МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

МОБІЛЬНИЙ ІГРОВИЙ AR-ЗАСТОСУНОК З ВИКОРИСТАННЯМ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Євгеній ТВЕРДИЙ**

«__» _____ 20__ р.

Керівник роботи
канд. техн. наук,
доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступінь	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну магістерську роботу здобувача вищої освіти

Твердого Євгенія Олеговича

1. Тема кваліфікаційної роботи «Мобільний ігровий AR-застосунок з використанням алгоритмів машинного навчання» затверджена наказом ректора ЧНУ ім. Петра Могили № _220_ від «_4_» _вересня_ 2024 р.
2. Строк представлення кваліфікаційної роботи «_20_» _грудня_ 2024 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні: розроблений мобільний ігровий AR-застосунок
4. Перелік питань, що підлягають розробці: проаналізувати існуючі рішення у сфері мобільних ігрових AR застосунків, дослідити алгоритми машинного навчання, придатних для інтеграції в AR застосунки, розробити концепції мобільного ігрового AR застосунку, реалізувати прототип застосунку, провести тестування та оптимізацю функціональності застосунку

5. Перелік графічних матеріалів: презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Дата видачі завдання « ____»_____20_____р

Календарний план виконання кваліфікаційної роботи

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема:_____«Мобільний ігровий AR-застосунок з використанням алгоритмів машинного навчання»_____
_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	1.09.2024р.	4.09.2024р.	виконано
2.	Огляд літератури за темою роботи	5.09.2024р.	6.09.2024р.	виконано
3.	Складання календарного плану КМР	7.09.2024р.	7.09.2024р.	виконано
4.	Аналіз предметної області	8.09.2024р.	15.09.2024р.	виконано
5.	Розробка проєктних рішень	16.09.2024р.	20.09.2024р.	виконано
6.	Моделювання та конструювання ПЗ	26.09.2024р.	9.10.2024р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування	12.10.2024р.	23.11.2024р.	виконано
8.	Відгук керівника КМР	25.11.2024р.	25.11.2024р.	виконано
9.	Оформлення КМР та презентації	26.11.2024р.	29.11.2024р.	виконано
10.	Попередній захист	2.12.2024р.	2.12.2024р.	виконано
11.	Рецензування	10.12.2024р.	10.12.2024р.	виконано
12.	Завершення оформлення КМР та презентації	11.12.2024р.	11.12.2024р.	виконано
13.	Захист кваліфікаційної роботи	19.12.2024р.	20.12.2024р.	

Здобувач _____ Євгеній ТВЕРДИЙ

«__» _____ 20__ р.

Керівник роботи

<наук. ступінь>,

<вчене звання> _____ Євген ДАВИДЕНКО

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Мобільний ігровий AR-застосунок з використанням алгоритмів машинного навчання»

Здобувач 608м гр.: Твердий Євгеній Олегович

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Кваліфікаційна робота магістра присвячена дослідженню аналогів та розробці мобільного ігрового AR-застосуноку з використанням алгоритмів машинного навчання.

Метою кваліфікаційної роботи є розробка мобільного ігрового AR-застосунку, що використовує алгоритми машинного навчання.

Завдання роботи:

- аналіз існуючих рішень у сфері мобільних ігрових AR застосунків;
- вивчення алгоритмів машинного навчання, придатних для інтеграції в AR застосунки;
- розробка концепції мобільного ігрового AR застосунку;
- програмна реалізація прототипу застосунку;
- тестування та оптимізація функціональності застосунку;
- оцінка користувацького досвіду та залученості користувачів.

Об’єктом кваліфікаційної роботи є технології доповненої реальності (AR) та машинного навчання, які використовуються у мобільних ігрових застосунках.

Предметом кваліфікаційної роботи є інструменти, методи та процеси інтеграції алгоритмів машинного навчання в AR застосунки для підвищення їх функціональності та користувацького досвіду.

Пояснювальна записка кваліфікаційної магістерської роботи складається з вступу, чотирьох розділів, висновків, додатків та переліку джерел посилання.

У вступі визначається актуальність, науково-практичне значення обраної теми, мету і завдання роботи, а також об’єкт та предмет дослідження.

У першому розділі описується аналітична частина, тобто огляд існуючих аналогів та те завдяки чому було сформовано розуміння предметної області.

У другому розділі описується процес розробки проєктних рішень, які допомагають задовольнити вимоги до програмного забезпечення. Цей процес включає в себе моделювання об'єкту та предмету дослідження, а також створення функціональних та інформаційних моделей програмного забезпечення.

У третьому розділі описується процес розробки, вибір мов програмування, вибір рушія, розробка концепції гри, її геймплей, діаграм та інтерфейсу.

У четвертому розділі демонструється проведена з розробки робота.

У висновках проводиться аналіз проведеної роботи та отриманих результатів.

КМР викладена на 81 сторінку, вона містить 4 розділи, 59 ілюстрацій, 4 таблиці, 18 джерел в переліку посилань.

Ключові слова: Unity, розробка ігор, C#, AR, ML, Python

ASTRACT
of the Master’s Thesis

"Mobile gaming AR app with using machine learning algorithms"

Student of group 608m: Tverdyi Yevhenii Olegovich

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor

Davydenko Y. O.

The master's qualification work is devoted to the research of analogues and the development of a mobile gaming AR-application using machine learning algorithms.

The purpose of the qualification work is to develop a mobile gaming AR application that uses machine learning algorithms.

Tasks of the work:

– analysis of existing solutions in the field of mobile gaming AR applications;
– study of machine learning algorithms suitable for integration into AR applications;
– development of the concept of a mobile gaming AR application;
– software implementation of the application prototype;
– testing and optimizing the functionality of the application;
– assessment of user experience and user engagement.

The object of the qualification work is augmented reality (AR) and machine learning technologies used in mobile gaming applications.

The subject of the qualification work is the tools, methods and processes of integrating machine learning algorithms into AR applications to improve their functionality and user experience.The explanatory note of the master's thesis consists of an introduction, four chapters, conclusions, appendices and a list of reference sources.

The introduction defines the relevance, scientific and practical significance of the chosen topic, the purpose and task of the work, as well as the object and subject of the research.

The first chapter describes the analytical part, i.e. the review of existing analogues and the way in which the understanding of the subject area was formed.

The second chapter describes the process of developing project solutions that help meet software requirements. This process includes the modeling of the object and subject of research, as well as the creation of functional and informational software models.

The third chapter describes the development process, the choice of a programming language, the choice of an engine, the development of the game concept, its gameplay, diagrams and interface.

The fourth chapter demonstrates the development work.

In the conclusions of the results of the analysis of the work carried out and the results obtained.

Master's qualification thesis is laid out on 81 pages, it contains 4 chapters, 59 illustrations, 4 tables, 18 sources in the list of references.

Keywords: Unity, розробка ігор, C#, AR, ML, Python

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4

ВСТУП 5

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Огляд предметної області 7

1.2 Аналіз аналогів 11

1.3 Специфікація вимог до програмного забезпечення 16

Висновки до розділу 1 18

2 АНАЛІЗ СИСТЕМИ 19

2.1 Створення сценаріїв 19

2.2 Створення діаграм сценаріїв використання 21

2.3 Побудова діаграм взаємодії (послідовності та кооперації) 23

2.4 Побудова діаграм станів 26

2.5 Створення мокапів 29

Висновки до розділу 2 35

3 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ТА ОГЛЯД СТЕКУ ЗАДІЯНИХ ТЕХНОЛОГІЙ 36

3.1 Розробка UML-діаграм 36

3.1.1 Діаграма класів 36

3.1.2 Діаграми компонентів та розгортання 38

3.1.3 Діаграма пакетів 42

3.2 Огляд технологій 43

3.2.1 Мова програмування C# 44

3.2.2 Мова програмування Python та середовище розробки Jupiter 46

3.2.3 Ігровий рушій 49

3.2.4 Графічний редактор 50

Висновки до розділу 3 52
4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ГРИ 53
4.1 Створення інтерфейсу гри 53
4.2 Створення героя гри, магазину та його предметів 56
4.3 Створення логіки генерації квестів 58
4.4 Створення та тестування моделі розпізнавання об’єктів 62
4.4.1 Вибір моделі 62
4.4.2 Набір даних та підготовка моделі 64
4.5 Підключення моделі 68
4.6 Тестування гри 72
Висновки до розділу 4 75
ВИСНОВКИ 76
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 78
Додаток А Код демонстрації роботи моделі 80
Додаток Б Апробація кваліфікаційної роботи 81

ПЕРЕЛІК СКОРОЧЕНЬ

КМР – кваліфікаційна магістерська робота

ОС – операційна система

ПЗ – програмне забезпечення

AR – augmented reality

UML – unified modeling language

ВСТУП

Розвиток мобільних ігрових AR-застосунків з використанням алгоритмів машинного навчання є надзвичайно актуальним через стрімке зростання популярності AR технологій та їх застосування у різних сферах, від ігор до освіти та бізнесу. AR (доповнена реальність) відкриває нові можливості для створення інтерактивних середовищ, де користувачі можуть взаємодіяти з віртуальними об'єктами в реальному світі. Завдяки машинному навчанню, ці взаємодії можуть бути не лише реалістичними, але й інтелектуальними, адаптуючись до потреб і дій користувача, а також прогнозуючи його наступні кроки.

Машинне навчання дозволяє створювати більш інтерактивні та персоналізовані досвіди для користувачів, підвищуючи залученість та надаючи нові можливості для інновацій. Наприклад, завдяки глибинному навчанню, моделі можуть аналізувати велику кількість даних про поведінку користувачів, що дозволяє прогнозувати їхні вподобання та налаштовувати контент на основі цих прогнозів. Це може бути корисним для ігрових додатків, де динамічно змінюється сюжет або середовище, реагуючи на дії гравця.

Використання машинного навчання в AR застосунках може значно покращити точність та ефективність взаємодії з користувачем, забезпечуючи адаптацію контенту до його поведінки та переваг. Це відкриває шлях для розробки нових методів навчання, реклами, та інших прикладних рішень, що мають велике значення для промисловості та освіти. У сфері освіти такі застосунки можуть адаптувати навчальний матеріал під рівень учня, надаючи йому більш персоналізований підхід. У бізнесі це може проявлятися в адаптивних рекомендаційних системах або маркетингових кампаніях, які використовують AR для створення унікальних візуальних та інтерактивних досвідів.

Крім того, інтеграція алгоритмів комп'ютерного зору з машинним навчанням дозволяє покращити здатність AR систем розпізнавати об'єкти, людей або навколишнє середовище, що розширює можливості застосування цих технологій у реальному світі. Це може бути корисним у таких галузях, як

медицина (для хірургічних операцій або навчання), архітектура (візуалізація проектів у реальному середовищі), роздрібна торгівля (віртуальні примірочні) та багато інших.

Об’єктом кваліфікаційної роботи є технології доповненої реальності (AR) та машинного навчання, які використовуються у мобільних ігрових застосунках.

Предметом кваліфікаційної роботи є інструменти, методи та процеси інтеграції алгоритмів машинного навчання в AR застосунки для підвищення їх функціональності та користувацького досвіду.

Метою кваліфікаційної роботи є розробка мобільного ігрового AR-застосунку, що використовує алгоритми машинного навчання.

Завдання роботи:

- аналіз існуючих рішень у сфері мобільних ігрових AR застосунків;
- вивчення алгоритмів машинного навчання, придатних для інтеграції в AR застосунки;
- розробка концепції мобільного ігрового AR застосунку;
- програмна реалізація прототипу застосунку;
- тестування та оптимізація функціональності застосунку;
- оцінка користувацького досвіду та залученості користувачів.

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання 2024», Миколаїв, 06-10 листопада, 2024 р. (Додаток Б).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Відеоігри – це справжній феномен сучасності, який не лише впливає на культуру, але й активно інтегрується у різні сфери життя за межами розваг. Сучасні технології, такі як доповнена реальність (AR), машинне навчання та штучний інтелект, значно розширюють горизонти використання ігрових механік.

Відеоігри не мають кордонів. Гравці з різних країн можуть зустрічатися в онлайн-світі, об’єднуватися в команди та вирішувати завдання разом. Це створює глобальну спільноту, яка об’єднує людей незалежно від їхнього місця проживання. Онлайн-ігри дозволяють створювати зв'язки між людьми на міжнародному рівні, що особливо важливо у світі, де цифрові технології грають ключову роль у комунікації та співпраці.

Від епічних пригод і фантастичних світів до спортивних симуляторів та головоломок – відеоігри пропонують безмежний вибір. Кожен може знайти щось за своїм смаком. Окрім традиційних жанрів, з'являються нові формати, які поєднують у собі елементи освіти, мистецтва та навіть терапії. Наприклад, відеоігри вже використовуються у медичній галузі для реабілітації пацієнтів або тренування фахівців.

Завдяки постійному росту обчислювальної потужності, графіки та інтерактивності, ігри стають все більш захоплюючими. Від віртуальної реальності до штучного інтелекту – технології розвиваються, а ігри з ними. Сучасні технології, такі як доповнена реальність (AR), відкривають нові можливості. AR інтегрується не тільки у відеоігри, але й у медицину, навчання, архітектуру та інші галузі. Наприклад, AR застосовують у хірургічних операціях, щоб візуалізувати внутрішні органи пацієнтів у реальному часі, або у навчальних програмах, щоб зробити процес засвоєння інформації більш інтерактивним і наочним.

Машинне навчання і штучний інтелект (AI) також відіграють важливу роль у відеоіграх та поза ними. У іграх AI дозволяє створювати складних і

реалістичних супротивників, що можуть адаптуватися до стилю гравця. Але ці технології знаходять застосування далеко за межами ігрової індустрії. Наприклад, в автономних автомобілях використовуються алгоритми, які спочатку були розроблені для оптимізації ігрових процесів. AI допомагає також у фінансових технологіях, медицини, маркетингу та багатьох інших сферах.

Відеоігри не завжди є суто розвагою – вони стають важливим інструментом для соціалізації. Граючи в онлайн-режимі, ми спілкуємося з іншими гравцями, об’єднуємося в команди та вирішуємо завдання разом. Це створює нові дружби та спільноти, які можуть підтримуватися навіть поза межами ігор. До того ж, через ігри можна навчатися колективній роботі, розвивати комунікативні навички та навички прийняття рішень.

Відеоігри – це не тільки розвага, але й спосіб розслабитися та відволіктися від повсякденних турбот. Вони дозволяють нам відчути себе героями, дослідниками чи воїнами, забуваючи про реальний світ. Проте технології доповненої реальності і віртуальної реальності можуть зробити цей досвід ще більш насиченим та реалістичним, дозволяючи людям взаємодіяти з віртуальними світами у нові способи. Також відеоігри дедалі частіше використовуються для зняття стресу, психотерапії та підтримки психічного здоров'я.

Кількість мобільних геймерів постійно зростає. За даними Pew Research Center, до 2030 року кількість мобільних геймерів може збільшитися від 2,22 до 2,9 мільярда[1].

У 2022 році користувачі завантажили близько 89,74 мільярда мобільних ігор і витрачали в середньому 5,3 години на мобільні застосунки. Найбільше мобільних геймерів проживає в Азії (48%), Європі (20%), Латинській Америці (11%) та Північній Америці (7%). Китай, США та Японія – найбільші ринки мобільних ігор.

Для прикладу розглянемо найпопулярнішу AR гру Pokemon Go[2] (рис. 1.1–1.2). Ця гра від Niantic стала справжнім проривом в світі AR який світ побачив 6 липня 2016 року. Гравці мандрують реальними місцями, ловлячи віртуальних покемонів. Вона залучила мільйони гравців по всьому світ.

Рисунок 1.1 – AR гра Pokemon Go

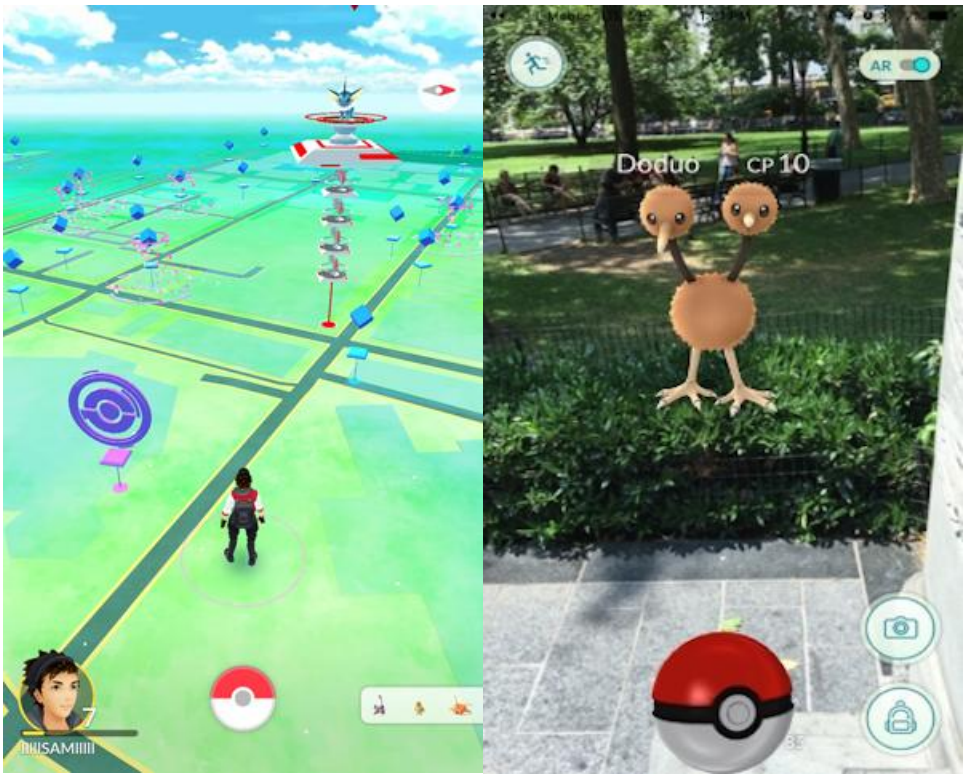


Рисунок 1.2 – Скріншоти геймплею гри

Мільйонам гравців гра Pokemon Go сподобалась саме через злиття реального і віртуального світів. Pokemon Go майстерно поєднує реальний світ з цифровим, дозволяючи гравцям взаємодіяти з об'єктами у віртуальному просторі, відвідуючи реальні локації у пошуках покемонів. Це створює унікальний досвід,

де межі між реальністю та вигаданим світом стираються, і, на відміну від традиційних відеоігор, гравці не залишаються вдома перед екраном, а стають частиною активного процесу дослідження навколишнього середовища.

Одним із найбільших нововведень Pokemon Go є використання доповненої реальності (AR), яка дозволяє гравцям бачити покемонів через екран смартфону, ніби вони знаходяться прямо перед ними в реальному світі. Це змінює традиційний підхід до відеоігор, перетворюючи її на інтерактивну пригоду, де гравці мають активно рухатися, відвідувати парки, історичні місця та навіть туристичні визначні пам'ятки. Така інтеграція з реальним світом не тільки робить гру цікавішою, але й стимулює дослідження навколишнього середовища та фізичну активність.

Pokemon Go сприяє об'єднанню людей через спільний інтерес до гри. Гравці зустрічаються у різних місцях, щоб зловити рідкісних покемонів, обмінюються порадами, досвідом та навіть предметами в грі. Це сприяє створенню спільноти, яка виходить за рамки простої гри. Відбуваються організовані зустрічі, спеціальні івенти та навіть глобальні змагання, що об'єднують мільйони людей по всьому світу. Така соціальна взаємодія не тільки розширює коло друзів, але й формує відчуття спільності та належності до глобальної групи з єдиними інтересами.

Гра використовує персонажів культової франшизи, яка радує шанувальників уже багато років, що викликає потужне почуття ностальгії. Гравці, які колись захоплювалися аніме-серіалом чи колекційними картами, тепер мають можливість знову поринути у світ покемонів, але тепер в інтерактивному, сучасному форматі. Вони можуть зустріти улюблених персонажів, таких як Пікачу чи Чарізард, у своєму рідному місті або на прогулянці в парку, що робить досвід ще більш захопливим. Це дозволяє гравцям не лише насолоджуватися новими пригодами, але й відчути приємні емоції, пов'язані з дитячими спогадами.

Однією з унікальних особливостей Pokemon Go є те, що вона мотивує гравців виходити на вулицю, рухатися і досліджувати світ. Гра поєднує розвагу із підтриманням активного способу життя, адже гравцям потрібно ходити пішки, щоб знайти нових покемонів, відвідати спеціальні покестопи чи взяти участь у

битвах у тренувальних залах. Крім того, гра заохочує щоденну активність, пропонуючи спеціальні нагороди за досягнення певної кількості кроків або відвідування нових місць. У цьому сенсі Pokemon Go відіграє важливу роль у популяризації здорового способу життя, перетворюючи прогулянки та фізичну активність на веселу ігрову подорож.

