдений ?
1	Відкриття списку випадаючого меню	При натисканні на кнопку-стрілку відкривається список всіх кабінетів.	При натисканні на кнопку-стрілку відкривається список всіх кабінетів.	+
2	Функціональність пошуку елементів випадаючого списку.	При введенні даних у поле пошуку, з'являється елемент, який відповідає введеним даним.	При введенні даних у поле пошуку, з'являється елемент, який відповідає введеним даним.	+
3	Функціональність кнопки "Очистити".	При натисканні кнопки "Очистити", поле вводу очищується, а покажчик маршруту зникає.	При натисканні кнопки "Очистити", поле вводу очищується, а покажчик маршруту зникає.	+
4	Розгортання мапи на весь екран.	При кліку на мапу, вона розгортається на весь екран, і з'являються кнопки збільшення та зменшення масштабування, а також кнопка повернення до головного екрану.	При кліку на мапу, вона розгортається на весь екран, і з'являються кнопки збільшення та зменшення масштабування, а також кнопка повернення до головного екрану.	+
5	Функціональність кнопок збільшення/зменшення	Кнопка плюс збільшує масштаб мапи, кнопка	Кнопка плюс збільшує масштаб мапи, кнопка мінус	+

	масштабування та кнопки повернення до головного екрану.	мінус зменшує масштаб мапи, а кнопка "назад" повертає користувача до головного екрану.	зменшує масштаб мапи, а кнопка "назад" повертає користувача до головного екрану.	
6	Функціональність кнопки для створення нової мітки.	Клік на кнопку "Додати" відкриває модальне вікно.	Клік на кнопку "Додати" відкриває модальне вікно.	+
7	Функціональність кнопок "Зберегти" та "Відмінити" в модальному вікні.	Після кліку на кнопку "Зберегти" модальне вікно закривається, створюється новий маркер, і з'являється push-повідомлення, створюється елемент у списку для видалення. Кнопка "Відмінити" закриває модальне вікно.	Після кліку на кнопку "Зберегти" модальне вікно закривається, створюється новий маркер, і з'являється push-повідомлення, створюється елемент у списку для видалення. Кнопка "Відмінити" закриває модальне вікно.	+
8	Функціональність кнопки для відкриття сторінки сканування QR-кодів.	Після кліку на кнопку відкривається вікно для сканування QR-кодів. При наведенні на QR-код, його зображення сканується, маркер змінює положення, а потім користувач	Після кліку на кнопку відкривається вікно для сканування QR-кодів. При наведенні на QR-код, його зображення сканується, маркер змінює положення, а потім користувач повертається на головну сторінку.	+

		повертається на головну сторінку.		
9	Функціональність кнопки для відкриття спику створених маркерів та їх видалення	Після кліку на кнопку відкривається вікно зі списком створених маркерів, і з допомогою чек-поінта можна вибрати потрібний елемент. Після натискання на кнопку видалення обраний елемент буде видалено з усіх відповідних місць системи.	Після кліку на кнопку відкривається вікно зі списком створених маркерів, і з допомогою чек-поінта можна вибрати потрібний елемент. Після натискання на кнопку видалення обраний елемент буде видалено з усіх відповідних місць системи.	+
10	Функціональність кнопки для відкриття вікна налаштувань.	Після натискання кнопки відкривається вікно налаштувань.	Після натискання кнопки відкривається вікно налаштувань.	+
11	Функціональність випадаючого списку для вибору мови інтерфейсу.	Клік на випадаючому списку розкриває його та відображає можливі варіанти мови. Після вибору мови, мова інтерфейсу змінюється відповідно до обраного варіанту.	Клік на випадаючому списку розкриває його та відображає можливі варіанти мови. Після вибору мови, мова інтерфейсу змінюється відповідно до обраного варіанту.	+
12	Функціональність зміни кольорової теми інтерфейсу.	Після кліку на кнопку змінюється кольорова тема інтерфейсу на темну, і при	Після кліку на кнопку змінюється кольорова тема інтерфейсу на темну, і при	+

		наступному кліку на кнопку тема змінюється на світлу.	наступному кліку на кнопку тема змінюється на світлу.	
13	Функціональність кнопки для вимкнення звуку в інтерфейсі.	Після кліку на кнопку, звукові ефекти не відтворюються	Після кліку на кнопку, звукові ефекти не відтворюються	+
14	Функціональність функції збереження та відновлення даних.	Після повторного запуску додатку, всі зміни, збережені в попередньому сеансі, відновлюються.	Після повторного запуску додатку, всі зміни, збережені в попередньому сеансі, відновлюються.	+

4.2. Usability тестування

Usability Test - це процес оцінки продукту або послуги шляхом тестування їх з представниками цільової аудиторії. Під час тестування учасники намагаються виконати типові завдання, а спостерігачі ретельно спостерігають, аналізують та роблять записи. Основна мета полягає в тому, щоб виявити будь-які проблеми зі зручністю використання, зібрати якісні та кількісні дані та визначити задоволеність учасника продуктом.

Шляхом такого тестування визначається зручність, зрозумілість та привабливість продукту для користувача. Це досягається за допомогою залучення користувачів з метою проходження тестових завдань і збору відгуків та висновків від них. Учасники тестування повинні виконувати різні завдання, які можуть включати пошук інформації, вибір певних опцій, навігацію по веб-сайту або додатку, взаємодію з інтерфейсом та інше. Наприклад, учасникам можуть пропонувати виконати наступні завдання:

- Знайти певний кабінет за допомогою пошукової функції;
- Вибрати потрібний кабінет в додатку;
- Пройти маршрут, використовуючи навігацію додатку;
- Натиснути на всі кнопки та інші елементи управління в додатку;
- Додати та видалити власний маркер;
- Відсканувати QR-код та перевірити правильність розпізнавання.

