МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. тех. наук,
доцент_____ Є. О. Давиденко

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

МОБІЛЬНИЙ ІГРОВИЙ ЗАСТОСУНОК З ВИКОРИСТАННЯМ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ

Спеціальність «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ Юрій АФОНІН
підпис
«__» _____ 2024 р.

Керівник канд. тех. наук, доцент _____ Євген ДАВИДЕНКО
підпис
«__» _____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

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

_____ Є. О. Давиденко

«_____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу здобувача

_____Афоніна Юрія Сергійовича_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи:

Мобільний ігровий застосунок з використанням алгоритмів штучного інтелекту_____

Затверджена наказом по ЧНУ від «09» вересня 2024 р. № __220__

2. Строк представлення кваліфікаційної роботи «____» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є прототип мобільного ігрового застосунку із використанням алгоритмів штучного інтелекту. Початково: мобільний пристрій.

4. Перелік питань, що підлягають розробці

− дослідження предметної галузі та аналіз застосунків-аналогів;

− визначення проблем та пошук рішень;

– вибір ігрової тематики, стилю, жанру;

− моделювання та проєктування ПЗ;

− розробка функціоналу ігрового застосунку;

− впровадження алгоритмів ШІ в ігровий процес;

− тестування прототипу ігрового застосунку;

− аналіз результатів розробки;

5. Перелік графічних матеріалів

Презентація.

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Євген ДАВИДЕНКО
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Юрій АФОНІН
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «03» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Мобільний ігровий застосунок з використанням алгоритмів штучного інтелекту_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	02.09.2024	04.09.2024	виконано
2.	Огляд літератури за темою роботи	05.09.2024	13.09.2024	виконано
3.	Складання календарного плану КМР	16.09.2024	17.09.2024	виконано
4.	Аналіз предметної області	17.09.2024	20.09.2024	виконано
5.	Визначення проєктних проблем та пошук рішень	23.09.2024	25.09.2024	виконано
6.	Моделювання та проєктування ПЗ	26.09.2024	02.10.2024	виконано
7.	Розробка функціоналу ігрового застосунку, тестування та апробація розробленого ПЗ, аналіз результатів тестування	03.10.2024	22.11.2024	виконано
8.	Оформлення КМР та презентації	25.11.2024	27.11.2024	виконано
9.	Попередній захист	02.12.2024	02.12.2024	виконано
10.	Завершення оформлення КМР та презентації	03.12.2024	12.12.2024	виконано
11.	Відгук керівника КМР	13.12.2024	16.12.2024	
12.	Рецензування	13.12.2024	17.12.2024	
13.	Захист кваліфікаційної роботи	19.12.2024	20.12.2024	

Здобувач _____ **Юрій АФОНІН**

«__» _____ 20__ р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Мобільний ігровий застосунок з використанням алгоритмів штучного інтелекту»

Здобувач гр. 608м: Афонін Юрій

Керівник: канд. техн. наук, доцент Давиденко Євген

Дана робота присвячена розробці мобільного ігрового застосунку з використанням алгоритмів штучного інтелекту з використанням ігрового рушія Unity.

Об’єктом роботи є процес моделювання та розробки мобільного ігрового застосунку з використанням алгоритмів штучного інтелекту та машинного навчання.

Предметом роботи є програмні засоби розробки мобільних ігрових застосунків на основі рушія Unity з використанням новітніх підходів та технологій у галузі штучного інтелекту (AI) та машинного навчання (ML).

Метою кваліфікаційної роботи є оптимізація, підвищення продуктивності та покращення ігрового досвіду при створенні казуальної гри завдяки впровадженню алгоритмів штучного інтелекту та машинного навчання.

Для досягнення було необхідно виконати такі **завдання**:

– проаналізувати предметну область;

– обрати ігрову тематику, стиль, жанр;

– визначити функції програмного забезпечення (ПЗ), специфікацію вимог та сценарії використання;

– спроєктувати застосунок, створити UML-діаграм;

– обґрунтувати та впровадити алгоритми штучного інтелекту в ігровий процес;

– створити машинні моделі з використанням Unity ML Agents;

– інтегрувати open-source AI плагіни;

– створити прототип мобільного ігрового застосунку.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків та переліку джерел посилання.

У вступі описується актуальність теми, визначається мета, об’єкт, предмет роботи та необхідні завдання.

У першому розділі проводиться аналіз предметної галузі, аналогів застосунків з використанням алгоритмів штучного інтелекту, їх переваг і недоліків, формується специфікація вимог до програмного забезпечення, що розробляється.

Другий розділ містить детальний опис ігрового процесу застосунку із наведенням сценаріїв використання, діаграм діяльності та інших засобів конструювання програмного забезпечення.

Третій розділ містить діаграму класів, особливості розробки мобільного ігрового застосунку та організацію Unity-проєкту.

Четвертий розділ містить опис розробки ігрового застосунку (кодування), скриншоти коду, ігрового дизайну та локацій, результати тестування гри.

Висновки містять аналіз проведених робіт та результатів.

КМР викладена на 103 с., містить 4 розділи, 8 табл., 77 рис., 4 дод., 28 джерел.

Ключові слова: *казуальна гра, ігровий рушій Unity, алгоритми штучного інтелекту, Unity ML, Unity Open AI.*

ABSTRACT

of the Master`s Thesis

"**Mobile game application using artificial intelligence algorithms**"

Student of group 608m: Afonin Yurii

Supervisor: Candidate of Tech. Sc. (Eng.), Docent, Davydenko Yevhen

This work is dedicated to the development of a mobile game application using artificial intelligence algorithms with the Unity game engine.

The object of work is a process of modeling and developing a mobile game application using artificial intelligence (AI) and machine learning (ML) algorithms.

The subject of work is the software tools for developing mobile game applications based on the Unity engine, utilizing the latest approaches and technologies in the field of artificial intelligence and machine learning.

Objective: optimization, enhancing and improvement of the gaming experience in creating a casual game by implementing artificial intelligence and machine learning algorithms.

To achieve the objective, it was necessary to complete the following tasks:

- analyze the subject area;
- select the game theme, style, and genre;
- define the software functions, requirements specification, and use cases;
- design the application and create UML diagrams;
- justify and implement artificial intelligence algorithms in the gameplay;
- develop machine models using Unity ML Agents;
- integrate open-source AI plugins;
- create a prototype of a mobile gaming application.

The qualification work of the bachelor consists of an introduction, four chapters, conclusions, and a list of sources references.

The introduction defines the relevance of the topic, sets up the object, work objective, subject of research, and brief tasks overview.

The first chapter determines the analytics part of researched app subject area and includes a comparison with similar apps using artificial intelligence algorithms, also their advantages and disadvantages. The last part of the chapter describes the formation of requirements specifications for the developed software.