Гра пропонує безліч завдань і викликів, які постійно тримають гравців у напрузі. Пошук рідкісних покемонів, участь у гімнастичних змаганнях, битви з іншими гравцями та спеціальні івенти – усе це робить гру захопливою та динамічною. Завдяки постійному оновленню контенту, новим сезонним завданням і подіям, гравці завжди мають нові цілі та стратегії. Це не дозволяє грі стати одноманітною і постійно підтримує інтерес до неї. Крім того, змагання між командами (Instinct, Mystic, Valor) додають додаткового елементу суперництва і колективної гри, що також зміцнює почуття спільноти.

Рівень популярності та значущості гри підтверджується не лише кількістю гравців, а й визнанням на міжнародній арені. У 2016 році Pokemon Go стала переможцем The Game Award у номінації «Найкраща гра для мобільних пристроїв/портативних гральних консолей». Це свідчить про інноваційність та якість гри, яка не тільки сподобалась мільйонам користувачів, але й отримала високу оцінку професіоналів ігрової індустрії.

Рівень популярності та значущості гри звісно визначають її нагороди, так у 2016 році гра стала переможцем The Game Award у номінації «Найкраща гра для мобільних пристроїв/портативних гральних консолей».

1.2 Аналіз аналогів

Pokemon Go

Таблиця 1.1 – Характеристика гри **Pokemon Go**

Назва характеристики	Опис
Назва	Pokemon Go

Кінець таблиці 1.1

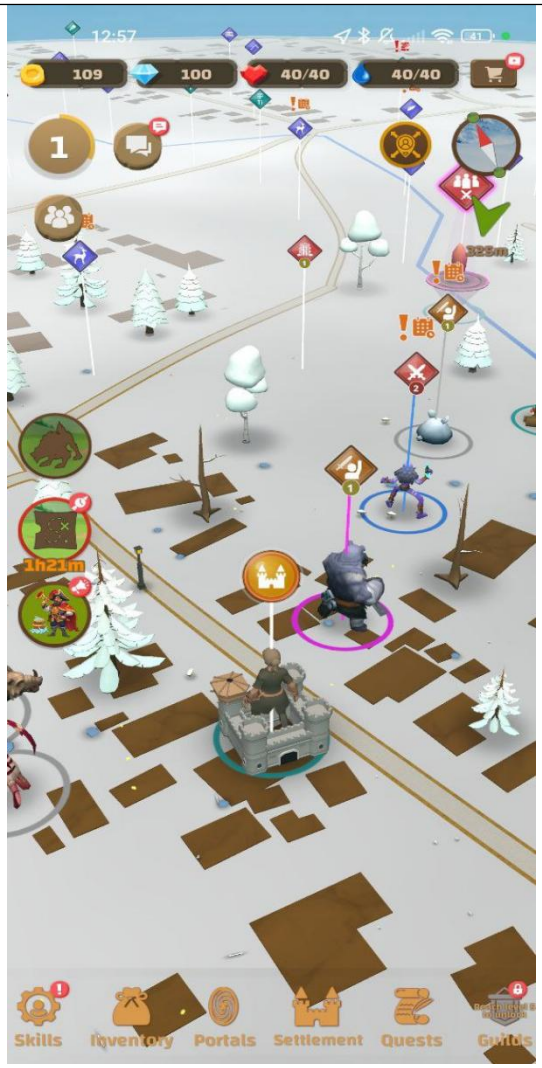
Розробник	Niantic
Архітектура	iOS, Android
Мова реалізації	C#
Перелік функцій, характеристик	1) сканувати місцевість; 2) знаходити нових покемонів шляхом ходьби в якісь локації в реальному світі (парк, ліс, вулиця, двір); 3) змагатися з іншими гравцями; 4) керувати налаштуваннями геймплею, відео та аудіо; 5) 3D.
Аналіз переваг та недоліків	Переваги: 1) багатокористувацький режим; 2) приємна графіка; 3) захопливі баталії. Недоліки: 1) не доступно в Україні; 2) інтенсивне використання GPS та камери швидко розряджає батарею; 3) можливі ризики під час гри в незнайомих або небезпечних місцях.
Приклад зображень	 Рисунок 1.3 – Зображення геймплею
Джерело інформації[2]	https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=uk

Magic Streets: the GPS realm

Таблиця 1.2 – Характеристика Magic Streets: the GPS realm

Назва характеристики	Опис
Назва	Magic Streets: the GPS realm
Розробник	Gearbox Publishing
Архітектура	iOS, Android
Мова реалізації	Не відома
Перелік функцій, характеристик	1) сканувати місцевість; 2) знаходити нових ворогів шляхом ходьби в якісь локації в реальному світі (парк, ліс, вулиця, двір); 3) змагатися з іншими гравцями; 4) прокачувати спорядження; 5) керувати налаштуваннями геймплею, відео та аудіо; 6) 3D.
Аналіз переваг та недоліків	Переваги: 1) багатокористувацький режим; 2) яскрава 3D графіка; 3) можливість будувати власне місто та захищати його; 4) захопливі баталії та завдання. Недоліки: 1) відсутність української; 2) може бути складною для новачків через велику кількість функцій.

Кінець таблиці 1.2

Приклад зображень	 Рисунок 1.4 – Зображення геймплею
Джерело інформації[3]	https://play.google.com/store/apps/details?id=com.builditgames.magicstreets&hl=uk

Cosmic Frontline AR

Таблиця 1.3 – Характеристика гри Cosmic Frontline AR

Назва характеристики	Опис
Назва	Cosmic Frontline AR
Розробник	Hofli Limited
Архітектура	iOS, Android
Мова реалізації	Не відома

Кінець таблиці 1.3

Перелік функцій, характеристик	1) сканування кімнати; 2) війни між планетами; 3) проходити рівні; 4) керувати налаштуваннями геймплею, відео та аудіо; 5) 3D.
Аналіз переваг та недоліків	Переваги: 1) захопливі баталії в локальному кооперативі; 2) купа персонажів та зброї; 3) глибокий стратегічний геймплей з можливістю керувати космічними флотами. Недоліки: 1) замало ігрового контенту; 2) потребує потужного пристрою для плавної роботи; 3) може швидко набриднути через обмежену кількість місій та завдань; 4) деякі функції можуть бути доступні лише за додаткову плату.

Приклад зображень	 Рисунок 1.5 – Зображення геймплею
Джерело інформації[4]	https://play.google.com/store/apps/details?id=com.hofli.cosmicfrontline&hl=uk

Кожна гра має унікальні особливості, які приваблюють певну аудиторію:

– Pokemon GO ідеально підходить для тих, хто шукає активність і соціальні взаємодії;

– Magic Streets: the GPS realm сподобається фанатам глибоких RPG-елементів та локальної адаптації;

– Cosmic Frontline AR – вибір для любителів космосу та стратегій, які цінують графічну якість.

Ці ігри добре ілюструють різноманітність використання AR та GPS у сучасному геймінгу.

1.3 Специфікація вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням гри є принесення задоволення гравцям, покращення настрою та вихід молоді на вулиці.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення гри буде використано ігровий рушій Unity Engine.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 23.11.2024р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Цей застосунок призначений для того, щоб користувач міг проводити свій вільний час з дозвілля та відпочинку, використовуючи його.

Характеристика користувачів

Основні характеристики користувачів: наявність смартфону.

Загальні обмеження

Обмежень немає.

ФУНКЦІЇ СИСТЕМИ

Функція персональних рекордів

Опис функції

Функція покращення проходження викликає у гравця більше почуття азарту та задоволення від гри.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

Основним джерелом вхідної інформації є користувач та камера смартфону.

Вимоги до способів організації, збереження та ведення інформації

Усі данні зберігаються у пам’яті смартфона.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

– ОС: Android 9 і вище

– Оперативна пам’ять: 4 GB

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура програмної системи складається лише з клієнтської частини.

Системне програмне забезпечення

Гра розроблена на платформі Unity з використанням мови програмування C#.

Мережне програмне забезпечення

Для створення застосунку було використано ОС Windows 11.

Мова і технологія розробки ПЗ

Гра була розроблена на платформі Unity з використанням мови програмування C#.

Інтерфейс користувача

Інтерфейс має задовольняти усі вимоги UI та UX дизайну, задля легкого розуміння користувачем роботи системи.

Апаратний інтерфейс

Апаратним інтерфейсом є смартфон користувача, з операційною системою Android 9, або вище.

Програмний інтерфейс

У ході розробки було використано дві категорії Unity Scripting API :

UnityEditor (Animations, Events тощо) та UnityEngine (Analytics, Audio тощо).

Комунікаційний протокол

Відсутній.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Гра є доступною для будь-якого користувача, за умови наявності у користувача апаратного інтерфейсу.

Супроводжуваність

Гра не потребує супроводжуваності.

Переносимість

Програмне забезпечення може працювати на ОС Android (9 версія та вище).

Продуктивність

Продуктивність роботи ПЗ залежить від характеристик смартфону.

Надійність

Відсутня.

Безпека

Відсутня.

ІНШІ ВИМОГИ

Усі вимоги сформовано та описано вище, доповнення не вимагається.

Висновки до розділу 1

У першому розділі було проведено дослідження популярності AR відеоігор у всьому світі. Для створення кращої гри проаналізовано аналоги від інших розробників, виявлено їх переваги та недоліки, розглянуто архітектуру та мову реалізації. В результаті цього аналізу було встановлено специфікації вимог до програмного забезпечення.

Отже, можна зробити висновок, що розробка якісної гри – це завдання, яке вимагає великих зусиль, навіть від цілої команди розробників. Але саме завдяки такому дослідженню можна буде уникнути багатьох помилок та вийде створити щось дійсно цікаве!

2 АНАЛІЗ СИСТЕМИ

2.1 Створення сценаріїв

Коротка форма написання usecase для задачі «Виконання квестів у грі»:

Користувач захотів зіграти в гру та та виконати квести, тож він запускає її на своєму смартфоні. Гра запускається та відображає на екрані головне меню гри. Користувач натискає на кнопку «Квести» після чого на екран виводиться вибір варіантів квестів. Користувач може обрати один з варіантів квестів та перейти до їх виконання.

Поверхнева форма написання usecase для задачі «Виконання квестів у грі»:

Головний сценарій (успішний):

Користувач захотів зіграти в гру та та виконати квести, тож він запускає її на своєму смартфоні. Гра запускається та відображає на екрані головне меню гри. Користувач натискає на кнопку «Квести» після чого на екран виводиться вибір варіантів квестів. Користувач може обрати один з варіантів квестів та перейти до їх виконання.

Альтернативні сценарії:

1. Гравець не може зіграти в гру, адже на смартфоні не було надано доступ до камери та геолокації.

2. Гравець не може зіграти в гру, адже смартфон не проходить по мінімально необхідним вимогам до характеристик для гри.

Повна форма написання usecase для задачі «Виконання квестів у грі»:

Таблиця 2.1 – Usecase

Use section	**Comment**
Use Case Name	Виконання квестів у грі
Scope	Гра для смартфонів
Level	Пограти в гру на смартфоні
Primary Actor	Користувач

Продовження таблиці 2.1

Stakeholders and interests	1. Користувач. Зацікавлений в тому аби запустити та зіграти в гру
Preconditions	1. Користувач тримає розблокований смартфон в руках
Main Success Scenario	1. Користувач знаходить гру на робочому столі смартфона 2. Користувач запускає гру 3. Гра запускається 4. Гра відображає на екрані смартфону головне меню 5. Користувач натискає на кнопку «Квести» 6. Гра відображає список з варіантів квестів на екрані смартфона 7. Користувач натискає на обрану категорію квестів 8. Система обробляє натискання та відображає список завдань 9. Користувач натискає на кнопку «Виконати завдання» 10. Система обробляє натискання та вмикає камеру смартфону
Extensions	1. Користувач вирішив змінити налаштування в головному меню 1) Користувач натискає на кнопки вмик/вимк звуку чи вібрації 2) Система обробляє натискання та змінює налаштування

Кінець таблиці 2.1

	2. Користувач не зміг знайти необхідний предмет для виконання квесту 1) Користувач натискає на кнопку «Назад» 2) Система обробляє натискання та повертає меню вибору категорі1 квесту 3. Користувач захотів змінити зовнішній вигляд свого персонажа 1) Користувач натискає на кнопку «Магазин» 2) Система обробляє натискання та відкриває список доступних для купівлі предметів
Special Requirements	Смартфон що проходить мінімально необхідні вимоги для запуску гри
Frequency of Occurrence	Гра може працювати майже безкінечно
Miscellaneous	Збереження даних гри

2.2 Створення діаграм сценаріїв використання

Діаграми варіантів використання (use case diagrams) служать для візуалізації взаємодії між системою та її користувачами, відображаючи, як саме вони використовують доступний функціонал[5]. На діаграмі актори, що представляють користувачів або інші зовнішні системи, зображуються у вигляді фігури людини. Кожен варіант використання, що зображує певну дію або сценарій взаємодії, подається у формі еліпса. У таких варіантах використання описуються можливі дії користувачів та відповідні реакції системи на ці дії, що дозволяє краще зрозуміти функціональні можливості системи та її поведінку під час взаємодії з різними акторами.

Розглянемо діаграму сценаріїв використання гри що проєктується (рис. 2.1).

Рисунок 2.1 – Діаграма сценаріїв використання

Опис елементів діаграми:

1) актори:

– гравець (player): це основний користувач системи, який взаємодіє з застосунком;

2) сценарії використання:

– початок квестів (start of quests): гравець починає виконувати квести;

– вибір квесту (select quest): гравець обирає конкретний квест для виконання;

– відкриття камери (open camera): гравець активує функцію камери в застосунку;

– зробити фото (take a photo): гравець робить фотографію за допомогою камери;

– купівля товару (buy an item): гравець вибирає товар для покупки в магазині;

– налаштування звуку (set sound settings): гравець встановлює параметри звуку (включити або вимкнути);

– налаштування вібрації (set vibration settings): гравець налаштовує параметри вібрації (включити або вимкнути);

– скасування квесту (cancel the quest): гравець відміняє виконання квесту;

– повідомлення про помилку (error message): система відображає повідомлення про помилку;

– статус завершення квесту (quest completed / quest not completed): гравець отримує інформацію про статус виконання квесту;

3) розширення та включення:

– діаграма також показує відносини розширення та включення між сценаріями.

На даній діаграмі зображено варіанти використання гри. Гравець може змінити налаштування звуку та вібрації. Весь ігровий процес зосереджений на виконання квестів, тому спочатку гравець має відкрити панель категорій квестів, обрати одну з чотирьох категорій, і там вже обрати квест який йому сподобається, основні квести в грі це показ якогось предмету на камеру, після чого гра визначає правильний предмет показано, чи ні та повідомляє про це гравця, і або дає винагороду за пройдений квест, або пропонує показати правильний предмет. Також гравець може купувати предмети для свого персонажа в ігровому магазині.

2.3 Побудова діаграм взаємодії (послідовності та кооперації)

Діаграми послідовності є одним із засобів опису сценаріїв використання системи. Вони дозволяють на ранніх етапах розробки визначити взаємодію між компонентами та потоки повідомлень, що проходять між ними. Це сприяє подальшій трансформації цих компонентів у класи (об'єкти) та відповідні методи. Крім того, діаграми послідовності допомагають ідентифікувати події (Actions), які система повинна підтримувати та обробляти [6].

Діаграми послідовностей належать до категорії діаграм взаємодії в UML і забезпечують можливість описати поведінкові аспекти системи, зосереджуючись на взаємодії об'єктів у часі. Основна мета цих діаграм – зобразити часові аспекти передачі та прийому повідомлень між об'єктами. Кожен об'єкт на діаграмі представлений у вигляді прямокутника, що розташований зверху вертикальної пунктирної лінії, відомої як "лінія життя" об'єкта. Лінія життя зображає життєвий цикл об'єкта під час його взаємодії з іншими об'єктами системи.

Розглянемо діаграму послідовності для гри що проєктується (рис. 2.2).

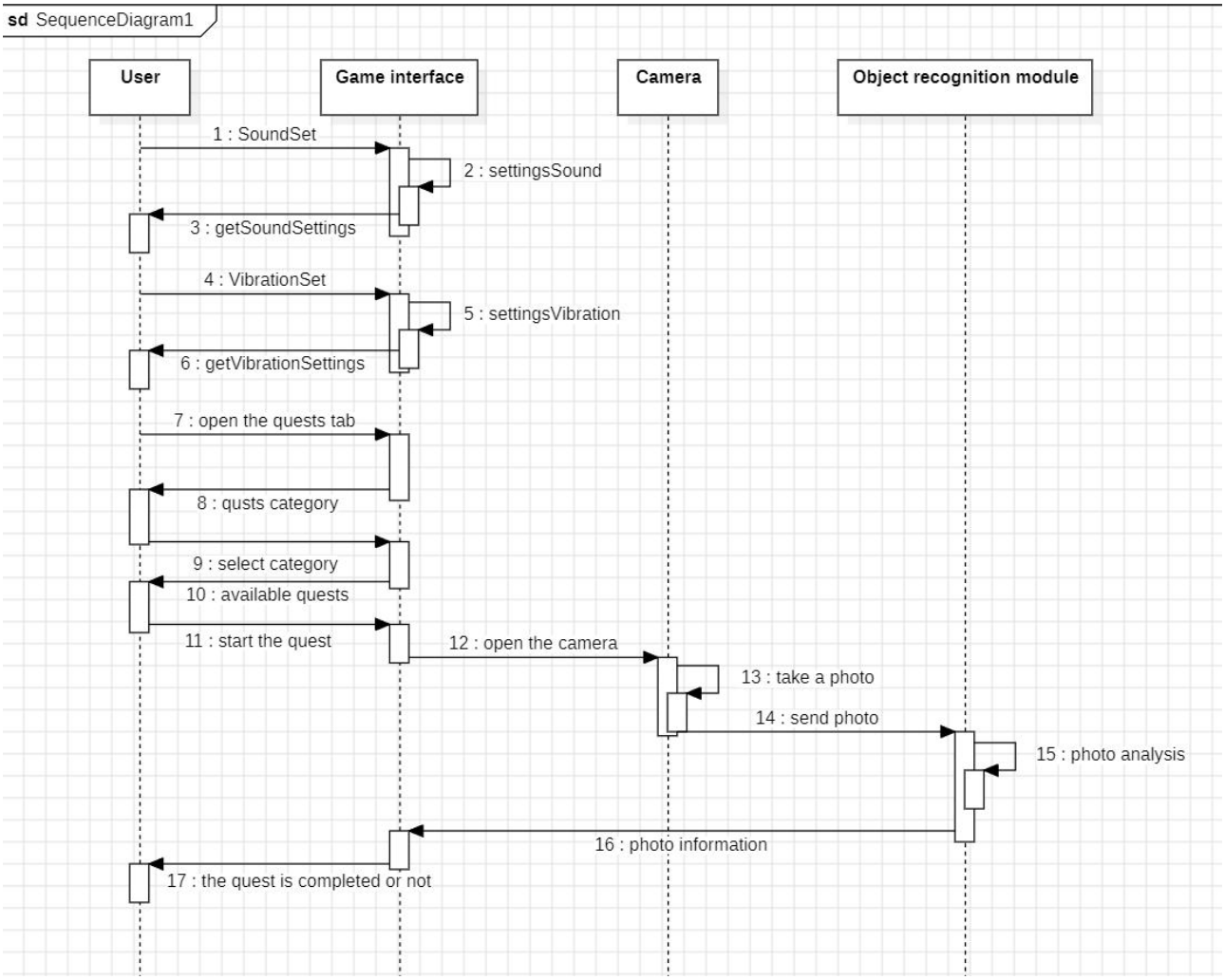


Рисунок 2.2 – Діаграма послідовності

На даній діаграмі показано взаємодію гравця з різними елементами гри, таких як головне меню, та саме квести гри.

Розглянемо елементи діаграми:

1) актори:

– гравець (user): основний користувач системи;

– інтерфейс гри (game interface): інтерфейс, через який гравець взаємодіє з грою;

– камера (camera): функціональність камери в застосунку;

– модуль розпізнавання об’єктів (object recognition module): модуль, який аналізує фотографію;

2) послідовність дій:

- гравець встановлює параметри звуку (soundset);
- гравець налаштовує параметри звуку (settingssound);
- запит на отримання параметрів звуку (getsoundsettings);
- гравець встановлює параметри вібрації (vibrationset);
- гравець налаштовує параметри вібрації (settingsvibration);
- запит на отримання параметрів вібрації (getvibrationsettings;
- гравець відкриває вкладку з квестами (open the quests tab);
- гравець відкриває камеру (open the camera);
- гравець робить фотографію (take a photo);
- фотографія відправляється (send photo);
- модуль розпізнавання об’єктів аналізує фотографію (photo analysis);
- відображення інформації про фотографію (photo information);
- визначення, чи квест завершено (the quest is completed or not).