За допомогою результатів тестування можна зробити висновки про те, як користувачі взаємодіють з продуктом, які проблеми вони зустрічають та як їх можна вирішити. Це допомагає розробникам покращити продукт, забезпечити більшу задоволеність користувачів та підвищити ефективність його використання.

Висновок до розділу 4

Всі задачі були виконані тестувальниками успішно та без проблем. Респонденти відмітили простоту керування, хороше оформлення та складність проходження. Usability test пройдено успішно.

ВИСНОВКИ

У даній кваліфікаційній роботі були розглянуті питання, пов’язані з розробкою мобільного додатку з Indoor-навігацією та навігацією з використанням доповненої реальності AR. Для досягнення поставленої мети було виконано наступні завдання.

В першу чергу було проведено аналіз актуальності теми та огляд існуючих аналогів. Було виявлено, що існуючі методи Indoor навігації та навігації з використанням доповненої реальності AR є ефективними, однак вони не завжди забезпечують точну локацію користувача та найшвидший шлях до шуканого місцезнаходження об’єкта. У зв’язку з цим, було запропоновано власний підхід, який дозволяє спростити навігацію всередині приміщень та отримати точну локацію користувача. Для реалізації додатку було виконано огляд актуальних програмних середовищ та спроектовано ігровий додаток. Було розроблено алгоритмічні та програмні засоби для реалізації Indoor-навігації та навігації з використанням доповненої реальності AR, які дозволяють швидко організовувати маршрути до вибраного місцеположення будь-якого об’єкта всередині будівлі. У якості новизни роботи можна відзначити, що запропонований підхід дозволяє отримувати точну локацію користувача та розраховувати найшвидший шлях до шуканого місцезнаходження об’єкта, що робить Indoor-навігацію та навігацію з використанням доповненої реальності AR більш ефективними в порівнянні з існуючими аналогами.

Таким чином, результатом роботи є функціональний додаток, який дозволяє ефективно навігувати користувачів всередині будівель з використанням Indoor-навігації та навігації з використанням доповненої реальності AR.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Марущак Я.В. Розробка мобільного додатку на операційній системі IOS з indoor-навігацією та навігацією з використанням доповненої реальності AR всередині торгових центрів / Я. В. Марущак // XLIX Науковотехнічна конференція факультету

комп'ютерних систем і автоматики (2020)Вінницького національного технічного університету, – Вінниця : ВНТУ, 2020.

2) A Framework for Device-to-Device Augmented Reality Social Network/Haller Michael// IEEE International Conference on Pervasive Computing and Communications. - China(Beijing), 2010. P. 1–10.

3) Курчеева Г. И. Информационное и програмное обеспечение: учеб. пособие / Г. И. Курчеева, М. А. Бакаев, В. А. Хворостов. – Новосибирск: Издво НГТУ, 2018. – 107 с. – ISBN 978-5-7782-3500-7

4) Billinghurst M. Emerging Technologies of Augmented Reality: Interfaces and Design. / Billinghurst Mark, Thomas Bruce, 2007. - p. 209 – ISBN 978-1-5990-4066-0.

5) Patrick D. Archeoguide: System Architecture of a Mobile Outdoor Augmented Reality System. / Patrick D. Karigiannis, John N., 2009. - p. 345 – ISBN 9780769517810.

6) Щербаков С. Проектирование и оценка сборки в среде дополненной реальности / С. Щербаков – СПб.: Питер, 2018. – 352 с.: ил. – ISBN 978-5-4461-0484-0

7) Петроченков А. Дополненная реальность, наконец, становится реальностью / А. Петроченков. – СПб.: Питер, 2018. – 224 с.: ил. – ISBN 978- 5-496-02929-2

8) Augmented Reality Navigation: Killer Feature for Your Mapping App [Електронний ресурс]. Режим доступу: https://agilie.com/en/blog/augmente d-reality-navigation-killer-feature-for-your-mapping-app. – Назва з екрану.

9) Балуев Д. Виртуальная, дополненная и смешанная реальность / Денис Балуев. – М.: Альпина Паблишерз, 2010. – 287 с. – ISBN 978-5-9614- 1274-1

10) How augmented reality-based indoor navigation system works [Електронний ресурс]. Режим доступу: https://mobidev.biz/blog/augment ed-reality-indoor-navigation-app-developement-arkit. – Назва з екрану.

11) Augmented reality development: guide for business owners and managers [Електронний ресурс]. Режим доступу: https://mobidev.biz/blog/augm ented-reality-development-guide. – Назва з екрану.

12) How Augmented Reality Works [Електронний ресурс]. Режим доступу: https://computer.howstuffworks.com/augmented-reality.htm#pt2 – Назва з екрану.

13) Mobile Augmented Reality [Електронний ресурс]. Режим доступу: http://web.cs.wpi.edu/~gogo/courses/imgd5100_2012f/papers/Hollerer_ AR_2004.pdf. – Назва з екрану.

14) Heffelfinger D. Development with ARkit / David R. Heffelfinger – Birmingham: Packt Publishing, 2015. – 366 p.

15) Santosh Kumar K. What are augmented reality markers?/ Santosh Kumar K. – New Delhi: Tata McGraw-Hill, 2009. – 513 p.

16) Hemrajani A. Getting Started with iBeacon/ Anil Hemrajani, Scott W. Ambler and Rod Johnson – USA: Sams Publishing, 2016. – 318 p.