The second chapter includes a detailed description of the gameplay process, including use cases, activity diagrams, and other software design tools.

The third chapter includes a class diagram, the features of mobile game application development and the organization of the Unity project.

The fourth chapter contains the description of game application development (coding), screenshots of the code, game design, and locations, as well as the results of game testing.

The conclusions analyze the work and obtained results.

The qualification work of the master is presented on 103 pages, it contains 4 chapters, 8 tables, 77 figures, 4 appendices, 28 sources in the list of references.

Keywords: *casual game, Unity game engine, artificial intelligence algorithms, Unity ML, Unity Open AI.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ	–	програмне забезпечення
ЦА	–	цільова аудиторія
ОС	–	операційна система
ТЗ	–	технічне завдання
ШІ	–	штучний інтелект
A*	–	A-star
AI	–	artificial intelligence
AR	–	augmented reality
API	–	application programming interface
BFS	–	broad-first search
DFS	–	depth-first search
DI	–	dependency injection
DL	–	deep learning
DRL	–	deep reinforcement learning
GPT	–	general pre-trained transformer
ML	–	machine learning
MLA	–	machine learning algorithm
NPC	–	non-playable character
POV	–	point of view
RPG	–	role play game
VR	–	virtual reality
UML	–	unified modeling language
UI	–	user interface
UPM	–	Unity package manager
UX	–	user experience
XR	–	extended reality

ВСТУП

Стрімке впровадження штучного інтелекту в усіх галузях інформаційних технологій протягом останнього десятиліття поступово збільшує частку програмних застосунків, що тим чи іншим чином використовують алгоритми ШІ. Розробка ігрових застосунків з використанням алгоритмів штучного інтелекту, як ніколи раніше, стала важливим напрямком у розробці програмного забезпечення [1].

Сфери бізнесу, науки, комерції та розваг потребують новітніх ефективних та нестандартних програмних рішень, які візуалізовують ту чи іншу проблематику та заохочують кінцевих користувачів до вибору цих рішень. Для вирішення подібних задач існує немало готових інструментів та ігрових рушіїв, які дозволяють задовольняти потреби замовників. Одним із найпопулярніших рушіїв із вбудованими інструментами для використання таких технологій є платформа Unity.

Актуальність теми кваліфікаційної магістерської роботи зумовлена останніми тенденціями впровадження алгоритмів штучного інтелекту в провідних галузях інформаційних технологій. Проводячи аналіз ринку та запитів суспільства (в т.ч. ігрових спільнот), можна дійти до логічного висновку щодо доцільності впровадження таких алгоритмів та технічних рішень, з метою підвищення ефективності ігрових застосунків, покращення ігрового досвіду користувачів та оптимізацією процесів розробки програмного забезпечення.

Об’єктом роботи є процес моделювання та розробки мобільного ігрового застосунку з використання алгоритмів штучного інтелекту та машинного навчання.

Предметом роботи є програмні засоби розробки мобільних ігрових застосунків на основі рушія Unity з використанням новітніх підходів та технологій у галузі штучного інтелекту (AI) та машинного навчання (ML).

Метою є оптимізація, підвищення продуктивності та покращення ігрового досвіду при створенні казуальної гри завдяки впровадженню алгоритмів штучного інтелекту та машинного навчання. Для досягнення мети необхідно виконати такі **завдання**:

- аналіз предметної області;
- вибір ігрової тематики, стилю, жанру;
- визначення функцій програмного забезпечення (ПЗ), специфікації вимог та сценаріїв використання;
- проєктування застосунку, створення UML-діаграм;
- обґрунтування та впровадження алгоритмів штучного інтелекту в ігровий процес;
- створення машинних моделей з використанням Unity ML Agents;
- інтеграція open-source AI плагінів;
- створення робочого прототипу мобільного ігрового застосунку.

Практичне застосування: ігровий застосунок можна використовувати для розваг, навчання та відпочинку. З точки зору програмного рішення, застосунок можна використовувати для освітніх цілей в якості наочного прикладу впровадження алгоритмів штучного інтелекту в ігрові процеси.

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання – 2024», Миколаїв, 06-10 листопада 2024 р. (Додаток Г) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ

1.1 Огляд методів та алгоритмів штучного інтелекту в ігрових застосунках

Першочерговим завданням перед прийняттям рішень щодо розробки програмного забезпечення є ознайомлення із предметною областю, оглядом технологій та алгоритмів.

Штучний інтелект використовується абсолютно всюди, де використовується інженерія ПЗ – від супутникових станцій до розумних зубних щіток. Індустрія відеоігор одна із перших наочно продемонструвала як алгоритми штучного інтелекту можна використовувати та поєднувати у різноманітних галузях, і не лише для ігор.

За допомогою провідних ігрових рушіїв, кінокомпанії здатні створювати реалістичні «масові батальні» сцени для фільмів, де тисячі «юнітів» поводяться згідно з прописаними алгоритмами ШІ, при цьому економлячи величезні суми на зйомці. Використання доповненої та віртуальної реальності з підтримкою алгоритмів ШІ сьогодні є невід’ємною частиною військової підготовки (симулятори безпілотних літальних комплексів, дронів, техніки тощо), що десятиліття тому можна було побачити лише у ігрових шутерах та стратегіях. Що ж тоді казати про тотальну гейміфікацію усіх сфер нашого життя? Здоров’я, освіта, розваги – кожен стикався із рекламою сотень ігрових застосунків з алгоритмами ШІ у ігрових магазинах.

Більшість ігор, якщо не всі, використовують загальні методи та алгоритми штучного інтелекту для ігрової логіки. Розглянемо типи ігрового ШІ, із них можна виділити такі [14]:

1. Rule-based AI – поведінкова логіка (за правилами, інструкціями), яка зазвичай описується розробниками, використовуються усюди. Агенти взаємодіючи з певними сутностями (тригерами, іншими сутностями) та

виконуючи (чи не виконуючи) певний набір інструкцій (правил) приймають рішення та виконують дії.

2. Finite State Machines – скінченні автомати (машини станів) ще один яскравий представник інтелекту для NPC, який використовується у геймдеві. Полягає у взаємодії та переході між станами сутностей в залежності від певних умов.

3. Pathfinding AI – алгоритми пошуку шляху є одними із найважливіших у сфері геймдеву, вони використовуються усюди, A*, BFS, DFS та інші є представниками цих алгоритмів.

4. Behavior trees – поведінкові дерева, цей ієрархічний підхід дозволяє з'єднувати «ноди» (дії, стани) між собою на основі певних правил (умов), цим самим генеруючи поведінку NPC та рішення, які він буде приймати на основі вхідних даних.