Якщо попередня діаграма використовується для візуалізації часових аспектів взаємодії, то діаграма кооперації спрямована на деталізацію структурних аспектів взаємодії. Основна особливість діаграми кооперації полягає у здатності графічно відобразити не лише послідовність взаємодій, але й усі структурні зв’язки між об'єктами, що беруть участь у цій взаємодії.

Діаграма кооперації слугує для відображення взаємодії між об'єктами, які виконують певні ролі в системі. Об'єкти на діаграмі подані у вигляді прямокутників із зазначенням їхніх імен, класів та можливих атрибутів. Асоціації між об'єктами позначаються лініями з можливими підписами імен асоціацій та ролей, які відіграють об'єкти. Також на діаграмі можуть бути зображені динамічні зв'язки, зокрема потоки повідомлень [7].

Розглянемо діаграму кооперації для гри що проектується (рис. 2.3).

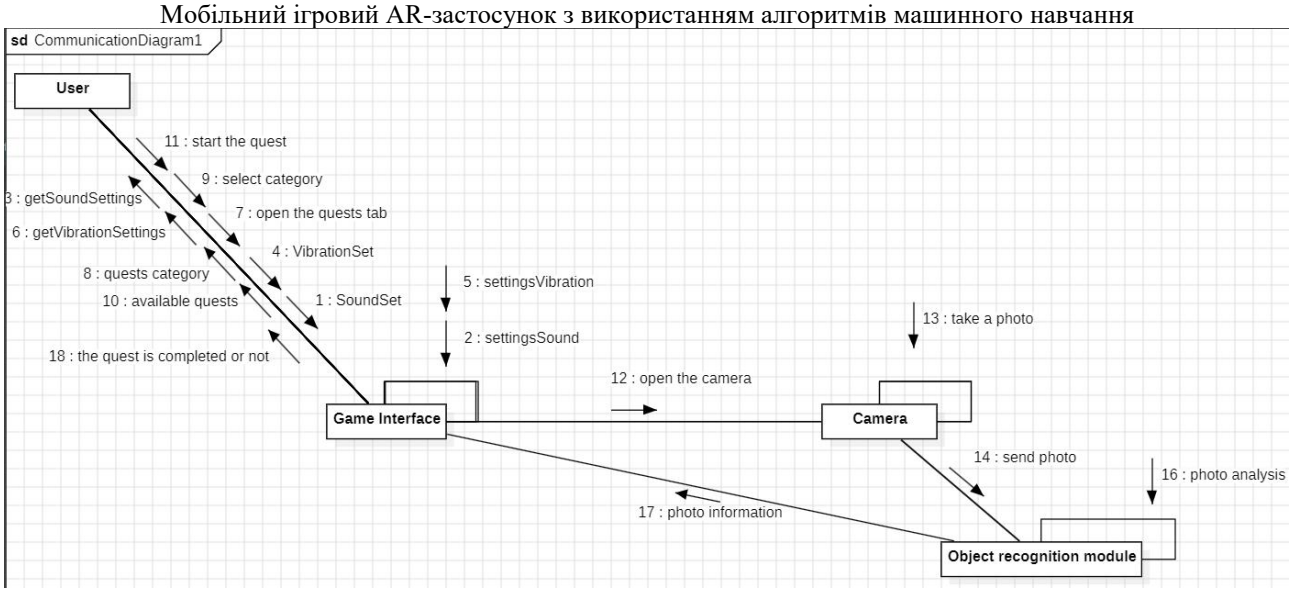


Рисунок 2.3 – Діаграма кооперації

На відміну від діаграми послідовності, діаграма кооперації не представляє час як окремий вимір, але може показувати номери повідомлень для позначення порядку взаємодій. Якщо необхідно детальніше відобразити взаємозв'язки між об'єктами з урахуванням їхнього часу, то доцільніше використовувати діаграму послідовності.

2.4 Побудова діаграм станів

Діаграми станів і переходів (statechart diagrams) є графічними засобами, що використовуються для відображення складних процесів у системах [8]. Вони моделюють поведінку об'єкта, який може перебувати в різних станах, а переходи позначають події, що переводять об'єкт з одного стану в інший. Такі діаграми дозволяють краще зрозуміти, як система функціонує за різних умов та які дії вона виконує в конкретних ситуаціях. Разом із діаграмами діяльності та взаємодії, вони сприяють уточненню вимог до системи і полегшують її розробку.

Основні елементи діаграми:

- коло, що позначає початковий стан;
- коло з точкою посередині, яке вказує на кінцевий стан;
- стрілка, що позначає перехід;
- назва події, що спричиняє перехід, розміщується над або під стрілкою.

Головна мета такої діаграми — описати можливу послідовність станів і переходів, які разом характеризують поведінку елемента моделі протягом його життєвого циклу. Діаграма станів відображає динамічну поведінку об'єкта, описуючи його реакцію на певні події. Системи, що реагують на зовнішні впливи з боку інших систем або користувачів, часто називають реактивними. Якщо така взаємодія відбувається в довільний момент часу, тоді говорять про асинхронну поведінку моделі.

На загальній діаграмі станів (рис. 2.4) представлено загальний приклад функціонування гри.

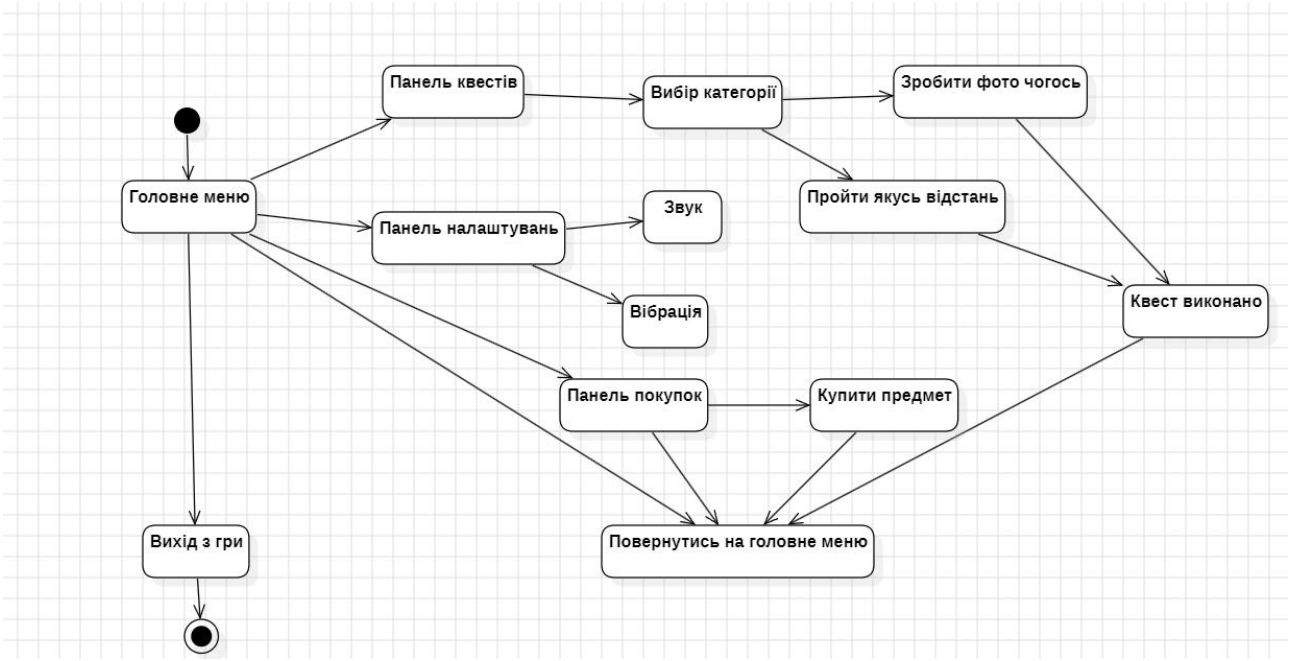


Рисунок 2.4 – Загальна діаграма стану

На загальній діаграмі стану показані основні можливості взаємодії гравця з грою. Таким чином гравець може змінювати налаштування гри, переходити до магазину гри та купувати там речі для свого персонажа. Основною ціллю гри є виконання квестів. Тож задля виконання квестів гравець може обрати одну з категорій квестів, також обрати який саме квест буде виконано.

Наступною діаграмою зображена діаграма станів геймплею (рис. 2.5).

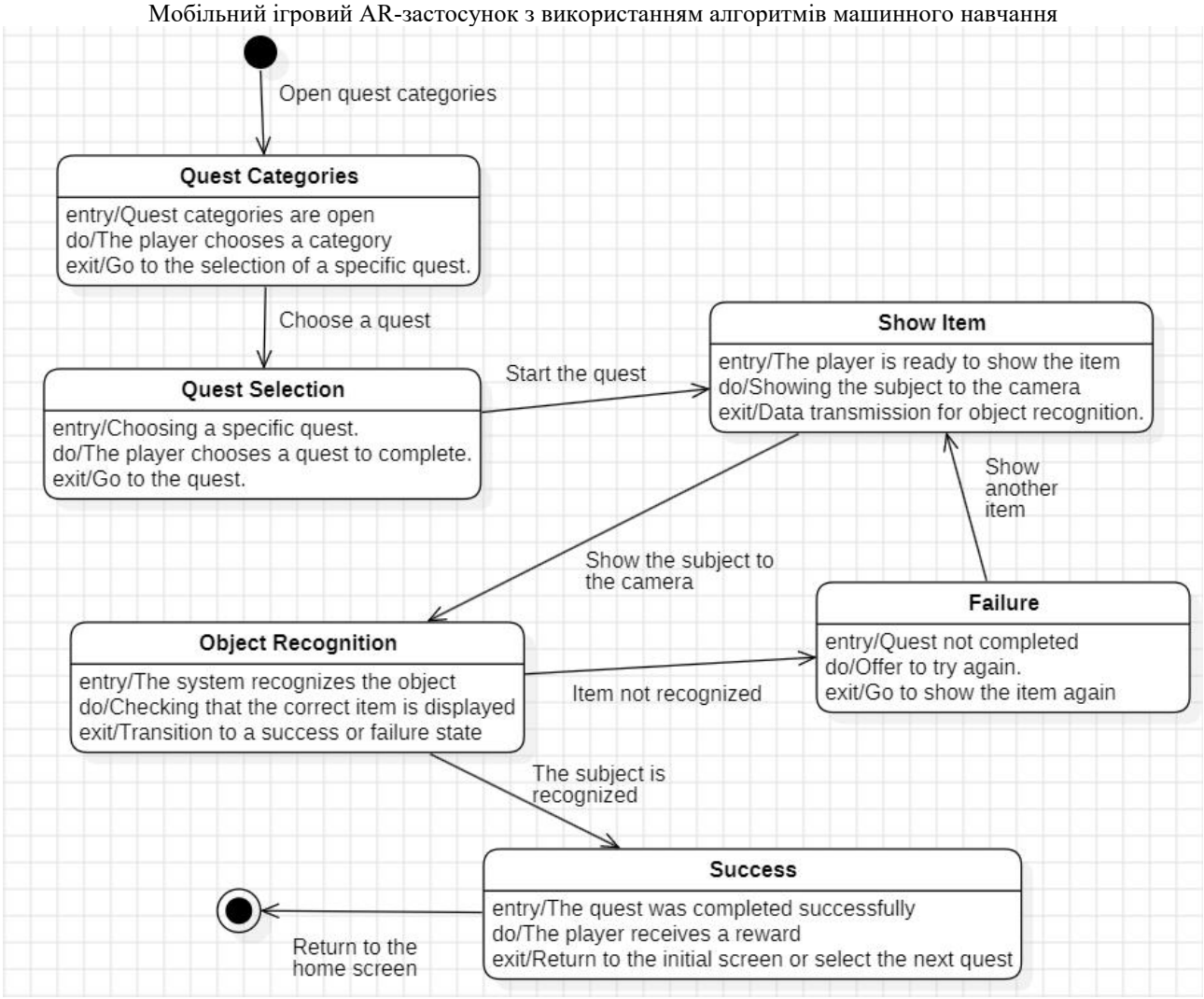


Рисунок 2.5 – Діаграма станів гравця

На діаграмі переходів станів геймплею показано процес гри:

– Quest Categories: Вибір категорії квестів.

– Quest Selection: Вибір конкретного квесту.

– Show Item: Гравець показує предмет на камеру для розпізнавання.

– Object Recognition: Система розпізнає, чи правильний предмет показано.

– Якщо успішно: Success, і гравець отримує винагороду.

– Якщо невдача: Failure, гравець повертається до стану "Show Item" і повторює спробу.

Діаграма станів для ігрової активності з виконання квестів відображає ключові етапи, через які проходить гравець під час гри. Вона містить стани для налаштувань, вибору квестів, показу предметів, розпізнавання та результатів

виконання квестів. Структура діаграми базується на використанні концепцій entry, do, exit, що дозволяє чітко визначити, що відбувається під час кожного етапу, а також передбачає можливі дії та переходи між станами.

Ця діаграма допомагає краще розуміти весь процес виконання квестів та можливі шляхи взаємодії гравця з грою.

2.5 Створення мокапів

Для створення мокапів гри (рис. 2.6-2.11), що розробляється було використано онлайн інструмент Moqups.

Moqups – це онлайн-інструмент, який дозволяє створювати макети, прототипи вебсайтів, додатків та інші візуальні матеріали[9]. Ось деякі ключові аспекти цього інструмента:

– Створення прототипів: Moqups дозволяє вам створювати прототипи вебсторінок та інтерфейсів користувачів. Ви можете розташовувати елементи, такі як форми, кнопки, іконки та текст, на віртуальному полотні, щоб відтворити вигляд вашого проекту.

– Багатий набір інструментів: Moqups має великий вибір готових елементів, які можна використовувати для створення макетів. Це дозволяє вам швидко зібрати візуальний контент без необхідності вмінь програмування або графічного дизайну.

– Колаборація: Інструмент дозволяє працювати в команді. Ви можете запрошувати інших учасників проекту, обмінюватися даними та відстежувати зміни та коментарі.

– Збереження в хмарі: Ваші проекти можна зберігати в хмарі, що дозволяє зручно працювати з ними з будь-якого пристрою.

Загалом, Moqups є потужним інструментом для дизайнерів, розробників та всіх, хто працює з вебдизайном та інтерфейсами користувачів.

Першим розробленим мокапом був мокап головного екрану гри (рис. 2.6).

Рисунок 2.6 – Мокап головного екрану гри

Головний екран гри. В ньому гравець може побачити власний username, рівень, а також кількість накопичених монет. Також гравець може перейти до магазину, перегляду категорій квесту, чи налаштувань.

Другим розробленим мокапом був мокап ігрового магазину (рис. 2.7).

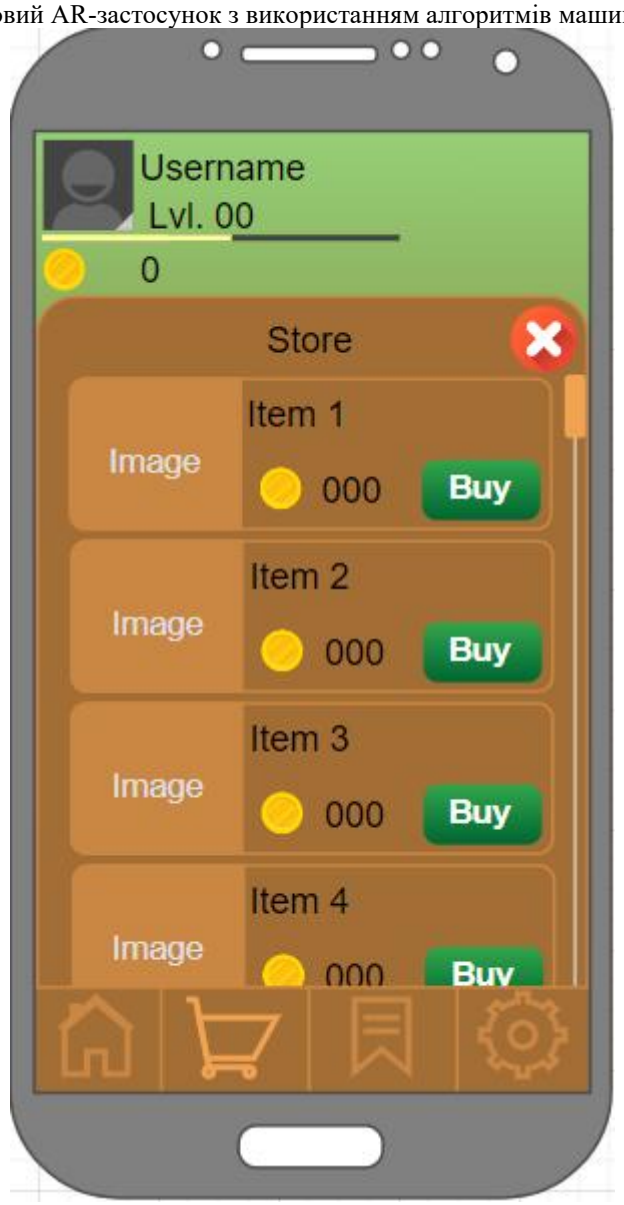


Рисунок 2.7 – Ігровий магазин

Ігровий магазин. Дане меню дозволяє купити якісь предмети для персонажа, одяг, прикраси тощо. Біля кожного предмету є його назва, ціна та кнопка придбання Buy.

Третім розробленим мокапом був мокап вибору категорій квестів (рис. 2.8).

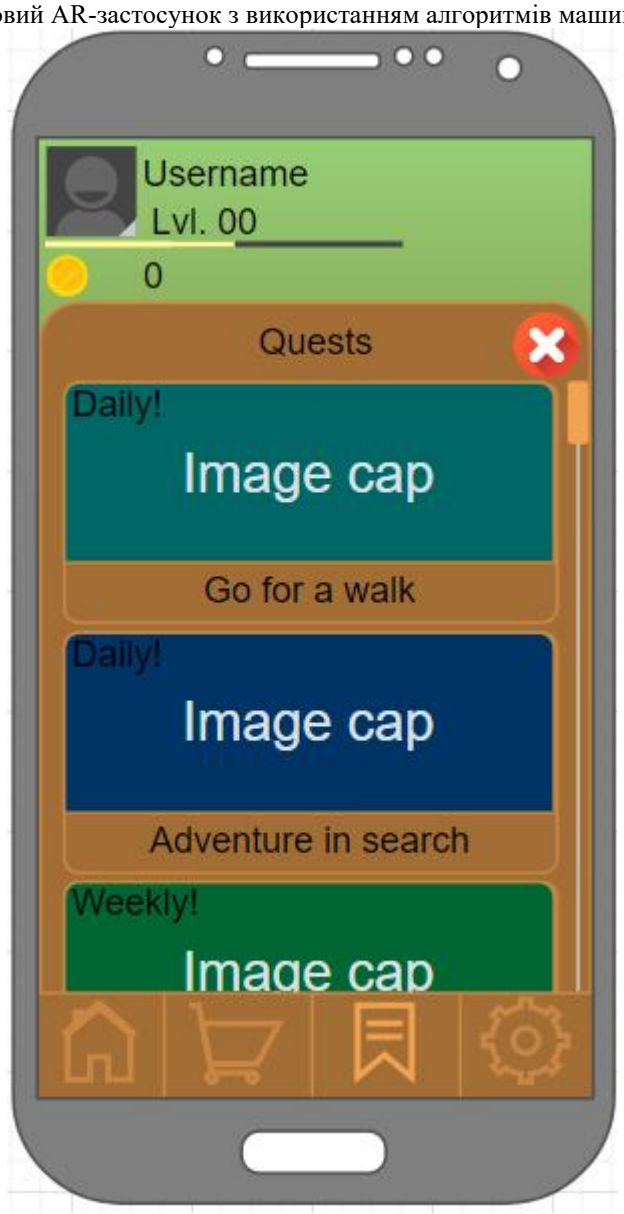


Рисунок 2.8 – Вибір категорії

Вибір категорії. На даний екран можна потрапити тільки натиснувши на кнопку квестів на навігаційній панелі, що доступна на всіх екранах. На даному екрані зображається категорії квестів доступні користувачу. Тут гравець може обрати квести за складністю та за типом виконання, пасивні та активні квести.

Четвертим та п’ятим розробленим мокапом були мокапи виконання пасивних квестів та вибору активних квестів (рис. 2.9 – 2.10).

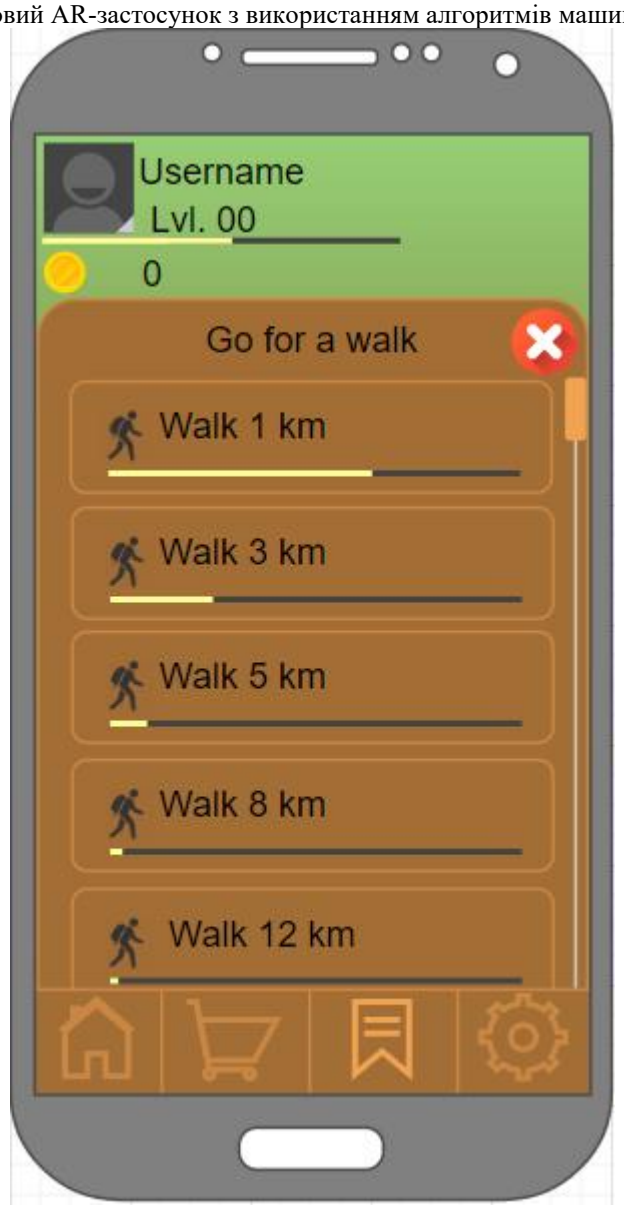


Рисунок 2.9 – Пасивні квести

Пасивні квести доступні доступні відразу і виконуються за рахунок визначення скільки пройшов гравець за один день. На даному екрані зображено прогрес виконання кожного з таких квестів, також можна повернутись на головний екран чи перейти у будь яке інше меню за допомогою навігаційної панелі внизу екрану.

Рисунок 2.10 – Активні квести

Активні квести. Головний тип квестів, їх є три категорії, щоденні, щотижневі та щомісячні. Категорія визначає складність. У даному меню можна перейти до виконання квесту, та переглянути всі доступні квести. Також можна повернутись на головний екран чи перейти у будь яке інше меню за допомогою навігаційної панелі внизу екрану.

Останнім розробленим мокапом був мокап налаштувань (рис. 2.11).

Рисунок 2.11 – Меню налаштувань

У даному меню можна змінювати налаштування звуку та вібрації гри. Також можна повернутись на головний екран чи перейти у будь яке інше меню за допомогою навігаційної панелі внизу екрану.

Висновки до розділу 2

У процесі написання другого розділу було досліджено створення UML-діаграм, таких як діаграми станів, використання та взаємодії, після чого були розроблені власні варіанти цих діаграм. Крім того, було описано сценарії використання в різних формах: короткій, поверхневій та повній. Після аналізу інструменту Moqups за його допомогою були створені макети для гри, що розробляється.

У підсумку можна зазначити, що проведена робота дала чітке розуміння напрямку подальшого розвитку проєкту та його початкових етапів.

3 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ТА ОГЛЯД СТЕКУ ЗАДІЯНИХ ТЕХНОЛОГІЙ

3.1 Розробка UML-діаграм

UML-діаграми – це універсальна мова, яка допомагає візуалізувати та зрозуміти навіть найскладніші програмні системи. Вони як мапи, які ведуть розробників від абстрактних ідей до конкретних реалізацій.

Під час розробки нашої гри ми активно використовували UML для створення детальної моделі. На ранніх етапах, коли ми визначали основні механіки та взаємодії персонажів, нам в пригоді стали діаграми діяльності та станів. Вони допомогли нам чітко уявити, як гратиметься наша гра.

Коли ми перейшли до технічної реалізації, ми звернулися до діаграм класів, компонентів та розгорнення. Завдяки їм ми змогли детально описати структуру коду, взаємодію між різними частинами системи та спланувати розгортання гри на різних платформах.

Для створення всіх цих діаграм ми використовували популярний інструмент StarUML [10]. Він надає зручний інтерфейс та широкий набір функцій для ефективної роботи з UML.

3.1.1 Діаграма класів

Діаграми класів UML: фундамент об'єктно-орієнтованого проектування

Діаграми класів є своєрідним "архітектурним планом" для програмних систем, побудованих за принципами об'єктно-орієнтованого програмування. Уявіть собі, що ви будуєте будинок: діаграма класів - це детальний креслення, на якому зображено всі кімнати (класи), їхні розміри (атрибути) та функції (методи), а також те, як вони з'єднані між собою (відносини).

Клас – це абстрактне представлення групи об'єктів з однаковими властивостями та поведінкою. Наприклад, клас «Автомобіль» може мати атрибути, такі як "колір", "марка", "рік випуску", і методи, такі як "їхати", "гальмувати".

Відносини між класами:

– асоціація: Показує, що між класами існує зв'язок. Наприклад, "Автомобіль" може бути пов'язаний з класом "Водій";

– агрегація: Означає, що один клас є частиною іншого, але може існувати окремо. Наприклад, "Колесо" є частиною "Автомобіля", але може бути замінено;

– композиція: Це більш сильний зв'язок, ніж агрегація. Один клас є невід'ємною частиною іншого і не може існувати окремо. Наприклад, "Двигун" є частиною "Автомобіля" і не може існувати без нього.

Важливістю діаграм є:

– ясність і зрозумілість: Діаграми класів допомагають візуалізувати структуру системи, що полегшує розуміння її принципів роботи як для розробників, так і для замовників;

– планування розробки: Завдяки діаграмам класів можна заздалегідь продумати всі компоненти системи та їх взаємодію, що дозволяє оптимізувати процес розробки;

– документація: Діаграми класів служать відмінною документацією, яка допомагає підтримувати систему та вносити зміни в майбутньому;

– автоматизація: Багато сучасних інструментів розробки дозволяють автоматично генерувати код на основі діаграм класів, що значно прискорює процес створення програмного забезпечення.

Діаграми класів широко використовуються в різних галузях програмної інженерії, від розробки вебдодатків до створення складних корпоративних систем. Так для КМР створена, діаграма класів (рис. 3.1), щоописує структуру даних і логіку роботи системи управління контентом.

.

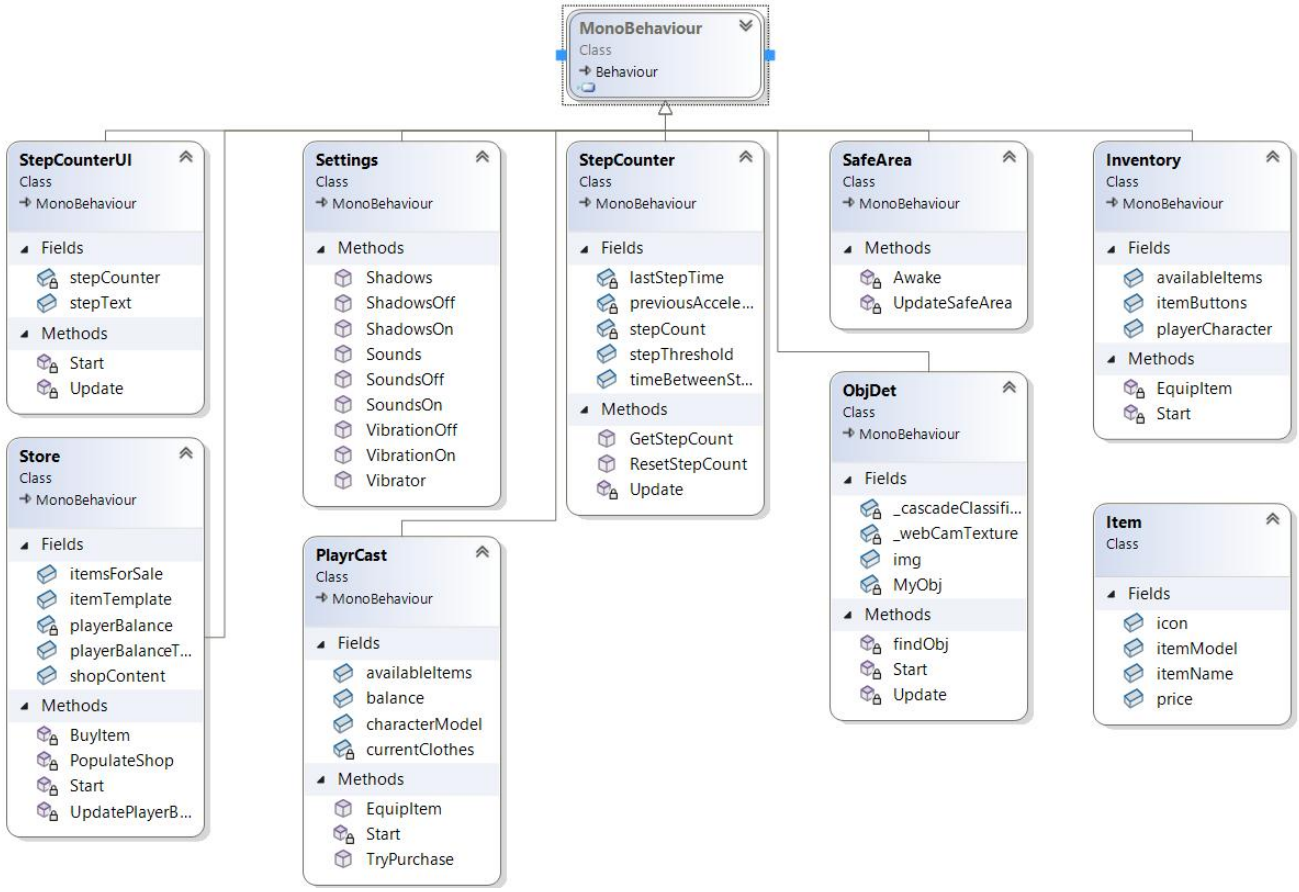


Рисунок 3.1 – Діаграма класів

Клас PlayerCast – відповідає за головний клас персонажа гравця.

Клас Settings– відповідають за збереження інформації про налаштування

Клас Inventory– відповідає за роботу системи інвентаря гри.

Клас ObjDet– відповідає за визначення об’єктів що бачить об’єктив веб камери.

Клас Store– відповідає за магазин у грі.

Клас StepCounter– відповідає за підрахунок кроків.

Клас StepCounterUI– відповідає за відображення кроків у вкладці квестів.

Клас Item– відповідає за визначення об’єктів магазину.

Клас SafeArea – відповідає за адаптацію до дисплеїв різних пристроїв.

3.1.2 Діаграми компонентів та розгортання

Діаграма компонентів є графічним представленням фізичної структури програмної системи. Вона відображає розподіл функціональності системи на

окремі, логічно пов'язані модулі – компоненти. Аналогічно до того, як пазл складається з окремих елементів, так і програмна система складається з компонентів, які в сукупності забезпечують реалізацію поставлених завдань.

Компонент – це модульна частина програмного забезпечення, яка має чітко визначену функціональність. Це може бути окремий файл з кодом, бібліотека, сервіс або навіть ціла підсистема. Кожен компонент має своє ім'я, яке допомагає ідентифікувати його, і зазвичай реалізований на певній мові програмування (наприклад, Java, Python, C++).

Важливість діаграми компонентів:

– візуалізація архітектури: діаграми компонентів допомагають візуалізувати фізичну структуру системи, показуючи, як компоненти взаємодіють між собою;

– планування розробки: завдяки діаграмам компонентів можна розподілити завдання між розробниками, визначити залежності між компонентами та спланувати процес розробки;

– управління змінами: діаграми компонентів допомагають відстежувати зміни в системі та оцінювати їх вплив на інші компоненти;

– документація: діаграми компонентів є важливою частиною документації програмної системи, оскільки вони дозволяють зрозуміти, як система побудована.

Типові елементи діаграми компонентів:

– компоненти: представляють окремі модулі системи;

– інтерфейси: визначають, як компоненти взаємодіють між собою;

– залежності: показують, як зміни в одному компоненті можуть вплинути на інші.

Застосування діаграм компонентів

Діаграми компонентів використовуються на різних етапах розробки програмного забезпечення, від аналізу вимог до тестування та супроводу. Вони особливо корисні для:

– моделювання архітектури великих систем: коли система складається з багатьох взаємопов'язаних компонентів;

– планування розгортання: для визначення, як компоненти будуть розгорнуті на різних середовищах;

– аналізу впливу змін: для оцінки того, як зміни в одному компоненті вплинуть на інші частини системи.

Демонстрація діаграми компонентів (рис. 3.2).

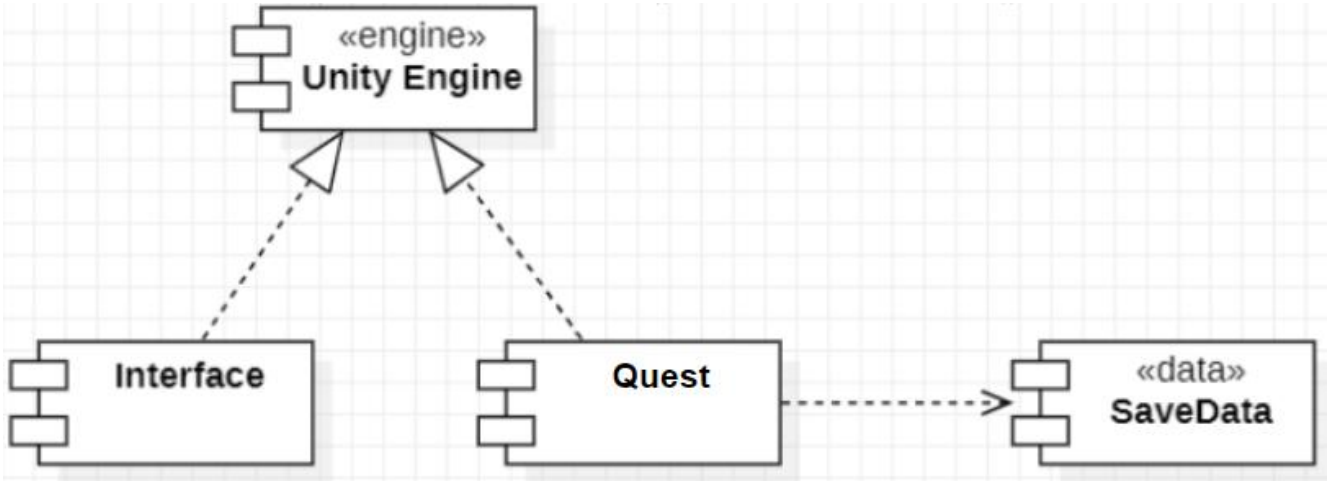


Рисунок 3.2 – Діаграма компонентів

Діаграми розгортання візуалізують фізичну архітектуру системи, демонструючи розташування та взаємодію компонентів на обчислювальних ресурсах. На таких діаграмах відображаються:

– екземпляри компонентів: конкретні реалізації компонентів, що використовуються в системі;

– асоціації компонентів: зв'язки між компонентами, які відображають залежності та взаємодії;

– вузли: фізичні ресурси, такі як сервери, робочі станції, на яких розгорнуті компоненти;

– інтерфейси: специфікації взаємодії компонентів, що визначають, як вони можуть взаємодіяти між собою;

– об'єкти: екземпляри класів, які можуть бути розміщені на вузлах.

Продемонстровано діаграму впровадження (рис. 3.3)

Рисунок 3.3 – Діаграма впровадження

Наступною продемонстрована діаграма розгортання (рис. 3.4).

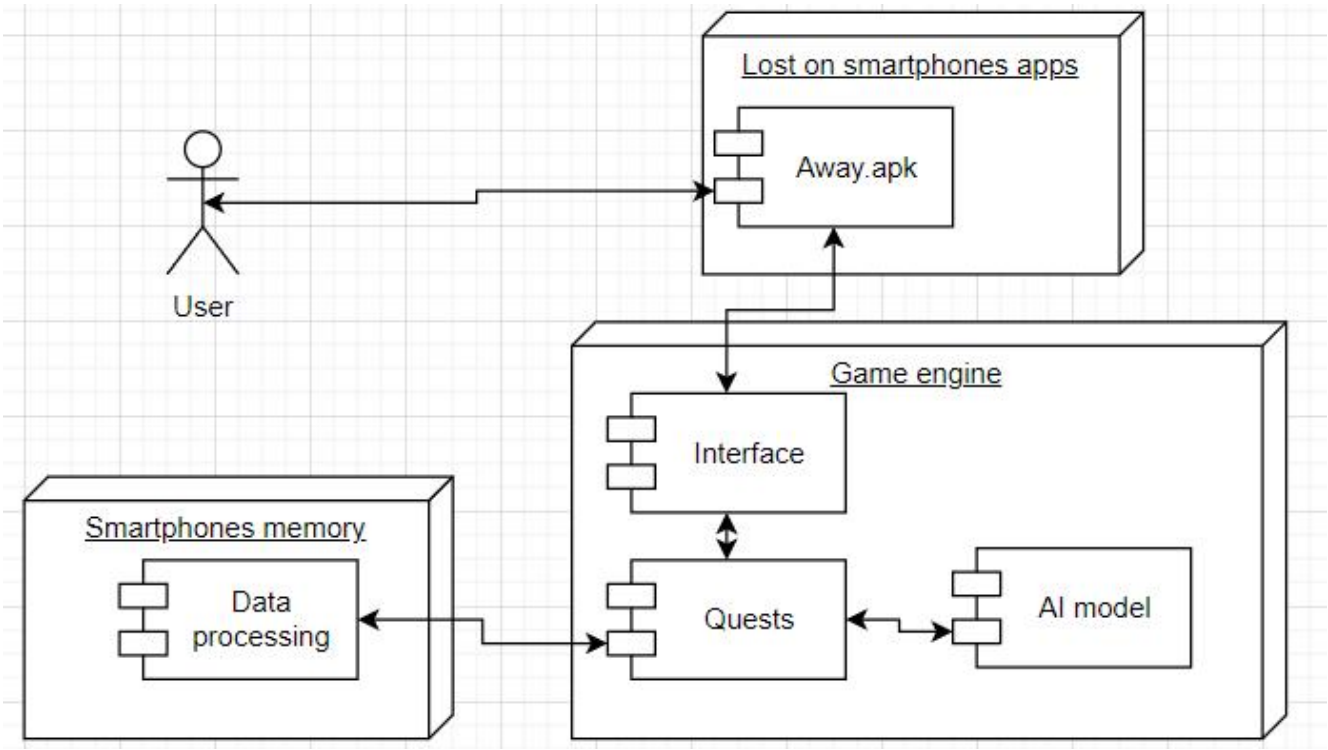


Рисунок 3.4 – Діаграма розгортання

Діаграма розгортання є деталізованим графічним представленням фізичної архітектури програмної системи [13]. Вона зображує топологію мережі, розташування обчислювальних ресурсів (вузлів), а також розподіл компонентів програмного забезпечення по цих ресурсах. Діаграма також ілюструє взаємозв'язки між компонентами та шляхи передачі даних, що дозволяє оцінити ефективність та надійність системи.

3.1.3 Діаграма пакетів

Діаграми пакетів є потужним інструментом в UML для структурування та візуалізації складних програмних систем[14]. Хоча вони базуються на концепції класів та їхніх взаємозв'язків, їхнє основне призначення - об'єднання класів у логічні групи, звані пакетами. Це дозволяє спростити сприйняття великих систем та управляти їхньою складністю.

Ключові особливості діаграм пакетів:

– інкапсуляція: пакети дозволяють приховати внутрішню структуру і деталі реалізації, забезпечуючи модульність системи;

– ієрархія: пакети можуть бути вкладені один в одного, створюючи ієрархічну структуру, що відображає логічну організацію системи;

– залежності: між пакетами можуть існувати різні типи залежностей, такі як імпорт (використання елементів іншого пакету) та злиття (об'єднання кількох пакетів в один);

– стереотипи: для більш детального опису залежностей між пакетами можуть використовуватися стереотипи, наприклад, для позначення залежностей між шарами архітектури.