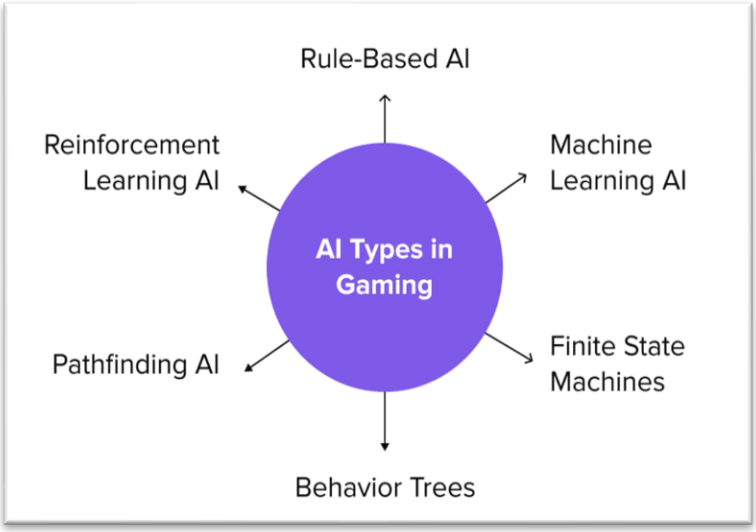


Рисунок 1.1 – Схематичне представлення розгалудження типів ШІ в ігрових застосунках

Штучний інтелект досить загальне поняття, і до нього можна віднести багато інструментів, технологій та алгоритмів, машинне навчання, нейронні мережі та еволюційні алгоритми – одні із них. Розглянемо, як вони можуть використовуватись в ігрових застосунках.

Методи машинного навчання

Більш специфічні ігрові застосунки зазвичай включають в себе алгоритми та методи машинного навчання. Вони значно спрощують розробку та налаштування поведінки ігрових NPC (агентів) на основі тренувальних даних або тренувальної симуляції. Під час «тренування» ШІ-моделі ігрових NPC навчаються поводитись відповідно до вимог та заданих обставин.

На рис. 2 наведено найпоширеніші сценарії використання ML в ігрових застосунках [20].

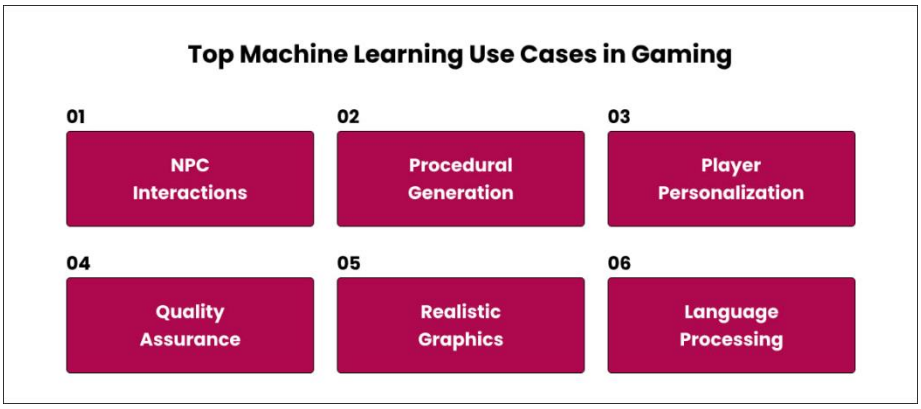


Рисунок 1.2 – Найпоширеніші сценарії використання ML у іграх

Отже, до методів ML належать такі:

- **supervised learning** – моделі навчаються на основі маркованих даних, де кожен приклад має відомий вихідний результат;
- **unsupervised learning** – моделі навчаються на немаркованих даних, зазвичай для виявлення прихованих залежностей або групування даних;
- **semi-supervised learning** – комбінація навчання з поєднанням маркованих та немаркованих даних;
- **reinforcement learning** – моделі навчаються шляхом взаємодії з динамічними середовищами та отримання винагород за правильні дії [2; 14];
- **imitation learning** – моделі вчаться взаємодіяти між собою реагуючи на поведінку інших агентів.

Нейронні мережі

Ігри з більш вузьким призначенням можуть використовувати нейронні мережі та глибинне машинне навчання для власних цілей, зазвичай це ігри, що моделюють певну симуляцію, пошук або використовуються для демонстрації певної проблематики у різних промислових та наукових галузях.

– **Convolutional Neural Networks (CNN)** – ефективні для обробки зображень та розпізнавання образів.

– **Recurrent Neural Networks (RNN)** – використовується для обробки послідовних даних, таких як текст або часові ряди.

– **Deep Neural Networks (DNN) –** мережі з багатьма прихованими шарами, здатні моделювати складні залежності в даних.

Сьогодні популярним серед інді-розробників ігор стало використовувати open-source інструменти, які дозволяють додати різнобарвності ігровому процесу.

Рисунок 1.3 – Логотип організації OpenAI

Наприклад, неофіційний плагін-інтеграція OpenAI у ігровому рушії Unity, який через запити до їх API дозволяє взаємодіяти із ChatGPT та використовувати їх для різноманітних цілей (квести, діалоги, speech-to-text, завдання тощо) [21].

Еволюційні алгоритми

Ігри, які мають певну прогресію та «еволюцію» гравця можуть використовувати генетичні алгоритми для збільшення складності, зміни поведінки NPC, тощо.

– **Genetic Algorithms** – імітують процес еволюції для оптимізації проблем.

– **Genetic Programming** – використовує еволюційний підхід для автоматичного породження комп'ютерних програм.

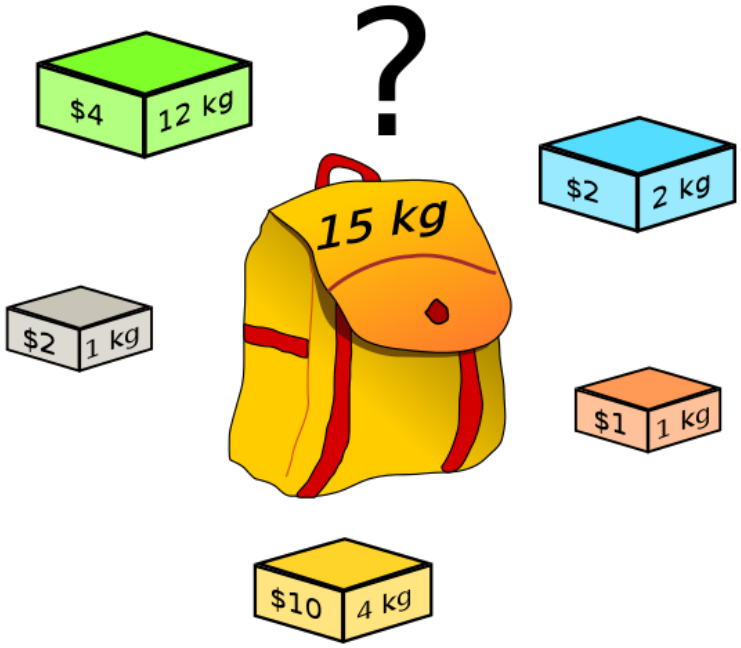


Рисунок 1.4 – Задача пакування (англ. “Knapsack problem”)

Найпростішими прикладами використання еволюційних алгоритмів для ігрових застосунків можуть бути такими:

- створення режиму автоматичного будівництва;
- вибір найменшої вартості набору, автоматичне розташування предметів в інвентарі («задача пакування»);
- оптимізація рішень ШІ при автоплануванні (стратегічні ігри)

Такі задачі є прямими проблемами генетичних алгоритмів і чудово підходять при розробці ігрових застосунків.