Застосування діаграм пакетів:

– розбиття системи на модулі: пакети дозволяють розділити систему на більш дрібні, легко керовані частини;

– візуалізація архітектури: діаграми пакетів можуть зображати різні рівні абстракції системи, від загальної структури до деталей реалізації;

– управління залежностями: за допомогою діаграм пакетів можна аналізувати та керувати залежностями між різними частинами системи, мінімізуючи їх вплив на зміни.

Приклади використання:

– моделювання багатошарових архітектур: пакети можуть представляти різні шари системи (презентаційний, бізнес-логіка, дані);

– організація кодів: пакети можуть використовуватися для організації файлів з кодом відповідно до їхніх функціональних призначень;

– визначення контексту для інших діаграм: діаграми пакетів можуть задавати контекст для інших типів діаграм UML, таких як діаграми класів або послідовності.

Розглянемо діаграму пакетів до гри що проектується (рис. 3.5).

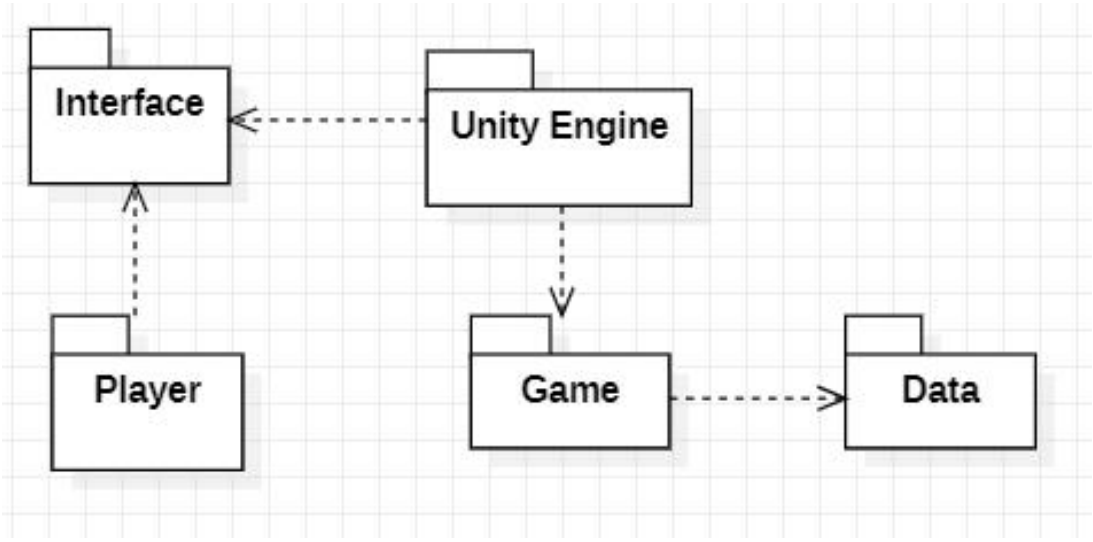


Рисунок 3.5 – Діаграма пакетів

Пакет Player, який відповідає за дії гравця, та Unity Engine, що є рушієм усього проекту, функціонують автономно, не залежачи від інших пакетів. Водночас вони впливають на пакет Interface, який обробляє взаємодію гравця з інтерфейсом та подіями у грі, а також на пакет Game. Пакет Game, відповідальний за геймплей, розпізнавання об’єктів, у свою чергу взаємодіє з пакетом Data, що забезпечує логіку збереження необхідної для гри інформації.

3.2 Огляд технологій

Під час розробки гри було використано такий стек технологій (рис. 3.6):

– мова програмування: C# та Python;

– ігровий рушій: Unity;

– графічний редактор: Photoshop;

– середовище розробки: Jupiter

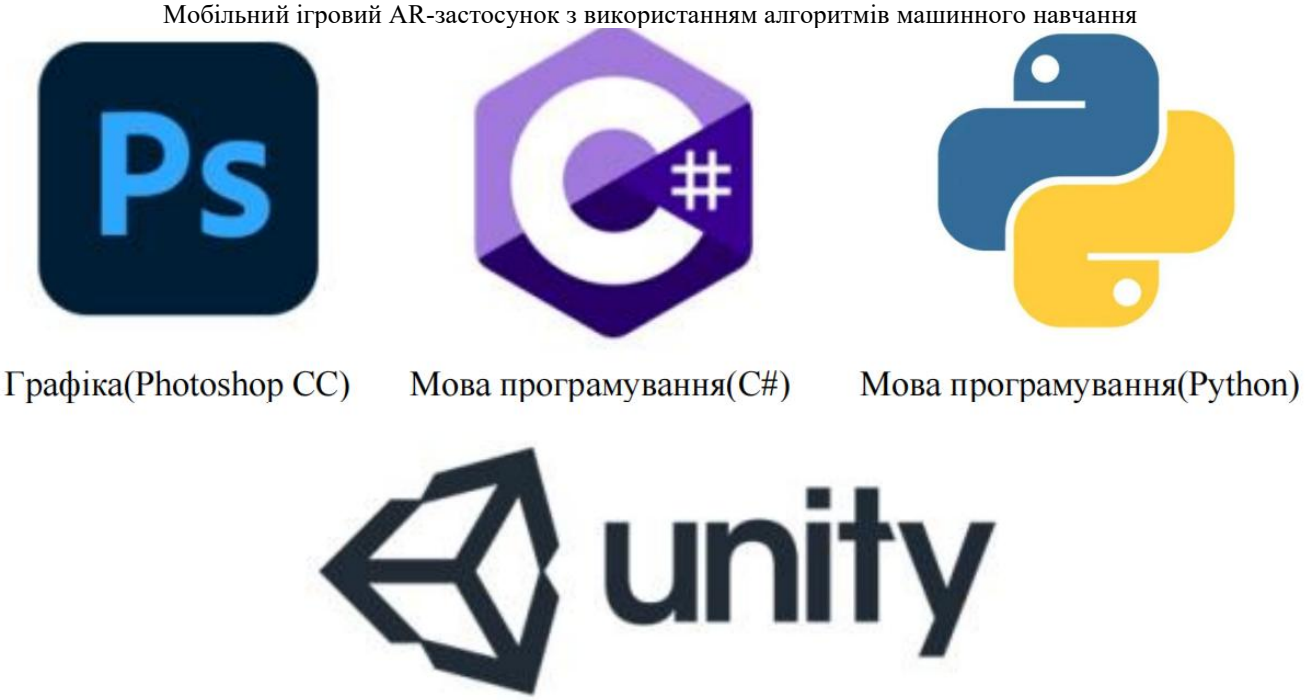


Рисунок 3.6 – Стек технологій з логотипами відповідно

Правильно підібраний стек технологій є фундаментом будь-якого успішного програмного проекту. Від нього залежать такі критичні фактори, як продуктивність, масштабованість, безпека та вартість розробки та підтримки. Неправильний вибір може призвести до технічних боргів, складнощів у розробці та обслуговуванні, а також до нездатності системи задовольнити вимоги користувачів.

3.2.1 Мова програмування C#

C# – потужна та універсальна мова програмування, розроблена компанією Microsoft. Вона поєднує в собі елементи об'єктно-орієнтованого та компонентного програмування, забезпечуючи високу продуктивність, надійність та гнучкість розробки [15].

Ключовими особливостями C# є об'єктно-орієнтоване програмування. C# повністю підтримує парадигму ООП, дозволяючи створювати модульні, повторно використовувані та легко підтримувані програми. Типова безпека: Строга типізація допомагає уникнути багатьох поширених помилок під час розробки, підвищуючи надійність коду. Управління пам'яттю: Вбудований механізм

збирання сміття автоматично звільняє пам'ять, що була зайнята об'єктами, які більше не використовуються. Багатопоточність: C# підтримує паралельне програмування, дозволяючи ефективно використовувати багатоядерні процесори. Інтеграція з .NET Framework: C# тісно інтегрована з .NET Framework, що надає багатий набір бібліотек і компонентів для розробки різних типів додатків. Платформонезалежність: Завдяки .NET Core, програми на C# можуть виконуватися на Windows, macOS, Linux та інших платформах.

Сфери застосування C#:

– веброзробка: ASP.NET забезпечує потужні інструменти для створення динамічних вебсайтів і вебсервісів.

– Розробка десктопних додатків: C# використовується для створення нативних додатків для Windows, macOS та Linux.

– Розробка ігор: Unity, популярний ігровий рушій, використовує C# як основну мову програмування.

– Мобільна розробка: Xamarin, заснований на C#, дозволяє створювати кросплатформні мобільні додатки для iOS, Android і Windows.

– Розробка корпоративних застосунків: C# широко використовується для створення масштабних і надійних корпоративних систем.

– Data Science і Machine Learning: C# може бути використаний для аналізу даних, створення моделей машинного навчання та побудови систем штучного інтелекту.

Переваги використання C#:

– Висока продуктивність: Компіляція в машинний код забезпечує високу швидкість виконання програм.

– Велике співтовариство: Активна спільнота розробників, велика кількість бібліотек і фреймворків.

– Підтримка Microsoft: Постійна підтримка з боку Microsoft гарантує довготривалу життєздатність платформи.

– Сучасні можливості: C# підтримує сучасні мовні конструкції, такі як LINQ, лямбда-вирази, асинхронне програмування.

C# є універсальною і потужною мовою програмування, яка дозволяє розробникам створювати різноманітні програмні продукти високої якості. Її широкі можливості, велика спільнота та підтримка Microsoft роблять її одним з найпопулярніших виборів для розробників у всьому світі.

3.2.2 Мова програмування Python та середовище розробки Jupiter

Python – це інтерпретована, високорівнева мова програмування загального призначення, відома своєю простотою та читабельністю синтаксису. Вона була розроблена Гвідо ван Россумом наприкінці 1980-х років і з того часу стала однією з найпопулярніших мов у світі [16].

Головні переваги Python:

– простий та інтуїтивний синтаксис: синтаксис Python нагадує звичайну англійську мову, що робить його легко вивчити, особливо для початківців;

– багатопарадигменність: Python підтримує як процедурне, так і об'єктно-орієнтоване програмування, а також функціональне програмування;

– величезна стандартна бібліотека: Python поставляється з великою кількістю вбудованих модулів, що дозволяють виконувати різноманітні завдання без необхідності писати додатковий код;

– багата екосистема: існує величезна кількість сторонніх бібліотек і фреймворків для Python, які розширюють його можливості в різних областях, таких як наука про дані, машинне навчання, веброзробка, автоматизація та багато інших;

– платформна незалежність: код, написаний на Python, може виконуватися на різних операційних системах без необхідності внесення змін.

Сфери застосування Python:

– веброзробка: Фреймворки Django і Flask дозволяють швидко створювати масштабовані вебдодатки;

– наукові обчислення і машинне навчання: бібліотеки NumPy, Pandas, SciPy, YOLO, Matplotlib і TensorFlow роблять Python потужним інструментом для аналізу даних, візуалізації та машинного навчання;

– автоматизація: Python широко використовується для автоматизації рутинних завдань, таких як тестування, адміністрування систем і збір даних;

– розробка ігор: фреймворки Pygame і Kivy дозволяють створювати ігри для різних платформ;

– скриптування: Python часто використовується для написання невеликих скриптів для автоматизації різних завдань.

Переваги використання Python:

– висока продуктивність розробки: завдяки простому синтаксису і великій кількості готових бібліотек розробники можуть швидко створювати прототипи і готові продукти;

– читабельність коду: чіткий і лаконічний синтаксис Python робить код більш зрозумілим і легким для підтримки;

– гнучкість: Python можна використовувати для вирішення широкого кола завдань, від простих скриптів до складних наукових обчислень;

– активна спільнота: велика і активна спільнота розробників Python забезпечує постійний розвиток мови і створення нових інструментів.

Python – це універсальна мова програмування, яка підходить як для початківців, так і для досвідчених розробників. Її простота, гнучкість і велика екосистема роблять її одним з найпопулярніших виборів для розробки різноманітних програмних продуктів.

Jupyter Notebook – це вебзастосунок для інтерактивних обчислень, який дозволяє створювати та ділитися документами, що містять живий код, рівняння, текстові пояснення, візуалізації та інші медіа (рис. 3.7). Він підтримує понад 40 мов програмування, включаючи Python, R, Julia та Scala.

Рисунок 3.7 – Дистрибуція Anaconda

Основні можливості Jupyter Notebook:

1) інтерактивний код: ви можете писати та виконувати код безпосередньо в браузері;

2) візуалізації: легко створювати графіки та діаграми для аналізу даних;

3) документація: додавати текстові пояснення та рівняння LaTeX для кращого розуміння коду;

4) спільний доступ: ділитися нотатками через електронну пошту, GitHub або спеціальний переглядач Jupyter Notebook.

Jupyter Notebook входить до дестрибуції Anaconda, а також є частиною проекту Jupyter, який також включає JupyterLab – більш сучасне середовище для роботи з нотатками, кодом та даними.

Anaconda – це безкоштовна та відкрита дистрибуція мов програмування Python та R, яка спеціально розроблена для наукових обчислень, машинного навчання та аналізу даних(рис. 3.8). Вона включає в себе Python-інтерпретатор та різноманітні пакети, необхідні для роботи з даними, такі як pandas, NumPy, scikit-learn та багато інших.

Рисунок 3.8 – Дистрибуція Anaconda

Основні переваги Anaconda:

1) легкість встановлення: всі необхідні пакети встановлюються разом з anaconda, що значно спрощує процес налаштування середовища;

2) conda: це система управління пакетами та середовищами, яка дозволяє легко встановлювати, оновлювати та видаляти пакети.

Anaconda також включає графічний інтерфейс Anaconda Navigator, який дозволяє керувати пакетами та середовищами без використання командного рядка.

3.2.3 Ігровий рушій

Unity – це не просто інтегроване середовище розробки (IDE), а ціла платформа для створення ігор, віртуальної та доповненої реальності. Це потужний інструмент, який дозволяє перетворити ваші творчі ідеї на захоплюючі інтерактивні досвіди [17].

Чому Unity такий популярний?

– універсальність: Unity дозволяє створювати проекти для широкого спектру платформ, від комп'ютерів до мобільних пристроїв і консолей;

– візуальне програмування: інтуїтивний візуальний редактор спрощує створення сцен, розміщення об'єктів і налаштування їхніх властивостей;

– гнучкість: скриптовий рушій на c# надає повну свободу у створенні власної ігрової логіки та поведінки об'єктів;

– розширені можливості: величезний асортимент готових активів, від моделей персонажів до графічних ефектів, дозволяє прискорити розробку;

– реалістична фізика: потужні інструменти для створення реалістичної фізики роблять ігри більш інтерактивними і привабливими;

– VR і AR: вбудована підтримка VR і AR відкриває нові можливості для створення іммерсивних досвідів.

Основні переваги Unity:

– доступність: unity має безкоштовну версію з широкими можливостями, що робить його доступним для розробників будь-якого рівня;

– спільнота: велика і активна спільнота розробників, що постійно ділиться досвідом і створює нові інструменти;

– регулярні оновлення: unity постійно розвивається, отримуючи нові функції і покращення.

Сфери застосування Unity:

– ігри: створення 2D і 3D ігор для різних платформ;

– віртуальна реальність: розробка VR-додатків для ігор, навчання та інших сфер;

– доповнена реальність: створення AR-додатків для мобільних пристроїв;

– інтерактивні інсталяції: розробка інтерактивних виставок, музеїв та інших проєктів;

– прототипування: швидке створення прототипів ігор і додатків.

Unity є одним з найпопулярніших інструментів для розробки ігор та інтерактивних додатків завдяки своїй потужності, гнучкості та розширюваності. Він дозволяє розробникам мати повний контроль над процесом розробки та створювати вражаючі творіння у світі геймдеву та інтерактивних додатків.

3.2.4 Графічний редактор

Adobe Photoshop – це не просто графічний редактор, а потужне програмне забезпечення, яке стало стандартом індустрії для роботи з растровою графікою. Розроблений компанією Adobe, Photoshop пропонує широкий спектр інструментів

і функцій, що дозволяють досягти найвищої якості результатів у фоторедагуванні, дизайні та створенні цифрового мистецтва[18].

Чому Photoshop є таким популярним:

– всебічність: від простих корекцій кольору до складних маніпуляцій зі зображеннями – photoshop впорається з будь-яким завданням;

– творчі можливості: інструменти для малювання, створення тексту, застосування ефектів та багато іншого дозволяють втілити будь-які творчі ідеї;

– професіоналізм: Photoshop використовується професійними фотографами, дизайнерами та ілюстраторами для створення високоякісних зображень;

– гнучкість: робота зі шарами надає повну свободу в редагуванні окремих елементів зображення;

– сумісність: підтримка великої кількості форматів файлів дозволяє працювати з різними типами зображень.

Основні сфери застосування Photoshop:

1) фоторедагування: корекція кольору, видалення недоліків, ретушування, створення колажів;

2) графічний дизайн: створення логотипів, макетів, ілюстрацій, баннерів;

3) вебдизайн: підготовка зображень для вебсайтів;

4) цифрове мистецтво: створення цифрових малюнків і ілюстрацій;

5) обробка фотографій для друку: підготовка зображень для друку в високій якості.

Чому варто обрати Photoshop:

– стандарт індустрії: Photoshop є найбільш розповсюдженим і визнаним інструментом у своїй галузі;

– постійна підтримка: компанія adobe регулярно оновлює і доповнює функціонал програми;

– величезна спільнота: існує велика кількість навчальних матеріалів, підручників і форумів, де можна знайти відповіді на будь-які питання.

Adobe Photoshop – це незамінний інструмент для всіх, хто працює з зображеннями. Незалежно від того, чи ви професійний дизайнер, фотограф-аматор або просто хочете покращити свої фотографії, Photoshop допоможе вам досягти найкращих результатів.

Висновки до розділу 3

Третій розділ зосереджено на проєктуванні гри та аналізі використаних технологій, таких як мови програмування (C# і Python), ігровий рушій (Unity Engine) і графічний редактор (Photoshop). У процесі проєктування застосунку були отримані нові навички у створенні UML-діаграм, зокрема діаграм класів, станів, переходів і пакетів. Кожна діаграма була детально та чітко описана, що сприяє зручності їх подальшого використання.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ГРИ

4.1 Створення інтерфейсу гри

Для створення інтерфейсу було за допомогою програми Photoshop створено спрайти для кнопок інтерфейсу (рис. 4.1).

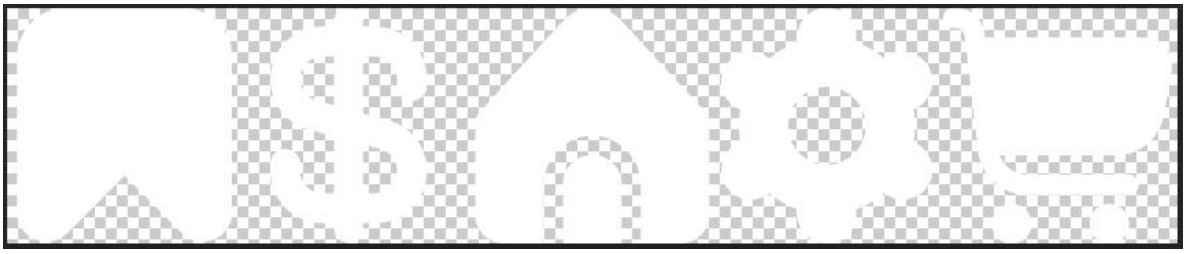


Рисунок 4.1 – Підготовка майбутніх спрайтів кнопок

Далі при створенні кожного спрайту було налаштовано параметри зображення за допомогою вікна інспектора в Unity (рис. 4.2) таким чином, щоб кожен спрайт можна було використати для окремої кнопки в інтерфейсі.

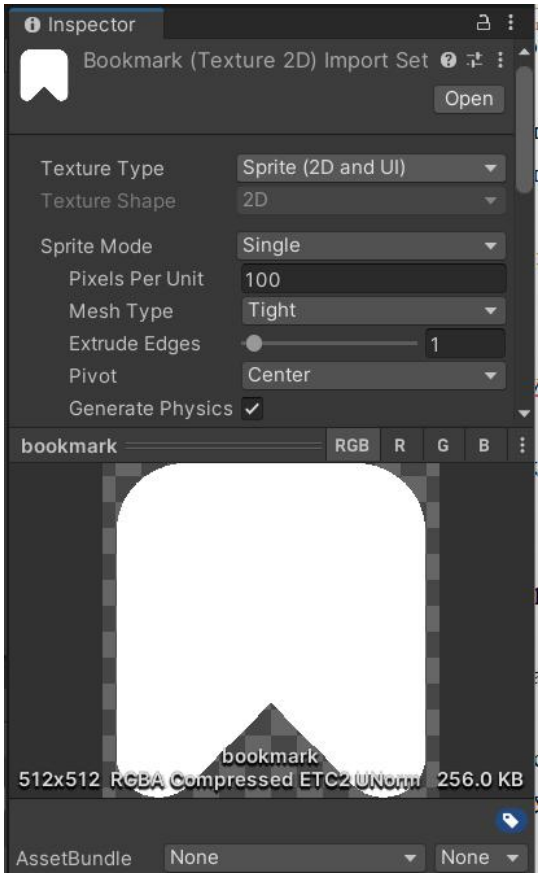


Рисунок 4.2 – Налаштування спрайтів у інспекторі Unity

Далі вже в самому інтерфейсі було створено UI елементи які відповідають за певні дії. Таким чином було створено бар навігації, сторінки магазину, квестів, головної сторінки та сторінки налаштувань (рис. 4.3).

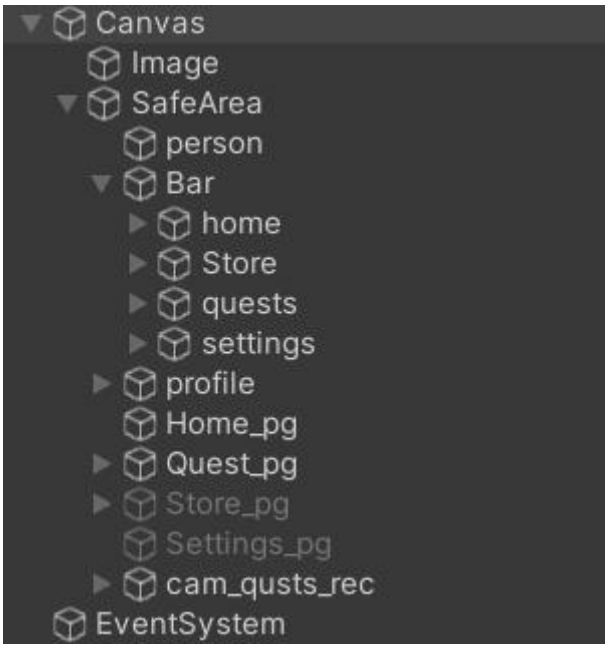


Рисунок 4.3 – Стек UI елементів інтерфейсу

Для адаптивності було створено об’єкт SafeArea та його відповідний клас (рис. 4.4), що відповідає за адаптивність інтерфейсу водповідно до особливостей екрану смартфонів(вирізи камер, заокругленість країв тощо).

```
using UnityEngine;

public class SafeArea : MonoBehaviour
{
    private void Awake()
    {
        UpdateSafeArea();
    }
    private void UpdateSafeArea()
    {
        var safeArea = Screen.safeArea;
        var myRectTransform = GetComponent<RectTransform>();

        var anchorMin = safeArea.position;
        var anchorMax = safeArea.position + safeArea.size;

        anchorMin.x /= Screen.width;
        anchorMin.y /= Screen.height;
        anchorMax.x /= Screen.width;
        anchorMax.y /= Screen.height;

        myRectTransform.anchorMin = anchorMin;
        myRectTransform.anchorMax = anchorMax;
    }
}
```

Рисунок 4.4 – Лістинг коду класу адаптації інтерфейсу SafeArea

Для прикладу проглянемо готові головну сторінку з персонажем та сторінки квестів (рис. 4.5-4.6).

Рисунок 4.5 – Головне меню гри

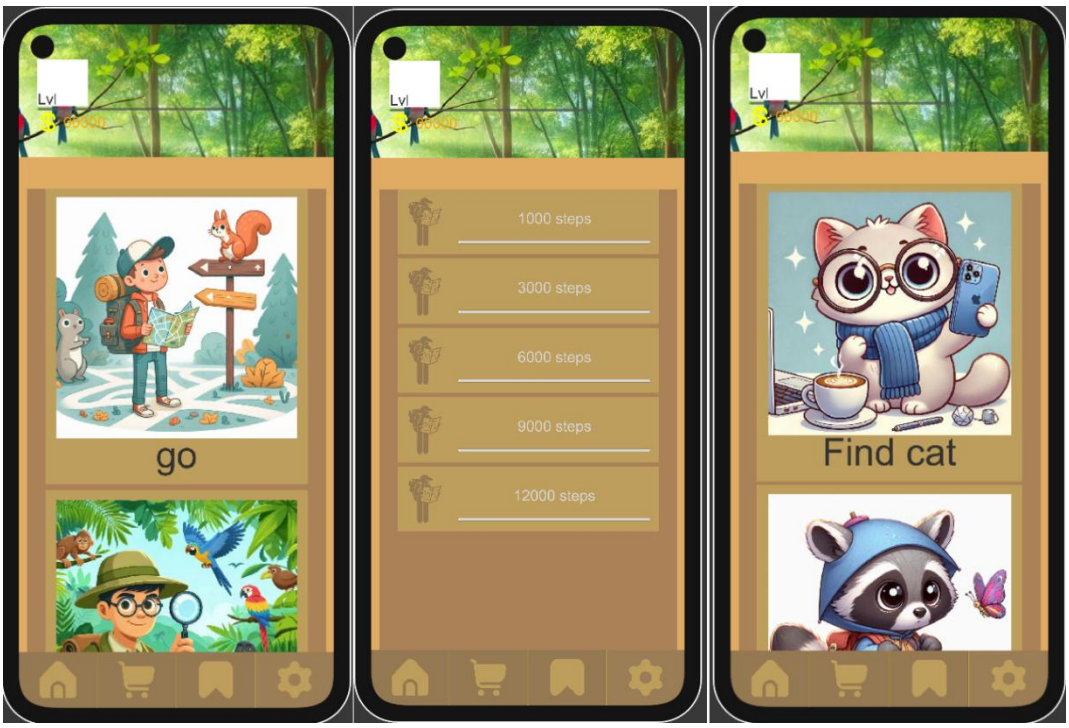


Рисунок 4.6 – Інтерфейс сторінок меню квестів

Таким чином було створено весь інтерфейс гри та всі головні меню та підменю.

4.2 Створення героя гри, магазину та його предметів

Для більш захоплюючого геймплею створено героя для гри та можливість купувати йому особливі артефакти які можуть підвищувати певні характеристики. Наступним зображенням демонструється наявність героя та відповідних його комірок для унікальних предметів (рис. 4.7).

Рисунок 4.7 – Головний герой та комірки інвертарю

Також для персонажа було написано два скрипти які відповідають за предмети що є у наявності у самого персонажа та предмети якими він користується наступним кроком представлено вікно інспектора відповідних скриптів (рис. 4.8).

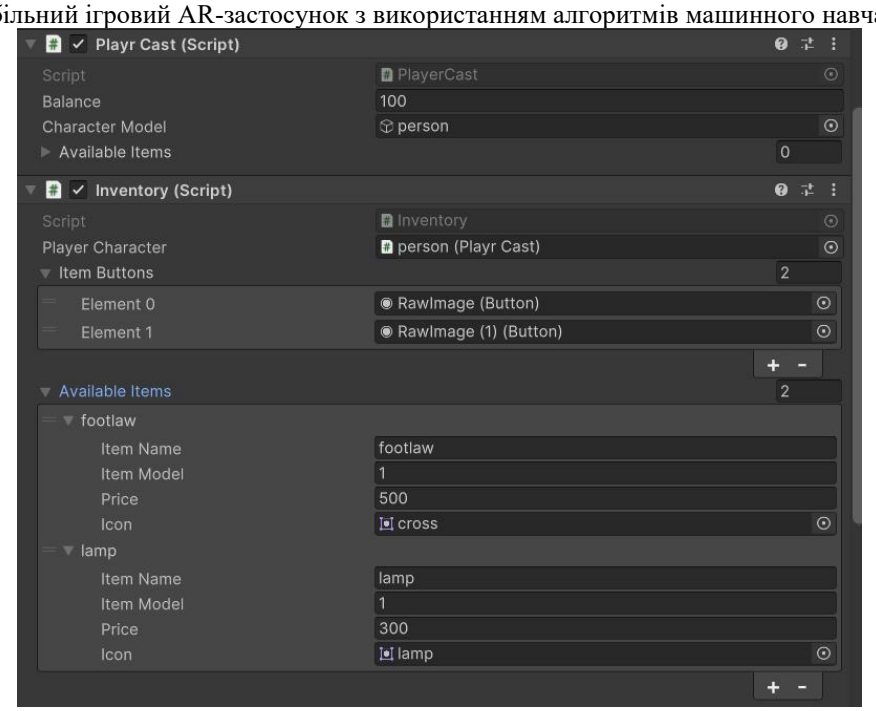


Рисунок 4.8 – Налаштування скриптів гравця у інспекторі

Для генерації інтерфейсу магазину було виконано проєктування самого інтерфейсу (рис. 4.9), а також описано скрипт логіки купівлі товарів з нього та логіку заповнення самого магазину предметами (рис. 4.10–4.11).

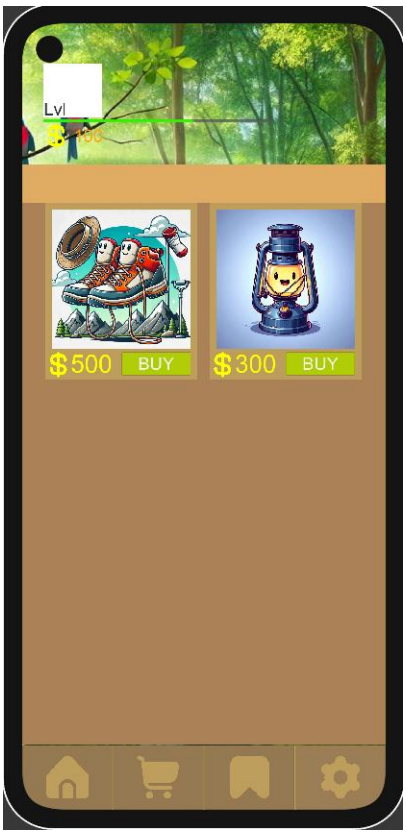


Рисунок 4.9 – Сторінка магазину в грі

```
1 reference
void PopulateShop()
{
    foreach (Item item in itemsForSale)
    {
        GameObject itemGO = Instantiate(itemTemplate, shopContent);
        itemGO.SetActive(true);

        itemGO.transform.Find("ItemName").GetComponent<Text>().text = item.itemName;
        itemGO.transform.Find("ItemPrice").GetComponent<Text>().text = item.price.ToString();
        itemGO.transform.Find("ItemIcon").GetComponent<Image>().sprite = item.icon;

        Button buyButton = itemGO.transform.Find("BuyButton").GetComponent<Button>();
        buyButton.onClick.AddListener(() => BuyItem(item));
    }
}
```

Рисунок 4.10 – Лістинг коду функції заповнення сторінки магазину

```
void BuyItem(Item item)
{
    if (playerBalance >= item.price)
    {
        playerBalance -= item.price;
        Debug.Log($"Ви купили {item.itemName} за {item.price} монет.");
        UpdatePlayerBalance();
    }
    else
    {
        Debug.Log("Недостатньо монет!");
    }
}
```

Рисунок 4.11 – Лістинг коду функції купівлі товарів на сторінці магазину

Дані скрипти дозволяють у майбутньому легко розширювати вміст магазину та додавати більше унікальних предметів у гру, що однозначно покращить ігровий досвід гравців.

4.3 Створення логіки генерації квестів

Головним елементом гри є квести які необхідно виконувати гравцям. У грі організовано 2 види квестів:

1) квести на пройдений шлях;

2) квести з логікою “Покажи мені щось”, де гравці мають продемонструвати необхідний предмет на камеру смартфона.

Розглянемо перший варіант квестів на підрахунок кроків. Для цього було створено інтерфейс який містить 5 варіантів відстаней (рис. 4.12) після проходження якої гравцю надається винагорода у вигляді монет.

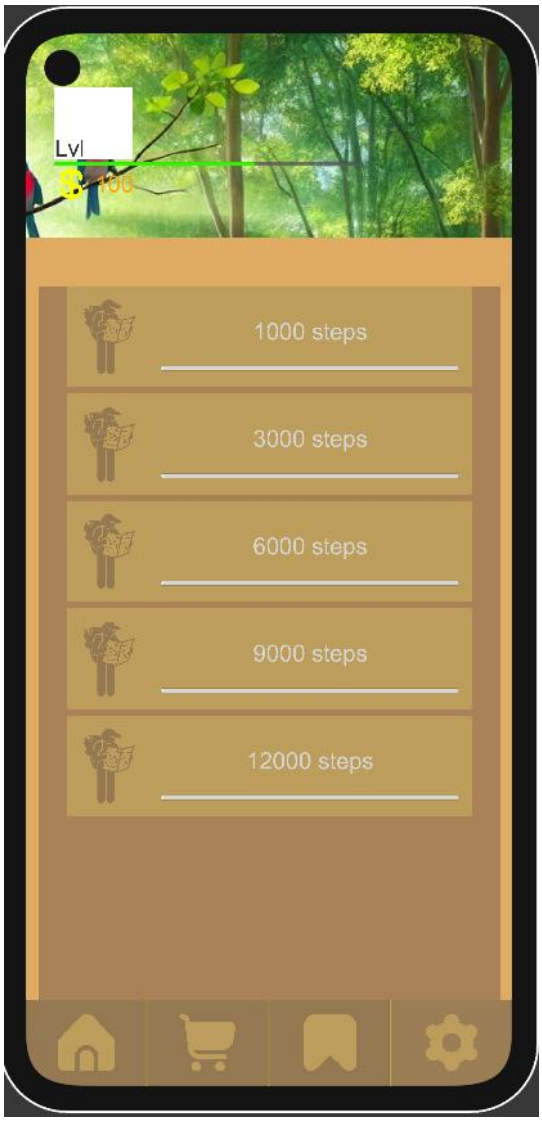


Рисунок 4.12 – Сторінка квестів на проходження відстані у грі

Також для підрахунку кількості кроків було створено скрипт який підраховує кроки на основі даних датчика акселератора телефону (рис. 4.13).

```
public class StepCounter : MonoBehaviour
{
    // Порогові значення
    public float stepThreshold = 1.0f; // Мінімальне прискорення для врахування кроку
    public float timeBetweenSteps = 0.5f; // Мінімальний час між кроками (у секундах)

    private int stepCount = 0; // Лічильник кроків
    private float lastStepTime = 0.0f; // Час останнього кроку
    private Vector3 previousAcceleration = Vector3.zero; // Попереднє прискорення

    Unity Message | 0 references
    void Update()
    {
        Vector3 acceleration = Input.acceleration;

        // Обчислення зміни прискорення
        float deltaAcceleration = (acceleration - previousAcceleration).magnitude;

        // Перевірка, чи перевищено поріг і чи пройшов час між кроками
        if (deltaAcceleration > stepThreshold && Time.time - lastStepTime > timeBetweenSteps)
        {
            stepCount++;
            lastStepTime = Time.time;
            Debug.Log($"Крок {stepCount}: Δa = {deltaAcceleration}");
        }

        // Оновлення попереднього прискорення
        previousAcceleration = acceleration;
    }

    1 reference
    public int GetStepCount()
    {
        return stepCount;
    }

    0 references
    public void ResetStepCount()
    {
        stepCount = 0;
    }
}
```

Рисунок 4.13 – Лістинг коду класу підрахунку зроблених кроків

Коротке пояснення коду:

– Input.acceleration: використовується для отримання поточних даних акселерометра;

– поріг (stepThreshold): мінімальна зміна прискорення, що вважається кроком. Залежить від чутливості пристрою;

– час між кроками (timeBetweenSteps): фільтр для уникнення помилкових спрацьовувань через шум;

– лічильник (stepCount): збільшується кожного разу, коли пристрій реєструє крок.

Було розроблено сторінку основних квестів гри та механізм їх генерації. На початковому етапі створено інтерфейс для квестів, пов’язаних із пошуком предметів (рис. 4.14).

Рисунок 4.14 – Інтерфейс квестів

Для його генерації було використано схожий скрипт як і для генерації магазину з використанням готового префабу та дублікаціїєю його на сторінці з різними зображеннями та варіантами пошуку необхідного об’єкту.

4.4 Створення та тестування моделі розпізнавання об’єктів

4.4.1 Вибір моделі

Для створення власної моделі розпізнавання об’єктів використано модель YOLOv5.

YOLOv5 (You Only Look Once, Version 5) — це популярна модель для задач комп'ютерного зору, зокрема об'єктного детектування [18]. Вона продовжує ідеї сімейства YOLO-моделей, які відомі своєю швидкістю та точністю. YOLOv5 була розроблена компанією Ultralytics і отримала широке використання завдяки своїй простоті та ефективності.

Основні характеристики YOLOv5:

1) Швидкість і продуктивність:

– YOLOv5 демонструє високу швидкість роботи на сучасному апаратному забезпеченні, включаючи GPU та CPU;

– вона оптимізована для реального часу, що робить її ідеальною для застосунків, де потрібне швидке виявлення об'єктів.

2) Різні розміри моделей: YOLOv5 доступна у декількох варіантах (моделях різного розміру):

– YOLOv5s (small): найлегша і найшвидша модель;

– YOLOv5m (medium): компроміс між швидкістю та точністю;

– YOLOv5l (large): точніша, але повільніша;

– YOLOv5x (extra-large): найточніша, але споживає більше ресурсів.

3) Сумісність з PyTorch:

– YOLOv5 написана на PyTorch, що спрощує її інтеграцію в проєкти;

– забезпечує легкий старт для тренування моделей з попередньо підготовленими вагами або з нуля.

4) Розширені функції:

– підтримка augmentation (розширення даних) для покращення продуктивності;

– автоматичне масштабування для адаптації до розмірів зображень;

– підтримка декількох форматів виводу (ONNX, CoreML, TensorRT), що дозволяє інтегрувати модель у мобільні застосунки.

5) Супутні функціональності:

– легка інтеграція для використання у хмарних сервісах (AWS, Google Cloud);

– зручні інструменти візуалізації результатів.

YOLOv5, як і інші версії YOLO, базується на підході, де зображення розбивається на сітку, і кожна комірка сітки прогнозує:

1) наявність об'єкта;

2) прямокутник, що охоплює об'єкт (bounding box);

3) клас об'єкта.

Це дозволяє YOLO бути дуже швидким, оскільки обробка всього зображення здійснюється за один прохід мережі.

Перевагами YOLOv5 є:

1) легкий запуск і навчання;

2) простота інтеграції з існуючими фреймворками;

3) висока швидкість і точність для реального часу;

4) підтримка малоресурсних пристроїв (включаючи мобільні платформи).

Недоліками ж є те що:

1) YOLOv5 не є офіційним продовженням оригінальної YOLO (розробленої Джозефом Редмоном). Через це виникають деякі непорозуміння в спільноті;

2) у певних випадках поступається точністю моделям останнього покоління (наприклад, YOLOv8, Detectron2 або EfficientDet).

Є доволі багато варіантів застосування YOLOv5, одними з яких є:

1) безпека: відеоспостереження та детекція порушень;
2) автомобільна індустрія: системи допомоги водіям (ADAS);
3) роздрібна торгівля: автоматизація інвентаризації;
4) робототехніка: автономна навігація;
5) AR/VR: реальний час об'єктного виявлення для інтерактивних застосунків.

4.4.2 Набір даних та підготовка моделі

Для навчання моделі було використано набір даних від Microsoft для YOLOv5. Набір містить 5000 фото на яких зображено мінімум 1 об’єкт (рис. 4.15).