Розширена реальність

Штучний інтелект не включає «реальності» напряму, проте, завдяки машинному зору, класифікації об’єктів та їх розпізнаванню, вони є чудовим тандемом. Алгоритми розширеної реальності (xR), що дозволяють використовувати віртуальну (VR), доповнену (AR) чи змішану (MR) реальність стали надзвичайно популярними серед гравців ігор, які використовують подібні технології. Окрім сфери розваг, великим попитом на **AR/VR/MR** користуються у сферах: військових технологій, медицини, промисловості, освіти та науки [11].

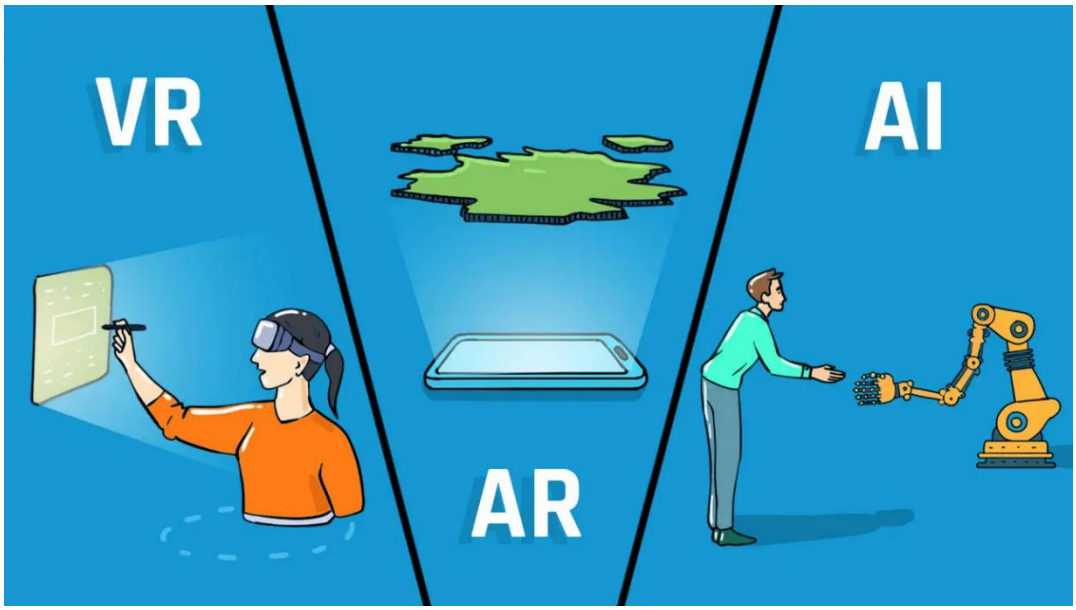


Рисунок 1.5 – Поєднання xR та AI

В основі цих «**трьох китів реальностей**» лежить комп’ютерний (машинний) зір та взаємодія із реальним світом за допомогою контроллерів (приймачами користувацького вводу). За допомогою алгоритмів штучного інтелекту, які симулюють та перетворюють на зображення реальні поверхні чи класифікують об’єкти (визначають форму, назву тощо) можна отримати чудовий користувацький досвід від ігрового застосунку.

1.2 Огляд застосунків-аналогів

Після аналізу технологій, проаналізуємо аналоги застосунків, що використовуються алгоритми ШІ. Для аналізу аналогів було обрано застосунки AI Dungeon (табл. 1.1), The Elder Scrolls: Blades (табл. 1.2) та Shadowverse (табл. 1.3).

AI Dungeon

Застосунок створений для унікальної взаємодії користувача зі штучним інтелектом на базі GPT. Застосунок є текстовою грою, де гравець вводить бажані дані про свого персонажа і застосунок генерує історію та розвиває її в залежності від прийнятих рішень гравця.

Таблиця 1.1 – Опис застосунку «AI Dungeon»

Назва	**AI Dungeon**
Виробник:	Latitude
Архітектура	Мобільний застосунок для Android, iOS, вебзастосунок, Standalone платформи (Windows, Linux, macOS)
Мова реалізації	Невідомо
Функції	1. Вибір сеттингу гри. 2. Генерація історії під обрану тему. 3. Взаємодії з іншими гравцями через мультиплеєр. 4. Генерація зображень на основі ігрової тематики та історії. 5. Підтримка обміну контентом між гравцями.
Переваги	1. Безкоштовний. 2. Кросплатформеність. 3. Напряму взаємодіє з AI (геймплей побудований на генеративному ШІ).

Кінець таблиці 1.1

Недоліки	1. Деякі користувачі відмічають проблеми з точністю вимірювань та синхронізацією даних з іншими додатками. 2. Відсутність активного ігрового світу, лише текстова взаємодія з AI. 3. Присутні проблеми з модерацією шкідливого контенту.
Вебсайт	https://aidungeon.com/

Рисунок 1.6 – Логотип AI Dungeon

The Elder Scrolls: Blades

Застосунок є мобільною грою у жанрі action RPG від першої, заточена під мобільні екрани. Дозволяє кастомізовувати свого персонажа та локації, відома своїми інтерактивними діалогами із NPC з використанням моделей штучного інтелекту.

Таблиця 1.2 – Опис застосунку «The Elder Scrolls: Blades»

Назва	**The Elder Scrolls: Blades**
Виробник:	Bethesda Game Studio
Архітектура	Мобільний застосунок для Android та iOS, консоль Nintendo Switch
Мова реалізації	Невідомо
Функції	1. Створення свого персонажа. 2. Створення ігрової локації. 3. Інтерактивна взаємодія з NPC. 4. Поступове підвищення складності в залежності від навичок гравця.
Переваги	1. Кросплатформеність. 2. Широкий спектр бойових механік. 3. Елегантний дизайн та локації. 4. Взаємодія з NPC з використанням AI. 5. Можливість гри у портретному та горизонтальному режимах. 6. Підтримка кількох ігрових режимів.
Недоліки	1. Деякі користувачі відмічають малу кількість сюжетів. 2. Присутні часті баги та проблема з продуктивністю застосунку. 3. Багато гравців скаржаться на не зовсім справедливий матч-мейкінг.
Вебсайт	https://elderscrolls.bethesda.net/en/blades

Гра також підтримується на консолях Nintendo Switch.