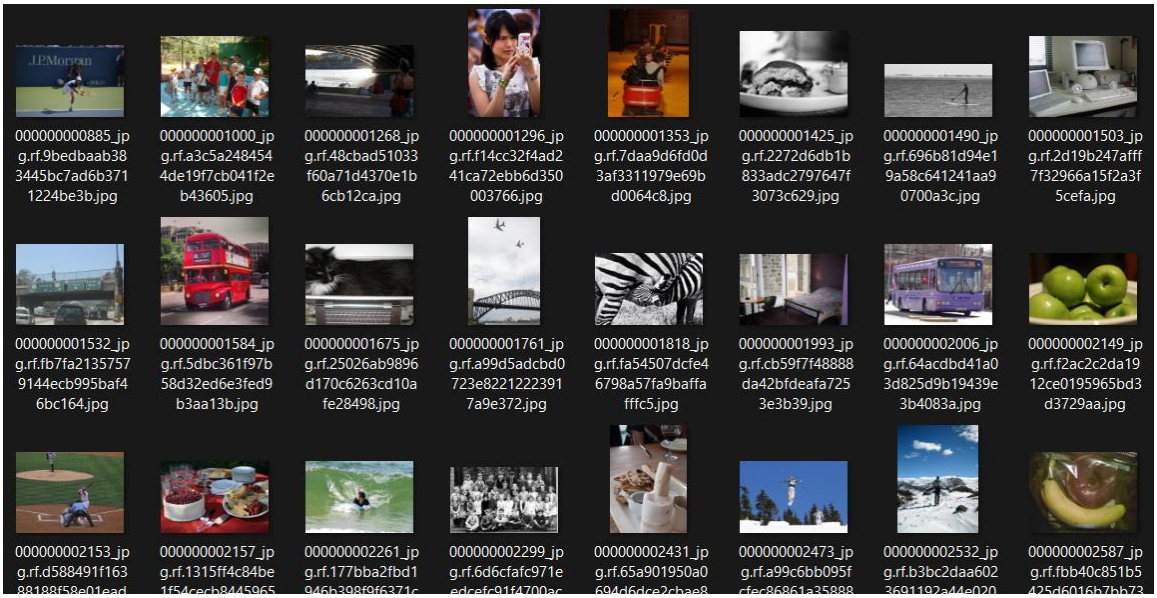


Рисунок 4.15 – Приклад зображень датасету

Також містяться 5000 текстових файлів що містять координати кожного об’єкту на певному зображенні (рис. 4.16–6.17).

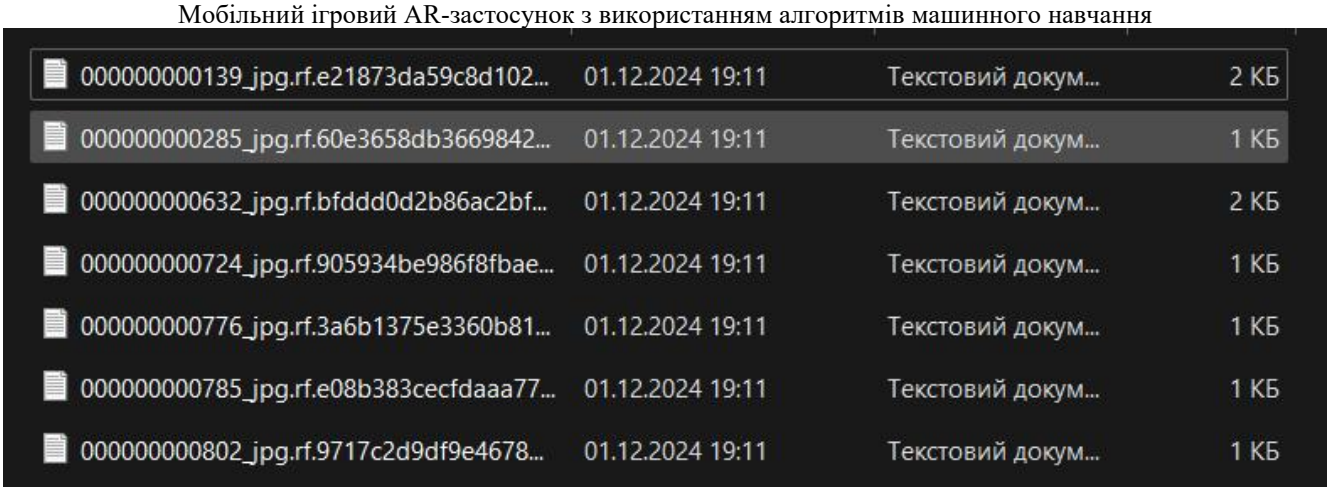


Рисунок 4.16 – Деякі текстові файли

```
50 0.38984375 0.4166666666666667 0.0390625 0.1643192488262911
75 0.12734375 0.5046948356807511 0.2328125 0.22300469483568075
75 0.934375 0.5833333333333334 0.1265625 0.18544600938967137
22 0.6046875 0.6326291079812206 0.0875 0.24178403755868544
22 0.50234375 0.6267605633802817 0.096875 0.2300469483568075
22 0.66953125 0.6185446009389671 0.046875 0.19014084507042253
22 0.5125 0.528169014084507 0.034375 0.028169014084507043
```

Рисунок 4.17 – Приклад запису координат кожного об’єкту зображенні

Також для визначення класів об’єктів є файл з назвами класів data.yaml. На прикладі першого рядку з рисунку прикладу запису координат (рис. 4.17) розберемо що означають всі ті числа:

– 50 – це індекс класу в файлі data.yaml;

– 0.38984375 0.4166666666666667 – це x та y координати верхнього лівого кута прямокутника виділення об’єкту на зображені;

– 0.0390625 0.1643192488262911 – це x та y координати нижнього правого кута прямокутника виділення об’єкту на зображені.

Написано код, що використовується для автоматизації запуску процесу навчання моделі YOLOv5 через Python-скрипт (рис. 4.18).

```python
from pathlib import Path
from subprocess import run

# Параметри навчання
img_size = 640
batch_size = 16
epochs = 50
data_yaml = 'dataset.yaml'
weights = 'yolov5s.pt'  # Предтреновані ваги

# Запуск навчання
run([
    "python", "train.py",
    "--img", str(img_size),
    "--batch", str(batch_size),
    "--epochs", str(epochs),
    "--data", data_yaml,
    "--weights", weights
])
```

Рисунок 4.18 – Приклад запису координат кожного об’єкту зображенні

Нижче наведено пояснення його складових:

– Path: модуль з бібліотеки pathlib для роботи зі шляхами до файлів і папок. У цьому коді не використовується, але може бути корисним для динамічного формування шляхів;

– run: функція з модуля subprocess, яка виконує команди в терміналі безпосередньо з Python;

– img_size: розмір зображення для вхідних даних (640x640 пікселів);

– batch_size: розмір пакету даних, що використовується на кожній ітерації. Зазвичай більший розмір вимагає більше пам'яті GPU;

– epochs: кількість епох навчання (проходів через весь датасет);

– data_yaml: шлях до YAML-файлу, який містить інформацію про датасет (тренувальні, валідаційні дані, кількість класів тощо);

– weights: шлях до попередньо тренованих вагів, які використовуються як початкові параметри для навчання;

– python train.py: вказує на виконання скрипта train.py (частина репозиторію YOLOv5);

– --img: задає розмір зображення;

– --batch: задає розмір пакету;

– --epochs: задає кількість епох;

– --data: вказує шлях до конфігураційного файлу даних;

– --weights: задає початкові ваги моделі (наприклад, yolov5s.pt);

Як працює:

1) Python запускає скрипт train.py з параметрами, переданими у вигляді списку;

2) train.py обробляє ці параметри та запускає процес навчання моделі.

Для перевірки роботи моделі в живу було написано код запуску моделі (Додаток А) та вікна камери (рис. 4.19-4.20).

Як цей код працює:

1) завантажується попередньо навчена модель YOLOv5;

2) використовується OpenCV для підключення до камери та отримання відеопотоку;

3) кожен кадр передається в модель YOLOv5, яка повертає виявлені об'єкти з координатами рамок;

4) об'єкти обводяться прямокутниками, а поруч відображається мітка з класом і рівнем впевненості.

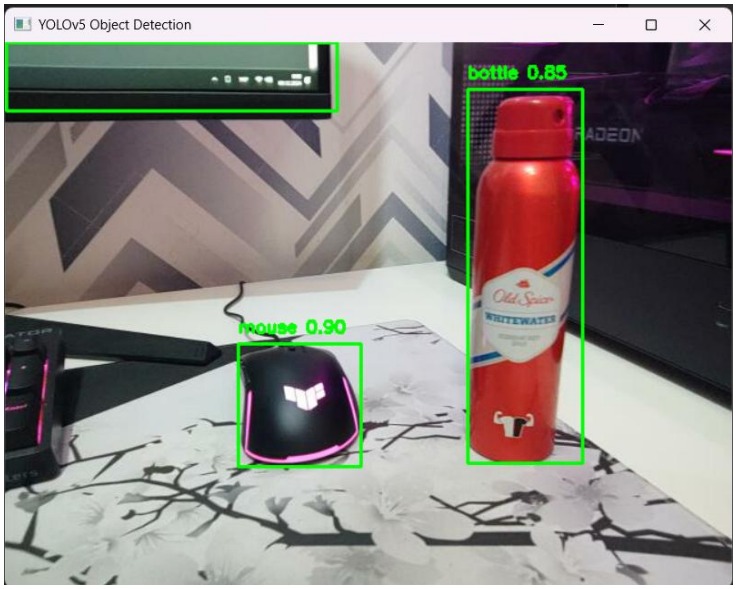


Рисунок 4.20 – Приклад запису координат кожного об’єкту зображенні

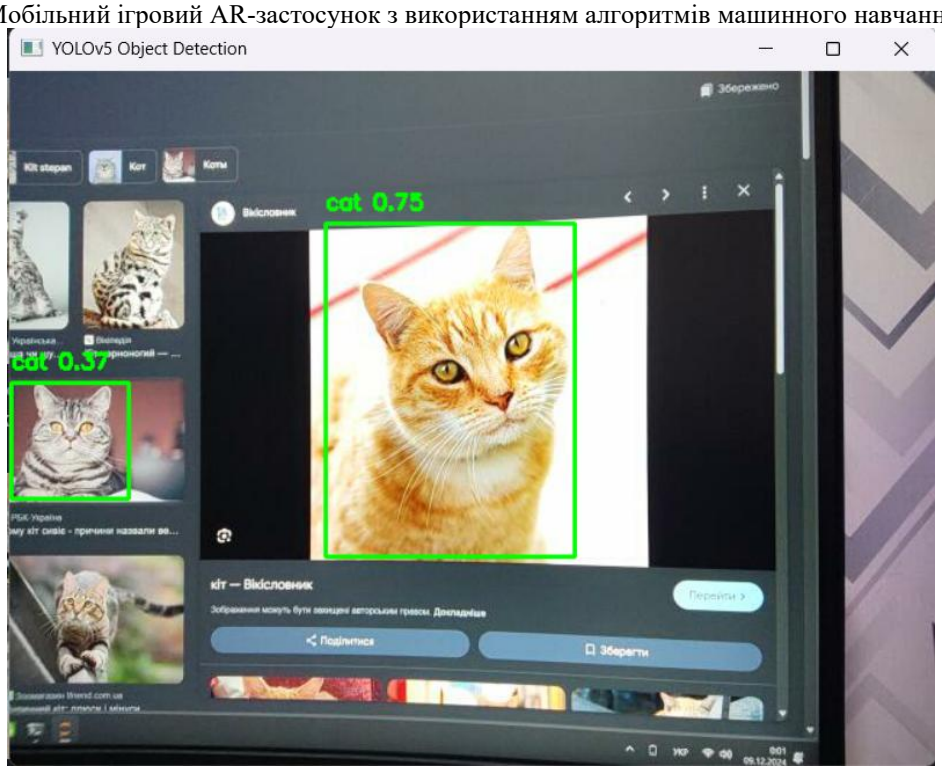


Рисунок 4.21 – Приклад запису координат кожного об’єкту зображенні

Як можна помітити модель чудово виконує свої функції та коректно розпізнає об’єкти, що дозволить коректно використовувати її у грі.

4.5 Підключення моделі

Для підключення моделі було використано конвертовану у ONNX формат модель YOLOv5. Файл міток класів моделі, плагін для роботи з комп’ютерним зором у Unity – Barracuda(рис. 4.22).

```
public NNModel modelAsset; // ONNX модель
public TextAsset labelsFile; // Файл із мітками класів
private IWorker worker; // Barracuda Worker
private string[] labels; // Список міток
```

Рисунок 4.22 – Створення змінних що містять у собі дані про модель

Наступним буде продемонстровано код підключення моделі до проєкту та налаштування камери для обробки зображення (рис. 4.23).

```
var model = ModelLoader.Load(modelAsset);
worker = WorkerFactory.CreateWorker(WorkerFactory.Type.Auto, model);

// Завантаження міток класів
labels = labelsFile.text.Split('\n');

// Налаштування веб-камери
webcamTexture = new WebCamTexture();
webcamTexture.Play();

// Створення текстури для відображення
renderTexture = new RenderTexture(webcamTexture.width, webcamTexture.height, 24);
```

Рисунок 4.23 – Код підключення моделі та камери до проєкту

Для детекції об’єктів було написано метод DetectObjects(), яка за допомогою отриманої з камери текстури та перетворює кадр із RenderTexture у формат Tensor (рис. 4.24–4.25). Виконує інференс моделі (передбачення). Аналізує результати моделі:

– визначає найбільш ймовірний клас об'єкта;

– перевіряє, чи відповідає виявлений об'єкт цільовому класу (targetClass).

Якщо об'єкт знайдено:

– викликає OnTargetObjectDetected() для обробки.

```
1 reference
void DetectObjects(RenderTexture inputTexture)
{
    // Перетворення текстури у Tensor
    var texture2D = new Texture2D(inputTexture.width, inputTexture.height, TextureFormat.RGB24, false);
    RenderTexture.active = inputTexture;
    texture2D.ReadPixels(new Rect(0, 0, inputTexture.width, inputTexture.height), 0, 0);
    texture2D.Apply();

    // Нормалізація пікселів
    var inputTensor = new Tensor(texture2D, channels: 3);

    // Виконання інференсу
    worker.Execute(inputTensor);
    Tensor output = worker.PeekOutput(); // Результат моделі

    // Інтерпретація результатів
    var scores = output.AsFloats();
    int maxIndex = scores.ToList().IndexOf(scores.Max());
    string predictedLabel = labels[maxIndex];
```

Рисунок 4.24 – Метод DetectObjects()

```
if (predictedLabel == targetClass)
{
    Debug.Log($"Target object '{targetClass}' detected with confidence {scores[maxIndex]:P2}");
    OnTargetObjectDetected(texture2D, boundingBox);
}
else
{
    Debug.Log($"Detected: {predictedLabel} with confidence {scores[maxIndex]:P2}");
}
```

Рисунок 4.25 – Частина коду що перевіряє відповідність об’єкту до класу

Метод OnTargetObjectDetected() вирізає область об'єкта (з рамки boundingBox). Створює нову текстуру для цього об'єкта. Зберігає зображення у файл (.png) у локальній файловій системі (Application.persistentDataPath). Викликає CapturePhoto() для відображення знімка (рис. 4.26).

```
void OnTargetObjectDetected(Texture2D sourceTexture, float[] boundingBox)
{
    // Витяг координат рамки
    int x = Mathf.Clamp((int)boundingBox[0], 0, sourceTexture.width);
    int y = Mathf.Clamp((int)boundingBox[1], 0, sourceTexture.height);
    int width = Mathf.Clamp((int)boundingBox[2], 0, sourceTexture.width - x);
    int height = Mathf.Clamp((int)boundingBox[3], 0, sourceTexture.height - y);

    // Вирізання області об'єкта
    Texture2D croppedTexture = new Texture2D(width, height);
    Color[] pixels = sourceTexture.GetPixels(x, y, width, height);
    croppedTexture.SetPixels(pixels);
    croppedTexture.Apply();

    // Збереження фото в файл та виведення на екран
    byte[] imageBytes = croppedTexture.EncodeToPNG();
    string filePath = Path.Combine(Application.persistentDataPath, "DetectedObject.png");
    File.WriteAllBytes(filePath, imageBytes);
    CapturePhoto();
}
```

Рисунок 4.26 – Метод OnTargetObjectDetected()

Метод CapturePhoto() захоплює весь екран як текстуру після рендерингу кадру та перетворює зображення екрана в Sprite, після чого відображає фото в UI (photoDisplayArea) і активує рамку (photoFrame)(рис. 4.27).

```
IEnumerator CapturePhoto()
{
    // Чекаємо завершення рендерингу кадру
    yield return new WaitForEndOfFrame();

    // Визначаємо область, яку потрібно зчитати (весь екран)
    Rect regionToRead = new Rect(0, 0, Screen.width, Screen.height);

    // Створюємо текстуру для збереження зображення
    Texture2D screenCapture = new Texture2D(Screen.width, Screen.height, TextureFormat.RGB24, false);

    // Зчитуємо пікселі з вказаної області екрана
    screenCapture.ReadPixels(regionToRead, 0, 0, false);

    // Застосовуємо зміни до текстури
    screenCapture.Apply();

    Sprite photoSprite = Sprite.Create(screenCapture, new Rect(0.0f, 0.0f, screenCapture.width, screenCapture.height),
        new Vector2(0.5f, 0.5f), 100.0f);
    photoDisplayArea.sprite = photoSprite;
    photoFrame.SetActive(true);
}
```

Рисунок 4.27 – Метод CapturePhoto()

Далі продемонтстровано роботу моделі у застосунку (рис. 4.28).

Рисунок 4.28 – Приклад запису координат кожного об’єкту зображенні

Таким чином було підключено та налаштовано роботу моделі розпізнавання об’єктів на основі моделі YOLOv5 у мобільному застосунку.

4.6 Тестування гри

Функціональне тестування – це процес перевірки того, чи реалізовані функції відповідають визначеним функціональним вимогам. Тести було проведено на пристрої Xiaomi 11 Lite з операційною системою Android.

Тест 1. Купівля предметів у магазині (рис. 4.29).

Вхідні дані: На сторінці ігрового магазину натиснути на кнопку BUY.

Очікуваний результат: Кількість монет спишеться а у інвентарі з’явиться необхідний предмет.

Отриманий результат: Результат співпадає з очікуваннями (рис. 4.30).