Рисунок 1.7 – Заставка The Elder Scrolls: Blades

Shadowverse

Застосунок є стратегічною картковою грою у 2D стилі, підтримується на багатьох ігрових платформах. AI застосується для кращого підбору гравців та аналітики.

Таблиця 1.3 – Опис застосунку «Shadowverse»

Назва	**Shadowverse**
Виробник:	Cygames
Архітектура	Мобільний застосунок для Android, iOS, Standalone платформи (Windows, Linux, macOS)
Мова реалізації	Невідомо
Функції	1. Підтримка багатокористувацької гри. 2. Підтримка великого обсягу карт та режимів. 3. Величезна авдиторія гравців по всьому світу. 4. Велика кількість контенту.

Кінець таблиці 1.3

Переваги	1. Безкоштовний. 2. Кросплатформеність. 3. Використовує AI для аналізу даних та при підборі гравців.
Недоліки	1. Деякі користувачі відмічають проблеми з анімаціями та ігровим процесом. 2. Відсутність активної взаємодії з AI. 3. Потрібен досвід гри у подібні ігри (по типу Hearthstone).
Вебсайт	https://www.cygames.co.jp/

Рисунок 1.8 – Логотип Shadowverse

Також усі ці ігри ніяким чином не використовують доповнену реальність разом із ШІ (в цілому вона й не всюди має сенс, адже кожна гра має суто свою ціль).

1.3 Аналіз застосунку, що розробляється

Створення ігрового застосунку з використанням алгоритмів штучного інтелекту у вигляді казуальної гри призначено для досягнення унікального ігрового досвіду та підвищення зацікавлення гравця. Підтримка застосунком режиму доповненої реальності ще більше урізноманітнить ігровий досвід.

Застосунок має підтримувати однокористувацький (клієнтський) режим роботи та бути сумісний із смартфонами, що використовують операційну систему Android (вище версії 6.0) та мають стабільний доступ до мережі інтернет (табл. 1.4).

Таблиця 1.4 – Опис застосунку, що розробляється

Функції	1) механіка навігації гравця; 2) механіка інвентарю гравця; 3) механіки знищення та респавну гравця; 4) механіки ігрових NPC з використанням алгоритмів ШІ та ML; 5) механіка оптимізації/комбінування зібраних ресурсів; 6) механіка покращення дрона; 7) система переходу між рівнями; 8) ігровий магазин; 9) прогресія гравця; 10) квестова система; 11) система діалогів та AI помічника (OpenAI плагін); 12) система збережень прогресу гравця;

Кінець таблиці 1.4

Користувачі	1) користувач-гравець
Сценарії роботи	1. Користувач встановлює застосунок та надає необхідні дозволи. 2. Користувач запускає гру та опиняється в ігровому меню. 3. Користувач натискає «почати гру», завантажується перший ігровий рівень, користувач «спавниться» на певній локації. 4. Користувач проходить ознайомлення із «лором гри» та керує персонажем (наземним дроном). 5. Користувач шукає прості ресурси (запчастини) , отримуючи підказками. 6. Користувач взаємодіє із запчастинами, коли знаходиться неподалік від них. 7. Користувач отримує завдання, виконує їх. 8. Після виконання завдання користувач рухається у вказане місце. 9. При досягненні вказаної локації користувач бачить іконку інвентарю із зібраними ресурсами та сховищем для переміщення ресурсів до головного місцепризначення. Ресурси та персонажа забирає транспортний дрон 10. Користувач закінчує ігровий рівень, повертаючись у хаб-ангар. 11. Користувач покращує свій дрон (збільшує розмір інвентарю, розширює «базу знань» про запчастини). 12. Користувач відкриває новий рівень.

1.4 Специфікація вимог до програмного забезпечення

Нижче наведено специфікацію вимог ПЗ для ігрового застосунку, що розробляється.

Призначення застосунку, для якого розробляється програмне забезпечення

Призначенням застосунку є демонстранція поєднання алгоритмів і методів штучного інтелекту при розробці ігрового застосунку.

Погодження, що ухвалені в програмній документації

Для створення ПЗ було погоджено використання таких технологій: платформа Unity у якості ігрового рушія, мова програмування C# у якості скриптової мови рушія, open-source плагіни для Unity, стороннє програмне забезпечення для генерації контенту для гри.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Дане ПЗ не має суттєвих обмежень у сфері споживання. Основна сфери призначення – розваги, також слугує прикладом використання алгоритмів штучного інтелекту в сфері ігор.

Характеристика користувачів (гравців)

Основні характеристики гравців: наявність смартфону з доступом до мережі інтернет.

Загальна структура і склад системи

Основна структура: ігровий застосунок, сторонні сервіси (API).

Загальні обмеження

Обмеження для роботи: стабільна інтернет-мережа

ОСНОВНІ ФУНКЦІЇ ІГРОВОГО ЗАСТОСУНКУ З ВИКОРИСТАННЯМ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ

Функція навчання NPC з використанням ML

Опис функції

Ця функція розкриває можливості бібліотеки Unity ML для ігрового застосунку. Надає інтерактивність та новизну взаємодії із NPC у ігровому світі.

Вхідна і вихідна інформація

Вхідна – дані про навколишнє середовище на ігровій мапі.

Вихідна – створення/оновлення «мізків» NPC.

Функціональні вимоги

Навчання агентів (NPC) через вбудовані алгоритми пакету Unity ML поведінки в залежності від вимог.

Функція AI-помічника

Опис функції

Функція надає інтерактивну складову ігровому застосунку, дозволяє гравцю взаємодіяти із AI-помічником через діалогове вікно, ввід та кнопки.

Вхідна і вихідна інформація

Вхідна – словесний ввід від користувача, або вибір певного сценарію/варіанту по кнопкам.

Вихідна – подія, яка дає ресурси, або провокує іншу подію/діалог.

Функціональні вимоги

Можливість взаємодії користувача з AI-помічником (OpenAI плагін) через діалогові вікна та введення/кнопки. Збереження результатів взаємодії в локальній історії діалогів, вплив на «базу знань» гравця, яка використовується при збиранні та розпізнаванні ресурсів.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

В даному ПЗ джерелом вхідної інформації є користувач (надання дозволів на пристрої, підключення нативних сервісів).

Нормативно-довідкова інформація

Вимоги відсутні.