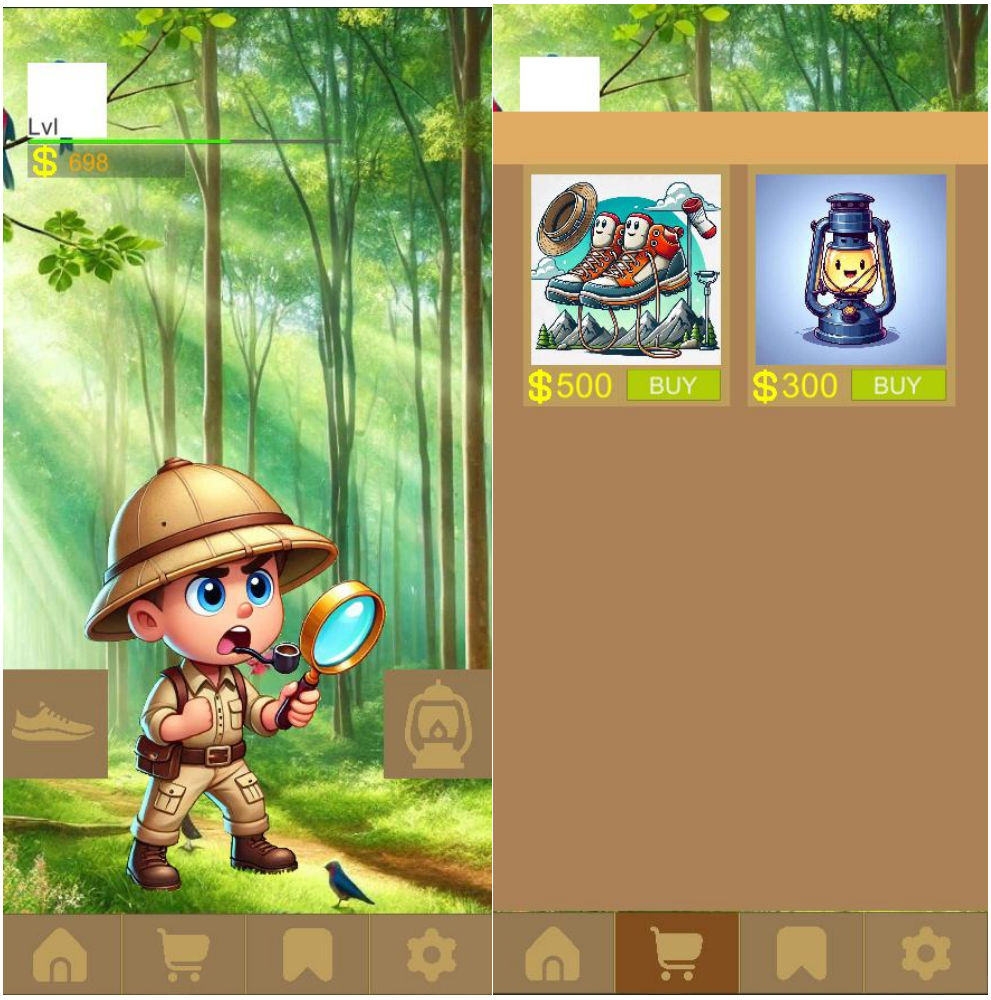


Рисунок 4.29 – Сторінка магазину та інвентаря

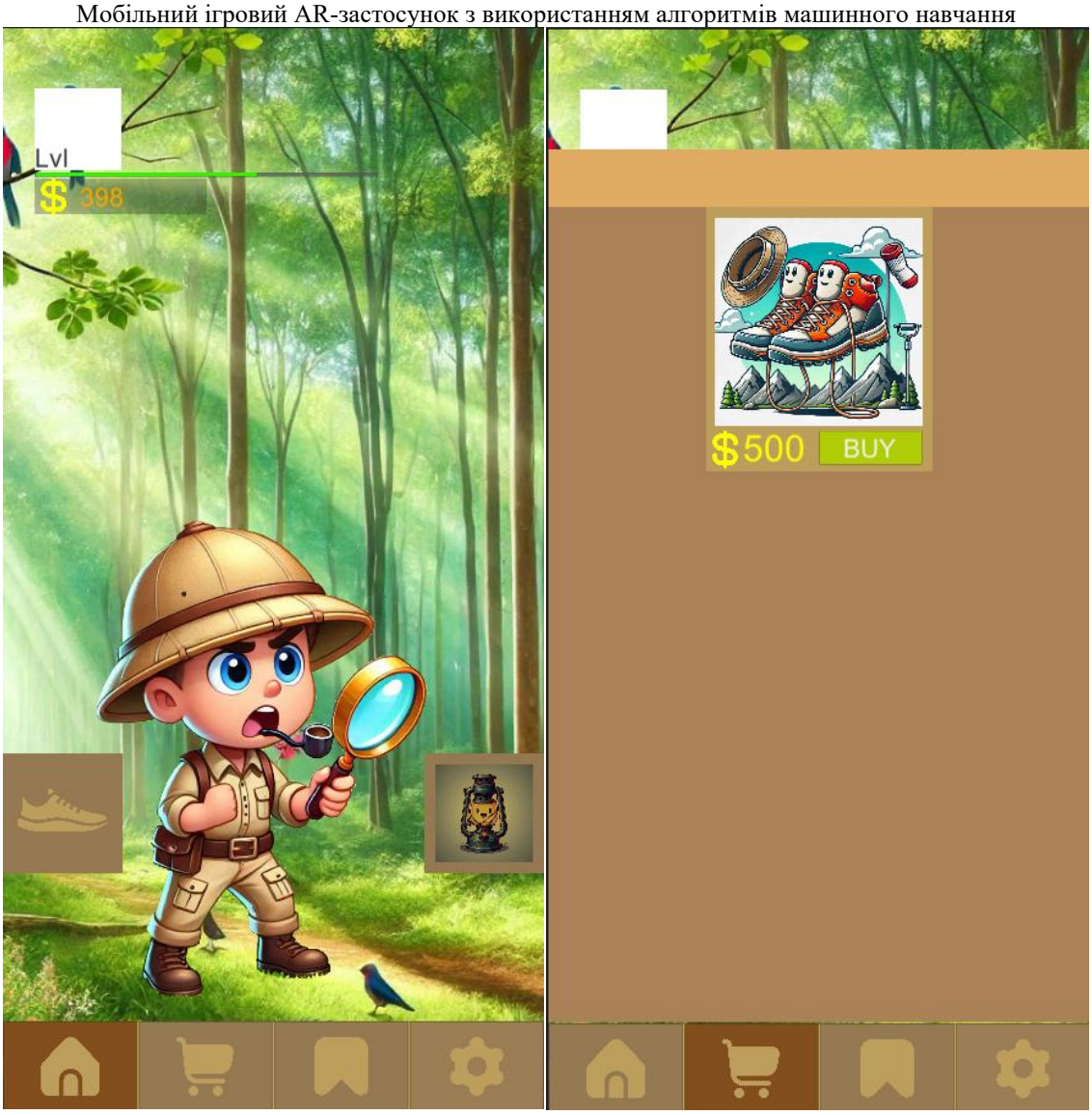


Рисунок 4.30 – Сторінка магазину та інвентаря

Тест 2. Підрахунок кроків

Вхідні дані: користувач йде.

Очікуваний результат: зарахування кроків.

Отриманий результат: співпадає з очікуваним (рис. 4.31).

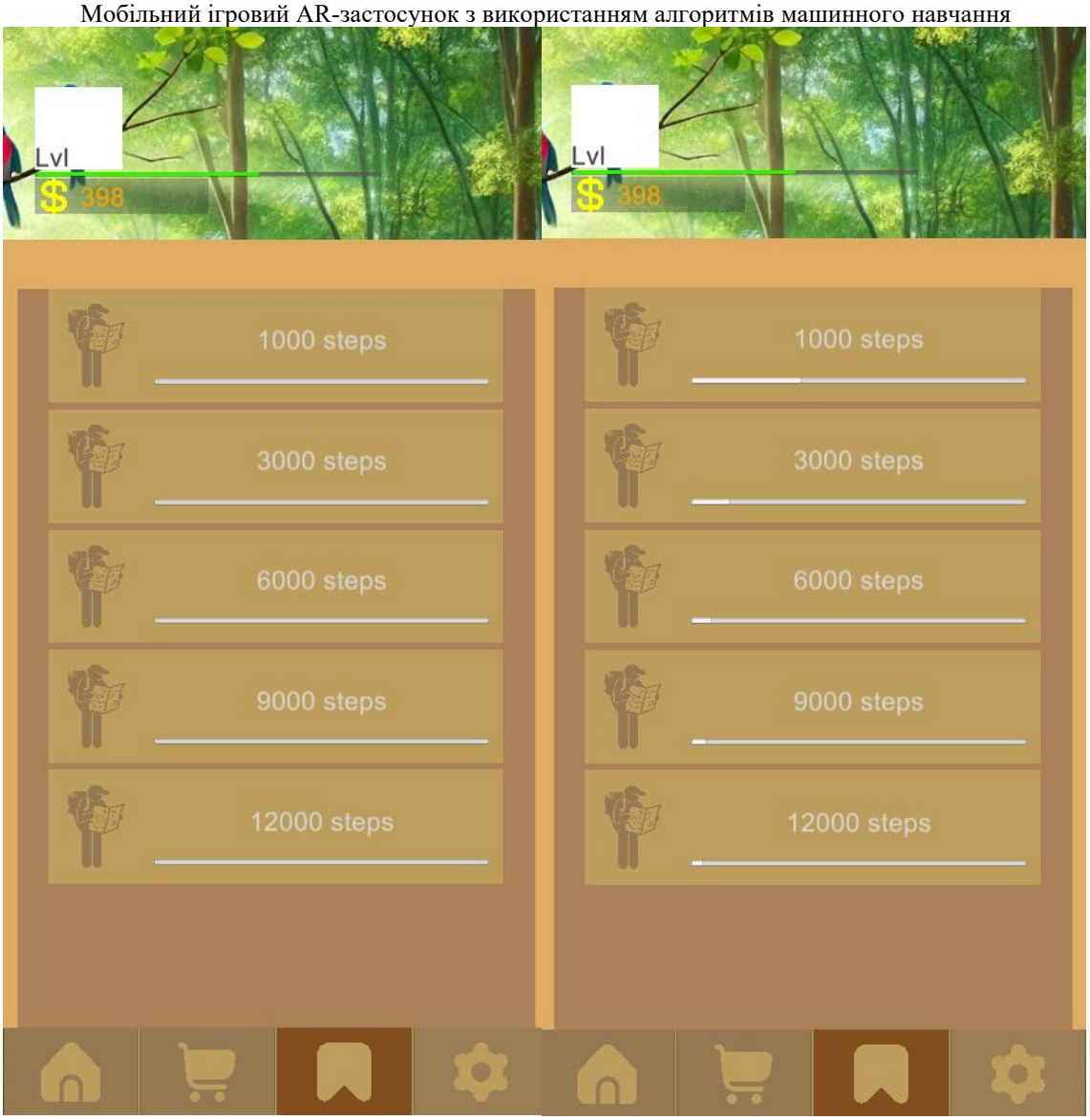


Рисунок 4.31 – Підрахунок кроків користувача

Тест 3. Тестування розпізнавання об’єктів.

Вхідні дані: гравець обрає квест та показує необхідний об’єкт.

Очікуваний результат: гра розпізнає об’єкт та зберігає його зображення у паралоїд.

Отриманий результат: співпадає з очікуваним (рис. 4.32).

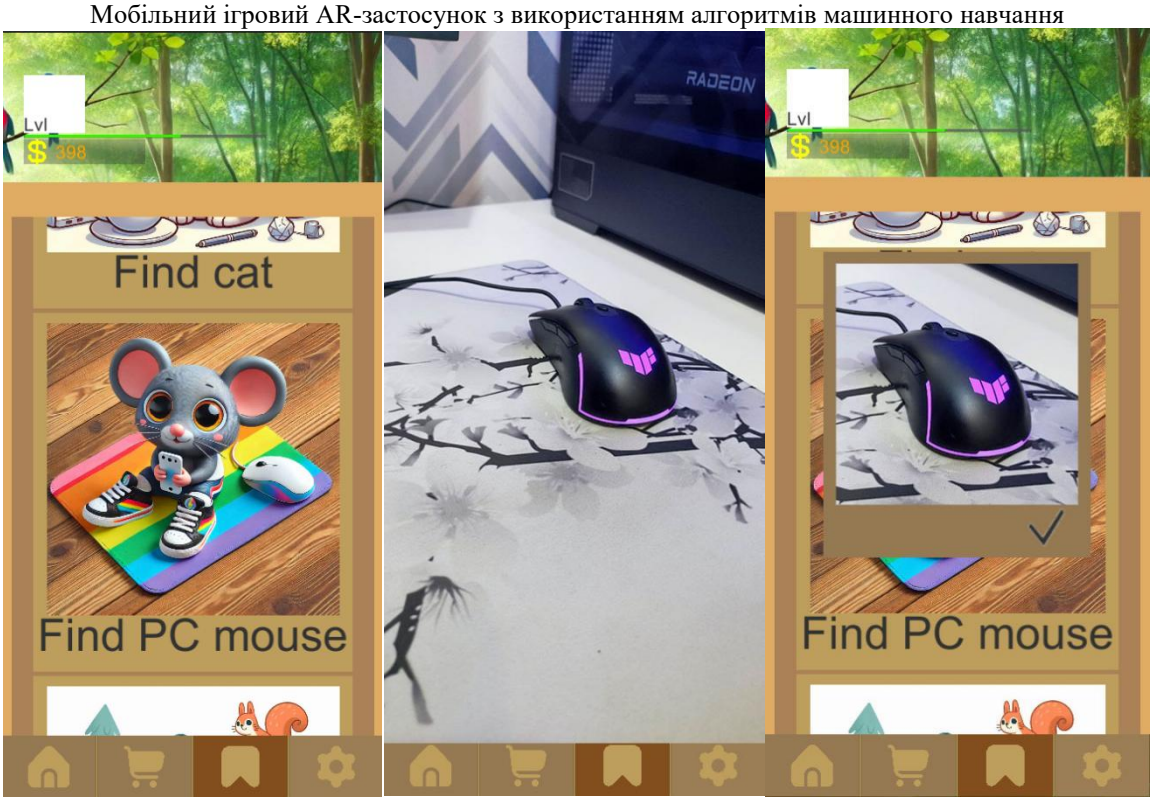


Рисунок 4.32 – Постріл гравця

Всі проведені тестування мали успішний результат.

Висновки до розділу 4

У ході роботи було створено мобільну гру з використанням технологій доповненої реальності та машинного навчання. Розроблено адаптивний інтерфейс, систему персонажів, магазин із динамічним наповненням та інтерактивні квести. Для впровадження функцій розпізнавання об’єктів інтегровано модель YOLOv5, яка показала високу точність у реальному часі. Проведене тестування підтвердило стабільність роботи гри та її відповідність поставленим цілям, забезпечивши цікавий ігровий досвід.

ВИСНОВКИ

У межах магістерської роботи було досліджено та реалізовано підхід до інтеграції алгоритмів машинного навчання у мобільні ігрові застосунки доповненої реальності (AR). Проведений аналіз довів, що поєднання AR-технологій із машинним навчанням відкриває значні можливості для створення інтелектуальних ігрових середовищ, які адаптуються до поведінки та потреб користувача.

Розроблений прототип мобільного ігрового ar-застосунку продемонстрував ефективність використання алгоритмів машинного навчання для підвищення функціональності застосунку. В процесі роботи виконано такі завдання:

– проаналізовано сучасні рішення у сфері мобільних ігрових AR-застосунків;

– досліджено алгоритми машинного навчання, придатні для інтеграції в AR-середовища;

– розроблено концепцію застосунку, яка враховує вимоги до інтерактивності та персоналізації;

– реалізовано програмний прототип із використанням технологій комп’ютерного зору та глибинного навчання;

– проведено тестування функціональності та оцінено залученість користувачів.

Результати роботи підтвердили доцільність використання машинного навчання для покращення взаємодії між користувачем і віртуальними об'єктами. Модель навчилася адаптуватися до поведінки гравців, що забезпечило підвищення рівня залученості та покращило користувацький досвід.

Отримані результати також вказують на перспективність застосування інтегрованих AR-рішень у різних галузях, таких як освіта, бізнес, медицина та розваги. Використання машинного навчання в AR-системах дозволяє автоматизувати персоналізацію контенту, покращувати точність розпізнавання об'єктів та розширювати межі інновацій.

Таким чином, виконана робота не лише розв'язала поставлені завдання, але й заклала основу для подальших досліджень у сфері мобільних AR-застосунків із використанням алгоритмів машинного навчання. Результати можуть бути використані для розробки подібних інтерактивних рішень, спрямованих на задоволення потреб сучасного цифрового суспільства.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Громова І. Популярність мобільних ігор зростає: як це вплине на бізнес та відносини з клієнтами?. Speka - онлайн медіа про технології та підприємництво URL: https://speka.media/u-2030-roci-29-mlrd-lyudei-budut-grati-v-mobilni-igri-yak-ce-vpline-na-biznes-ta-vidnosini-z-klijentami-p630zp (дата звернення: 08.09.2024).

2. Niantic Inc. Pokémon GO. Google Play. URL: https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=uk (Last access: 08.09.2024).

3. The location based RPG games developer (Geo GPS). Magic Streets: The GPS realm. Google Play. URL: https://play.google.com/store/apps/details?id=com.builditgames.magicstreets&hl=uk (Last access: 08.09.2024).

4. Hofli Limited. Cosmic Frontline AR. Google Play. URL: https://play.google.com/store/apps/details?id=com.hofli.cosmicfrontline&hl=uk (Last access: 08.09.2024).

5. Use case diagram. StarUML documentation. URL: https://docs.staruml.io/working-with-uml-diagrams/use-case-diagram (Last access: 11.09.2024).

6. Sequence diagram. StarUML documentation. URL: https://docs.staruml.io/working-with-uml-diagrams/sequence-diagram (Last access: 11.09.2024).

7. Communication Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/communication-diagram (Last access: 11.09.2024).

8. Statechart Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/statechart-diagram (Last access: 20.03.2024).

9. Online Mockup. Wireframe & UI Prototyping Tool. Moqups. URL: https://moqups.com/ (Last access: 12.09.2024).

10. Табунщик Г. В., Каплієнко Т. І., Петрова О. А. Життєвий цикл та розроблення інформаційних систем. *Проектування та моделювання програмного забезпечення сучасних інформаційних систем*. Запоріжжя, 2016. С. 8–12.

11. Class diagram. StarUML documentation. URL: https://docs.staruml.io/working-with-uml-diagrams/class-diagram (Last access: 02.04.2023).

12. Component Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/component-diagram (Last access: 22.03.2023).

13. Profile Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/profile-diagram (Last access: 26.03.2023).

14. Package Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/package-diagram (Last access: 30.03.2023).

15. Chan J. Chapter 1: Introduction to C#. *Learn C# in one day and learn it well : C# for beginners with hands-on project : the only book you*. United States, 2017. P. 10–11.

16. Що таке Python?. *aCode*. URL: https://acode.com.ua/intro-python/ (дата звернення: 09.12.2024).

17. Learn. *Unity*. URL: https://unity.com/learn (Last access: 03.05.2023).

18. Photoshop. *Adobe*. URL: https://www.adobe.com/ua/products/photoshop.html (Last access: 04.05.2023).

Додаток А

Код демонстрації роботи моделі

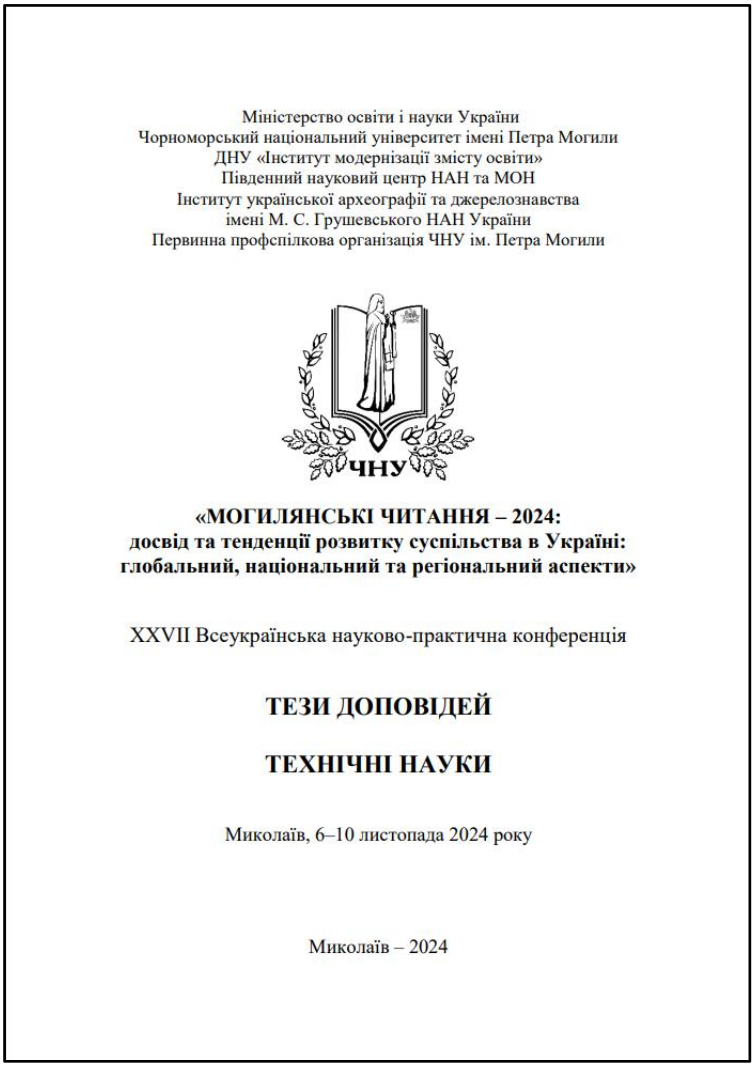
```
import cv2
import torch
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# Завантаження моделі YOLOv5
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

# Ініціалізація камери
cap = cv2.VideoCapture(0)  # 0 - це перша камера, змініть номер, якщо потрібно
if not cap.isOpened():
    print("Не вдалося відкрити камеру")
    exit()

try:
    while True:
        # Зчитування кадру з камери
        ret, frame = cap.read()
        if not ret:
            print("Не вдалося зчитати кадр")
            break

        # Перетворення кадру в формат, прийнятний для моделі YOLOv5
        results = model(frame)

        # Отримання даних для виявлених об'єктів
        detections = results.xyxy[0].cpu().numpy()  # Координати рамок, клас,
впевненість

        # Перебір виявлених об'єктів і малювання рамок
        for det in detections:
            x1, y1, x2, y2, conf, cls = det
            label = f"{model.names[int(cls)]} {conf:.2f}"
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255,
0), 2)
            cv2.putText(frame, label, (int(x1), int(y1) - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        # Відображення кадру
        cv2.imshow("YOLOv5 Object Detection", frame)

        # Завершення роботи при натисканні клавіші 'q'
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

finally:
    # Закриття камери та вікна
    cap.release()
    cv2.destroyAllWindows()
```

Додаток Б

Апробація кваліфікаційної роботи

Результати досліджень були представлені на конференції:

Могилянські читання – 2024 : досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти. Технічні науки : XXVII Всеукр. наук.-практ. конф. : 6–10 листоп. 2024 р., м. Миколаїв : тези / М-во освіти і науки України ; ЧНУ ім. Петра Могили ; ДНУ «Ін-т модернізації змісту освіти» ; Півд. наук. центр НАН та МОН України ; Ін-т укр. археографії та джерелознавства ім. М. С. Грушевського НАН України; Первинна профспілкова орг. ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024.



Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Рисунок Б.1 – Обкладинка збірника тез конференції Могилянські читання – 2024