Вимоги до способів організації, збереження та ведення інформації

Отримання даних через API сторонніх сервісів відбувається через HTTP-протокол. Інформація, що передається не зберігається. Застосунок може зберігати прогрес користача у локальному сховищі, що знаходиться на самому пристрої

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Вимоги до технічного забезпечення мають стандартні обмеження у вигляді підтримки смартфоном підключення до мережі Інтернет (мобільні дані або Wi-Fi).

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура включає питомі для ігрового застосунку частини, а саме: контроллер користувацького вводу, ігрові налаштування, механіки взаємодії з ігровим оточенням, механіку генерації ігрового світу, систему винагород, механіку інветарю тощо.

Системне програмне забезпечення

Застосунок має бути побудований з використанням платформи Unity в якості ігрового рушія. Для виконання функціональних вимог, застосунок повинен використовувати підтримувані версією Unity пакети (ML Agents), ассети (3D моделі розширення .FBX, картинки, спрайти) та плагіни (DOTween, Unity OpenAI). Усі вони будуть використовуватись при створенні ігрових рівнів, написанні алгоритмів взаємодії ігрових сутностей та не повинні конфліктувати один з одним.

Мова і технологія розробки ПЗ

ПЗ має розроблюватись за допомогою мови C# на платформі Unity. Додатково може використуватись мова Python (для Unity ML Agents).

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс має задовольняти вимоги UI/UX-дизайну для швидкого розуміння користувачем ігрового процесу застосунку. Ігровий інтерфейс має містити показчики основних ресурсів, предметів, здоров’я, кнопку виходу до пауз-меню. Адаптивні панелі налаштувань, меню та інших ігрових панелей.

Апаратний інтерфейс

Мобільний девайс користувача (смартфон) на базі ОС Android.

Програмний інтерфейс

У якості програмного інтерфейсу має використовуватись вбудована у платформу Unity UI Canvas компоненти для створення адаптивного користувацького інтерфейсу, система подій для обробки користувацького вводу. Для розробки ігрового застосунку використовується IDE Rider та Unity Editor, що забезпечує швидкість та зручність процесу розробки. Для пришвидшення розробки може використовуватись ШІ-контент, згенерований сторонніми сервісами.

Комунікаційний протокол

Застосунок передбачає використання протоколу HTTP – протоколу для спілкування між застосунком та сторонніми веб-сервісами (API).

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

ПЗ є доступним для користувачів, що грають у казуальні, пригодницькі ігри та бажають отримати ігровий досвід від нестандартної взаємодії із ігровим світом та персонажами.

Супроводженість

Застосунок не потребує супроводженості.

Переносимість

ПЗ повинно працювати на ОС Android. За потреби перенесення ПЗ на іншу операційну систему (Windows, iOS тощо) доведеться дещо змінювати користувацький інтерфейс та ігровий контроллер під конкретну платформу. Загалом Unity підтримує кросплатформність застосунку, що значно пришвидшує адаптації під задану ОС, проте, для кожної з ним є свої притаманні особливості, що потребують додаткового пропрацювання.

Продуктивність

Продуктивність ПЗ залежить від навантаженності ігрових сцен контентом (кількість ігрових персонажів, якість локації, об’єктів, що відмальовуються).

Надійність

Персональні дані користувача не зберігаються на сервері та не відправляються по мережі, однак ігровий прогрес (ресурси, поточний прогрес завдань, час ігрової сесії тощо) зберігається локально та у спеціальному сховищі Unity Cloud Save.

Безпека

ПЗ хоч і взаємодіє із сторонніми API через мережу інтернет, але не містить потенційних ризиків для безпеки.

Висновки до розділу 1

В першому розділі проведено аналітичний огляд алгоритмів та методів штучного інтелекту для розробки ігрових застосунків, проаналізовано застосунки-аналоги. Виокремлено переваги та недоліки аналізованих систем, проведено детальний аналіз застосунку, що розробляється, визначено функціональні вимоги, сценарії використання, складено специфікацію вимог (ТЗ) до ігрового застосунку.

2 МОДЕЛЮВАННЯ ІГРОВОГО ЗАСТОСУНКУ

2.1 Моделювання сценаріїв використання

Сценарій використання – перелік дій, відповідно до якого користувач взаємодіє із застосунком, виконуючи будь-яку дію, що дозволена функціоналом застосунку.

Даний ігровий застосунок розробляється у казувальному жанрі для мобільної платформи Android.

Основна задумка гри, яка проєктується – людство після численних катаклізмів, воєн та стрімкої революції штучного інтелекту поринуло у постапокаліптичну «пітьму». Люди вимушені жити у невеликих резерваціях та добувати ресурси для виживання. Для цього вони використовують уцілілих дронів та роботів ранніх моделей, які не отримували настільки розвиненого AI, щоб заподіяти шкоду людям та ефективні у протидії «зараженим» кібервірусами роботам і дронам.

Рисунок 2.1 – Згенерований інструментами AI концепт наземного дрона

Гра представляє собою керування наземним колісним дроном, за допомогою якого потрібно збирати запчастини та припаси на різних локаціях і доставляти їх до хабу (ангару), щоб підтримувати належний стан резервації.

Під час збору ресурсів, гравця можуть переслідувати NPC-роботи, керовані «зараженим» штучним інтелектом, програма якого – знищити усе, що взаємодіє з людьми.

У хабі гравці зможуть покращувати свій дрон, роблячи його ефективнішим і швидшим. Важливу роль у грі відіграватимуть алгоритми штучного інтелекту, які будуть підтримувати інтерактивність та динаміку взаємодії з ігровим світом. Також застосунок включатиме AI-помічника, що буде завжди доступний у хабі для надавання інформації про невідомі запчастини та деталі.

Короткий Use Case

Користувач встановлює гру на мобільний пристрій. Пристрій запускає застосунок та робить запит у користувача на доступ інтернет-з'єднання. Користувач підтверджує надання застосунку права доступу та переходить у головне меню гри, де розпочинає ігровий процес.

Поверхневий Use Case

Головний сценарій (успішний)

Користувач встановлює ігровий застосунок на мобільний телефон на системі Android. Пристрій запускає застосунок та робить запит у користувача на доступ до даних про камеру та інтернет-з'єднання.

Користувач підтверджує надання застосунку права доступу та переходить у головне меню гри. Меню вітає користувача та пропонує розпочати гру.

Головне меню являє собою хаб-ангар, де гравець бачить перед собою наземний дрон та велику кількість запчастин та всякого приладдя.

По натисненні кнопки «Почати експедицію» починається завантаження ігрової сцени, гравець бачить індикатор завантаження та опис локації. Після завантаження гравцю показується коротка кат-сцена із описом ситуації, чому і що має робити гравець.

Після закінчення, гравцю стає доступний джойстик керування, стає видимою панель із даними про здоров’я, ліміт ресурсів, батареї та хот-панель із завданнями. Гравець розпочинає виконання завдань (збирання запчастин та деталей на ігровій локації), під час збору показується взаємодія запчастин із наземним дроном та збільшення заповненого ліміту запчастин у інвентарі.

Успішно закінчивши перше завдання гравець отримує вказівку їхати до місця вивантаження зібраних запчастин. На шляху головному персонажу можуть зустрічатись ворожі роботи та дрони, які можуть заважити (забирати запчастини або намагатись знищити дрон гравця), користувачу треба об’їжджати перешкоди та уникати сутичок, адже він ще не має можливості захищатись від ворожих атак.

По прибуттю до точки гравець бачить зону вивантаження та «скидає» зібрані запчастини туди. Гравця вітають з успішної експедицією та за ним і ресурсами прилітає транспортний дрон, який евакуює їх разом до хабу. Гравцю показується панель закінчення рівня, к-ть зібраних запчастин та витраченого часу.

Повноцінна Use Case діаграма наведена у додатку А.

Альтернативні сценарії:

1) Гравець відмовляється надати доступ до даних. Застосунок повідомляє, що без доступу гра не зможе працювати без цих доступів та вимкнеться;

2) Після початку ігрової сесії гравець намагається зібрати запчастини, проте наштовхується на ворожих роботів, які його знищують. Гравцю показується підказка, що можна використати поточний баланс запчастин для посилання в «експедицію» нового наземного дрону. Гравець натискає на потрібну кнопку, ще раз програється коротка катсцена і гравець повторно отримує наземний дрон і продовжує виконувати задачу. Якщо дрон буде знову знищено, в цей раз гравцю буде виведено повідомлення про кінець «експедиції», яка, на жаль, була невдалою і поверне його у головний хаб.

Якщо ж гравець не втратись дрон, то далі сценарій буде збігатись із головним успішним.

3) Гравець після переходу до хабу після виконання/невиконання завдання першої «експедиції» бачить у головному меню підказку «AI асистент», що блимає. При натисненні гравцю показується його поточний стан ангару (загального інвентарю), кількість конкретних запчастин. За замовченням у гравця буде доступна 1 «невідома» запчастина зі «знаком питання». Гра запропонує використати «AI асистента» для того, щоб «класифікувати» запчастину та отримати випадкову нагороду.

Гравець настискає «Класифікувати», «AI асистент» генерує відповідь, «знак питання» зникає та з’являється детальний опис для запчастини, гравець такоє отримує у нагороду кілька невеликих простих запчастин за «класифікацію».

Далі гравцю показується підказка, що на ігрових локаціях завжди можна знайти цікаві та невідомі деталі, за класифікацію яких можна отримувати додаткову винагороду.

2.2 DFD та IDEF діаграми

DFD (Data Flow Diagram) – це графічне зображення, яке використовується для моделювання потоків даних у системі. DFD діаграми показують, як дані переміщуються між різними процесами в системі, з яких джерел надходять дані, як вони обробляються та куди передаються.

Основні елементи DFD:

1. **Процеси** (Processes) – дії або операції, які обробляють дані. Позначаються колом або овалом.

2. **Потоки даних** (Data Flows) – стрілки, які показують переміщення даних між елементами діаграми.

3. **Зовнішні сутності** (External Entities) – зовнішні джерела або споживачі даних. Це можуть бути користувачі, системи або організації. Позначаються прямокутниками.

4. **Сховища даних** (Data Stores) – місця, де зберігаються дані. Позначаються двома паралельними лініями або відкритим прямокутником.

DFD зазвичай використовують на різних рівнях деталізації:

- **рівень 0** – загальна картина системи, показує взаємодію між системою та зовнішніми сутностями.
- **рівень 1 і далі** – деталізованіші діаграми, які описують внутрішні процеси системи.

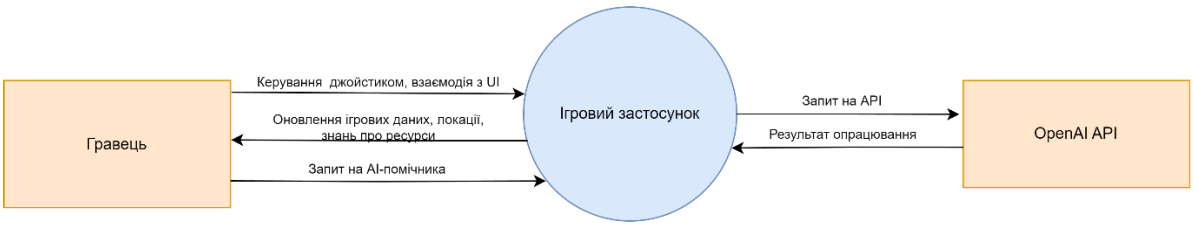


Рисунок 2.2 – DFD-діаграма 0 рівня (верхньорівнева)

DFD допомагає розуміти, як дані циркулюють у системі, і використовується для аналізу вимог та проєктування систем.

Для ігрового застосунку, що проєктується основними зовнішніми сутностями є гравці та сторонні сервіси (їх API), в даному випадку OpenAI API, доступ до якого буде налагоджено через безкоштовний плагін, який розглянуто в підрозділі з оглядом технологій.

Основні процеси застосунку будуть показані на DFD-діаграмі 1-го рівня нижче. Гравець виступає у ролі головного джерела та споживачі потоків даних, адже сам він взаємодіє з UI та провокує різного роду події та поведінку застосунку.

На діаграмі 1-го рівня показано більш деталізовану декомпозицію сервісів, які будуть створені для виконання основних взаємодій гравця із ігровим циклом. Сюди увійдуть:

– сервіс ігрових сцен;

– сервіс покращення характеристик дрону;
– сервіс взаємодії з AI-помічником;

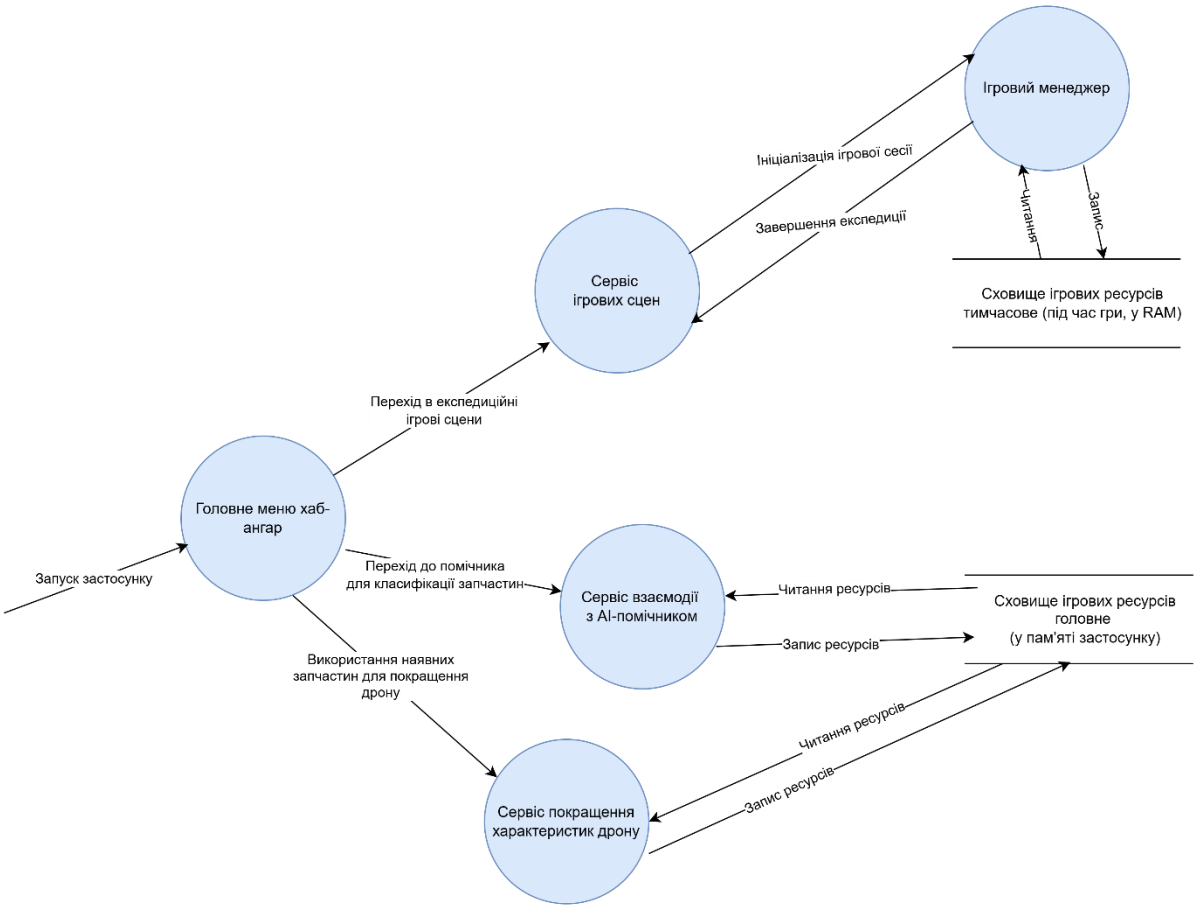


Рисунок 2.3 – DFD-діаграма 1-го рівня

IDEF0 (Integration Definition for Function Modeling) – це метод моделювання функцій, який використовується для графічного зображення та аналізу функцій, процесів або систем у бізнесі та інженерії. IDEF0 дозволяє візуалізувати систему як набір функцій, що обробляють певні вхідні дані та продукують вихідні результати на верхньому абстрактного рівні без декомпозиції.

Завдяки своїй структурованості та наочності, IDEF0 сприяє покращенню розуміння функціональних аспектів системи, що, в свою чергу, полегшує її розробку. На цій діаграмі (рис. 2.4) зображено функцію аналізу невідомої запчастини (яку гравець зібрав на ігровій локації) за допомогою AI-помічника, що доступний у головному меню хаб-ангарі.

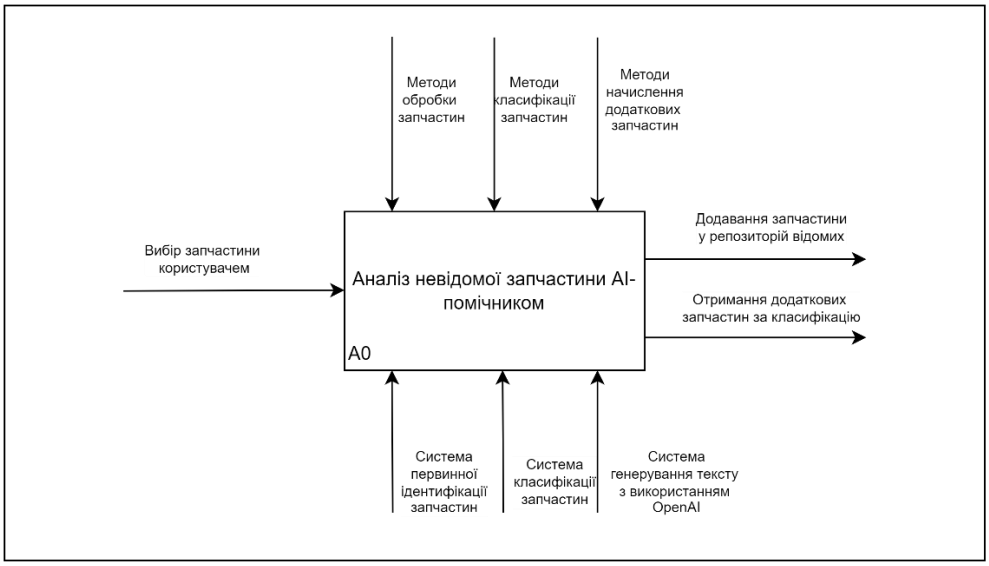


Рисунок 2.4 – IDEF0 діаграма для аналізу невідомої запчастини

Основні елементи IDEF0:

1. **Функції** (Functions) – це процеси або дії, які перетворюють вхідні дані на вихідні результати. На діаграмі функції зображаються прямокутниками.

2. **Входи** (Inputs) – дані або матеріали, які функція обробляє. Вони зображаються стрілками, що входять у лівий бік прямокутника.

3. **Виходи** (Outputs) – це результати роботи функції, які генеруються після обробки входів. Стрілки виходять з правого боку прямокутника.

4. **Механізми** (Mechanisms) – ресурси, які потрібні для виконання функції (наприклад, люди, обладнання, програмне забезпечення). Стрілки входять у нижню частину прямокутника.

5. **Контроль** (Control) – це правила або умови, які обмежують або керують виконанням функції (наприклад, політики, стандарти, регламенти). Стрілки входять у верхню частину прямокутника.

На рисунку чітко видно вхідні дані (вибір користувачем запчастини для аналізу), механізми виконання аналізу (системи ідентифікації та класифікації), контроллери (методи обробки та класифікації) і результат виконання функції.

IDEF дозволяє декомпозовувати функції на нижчих рівнях. Для цього створюються IDEF1, IDEF2 та інші діаграми, якщо в цьому є потреба.

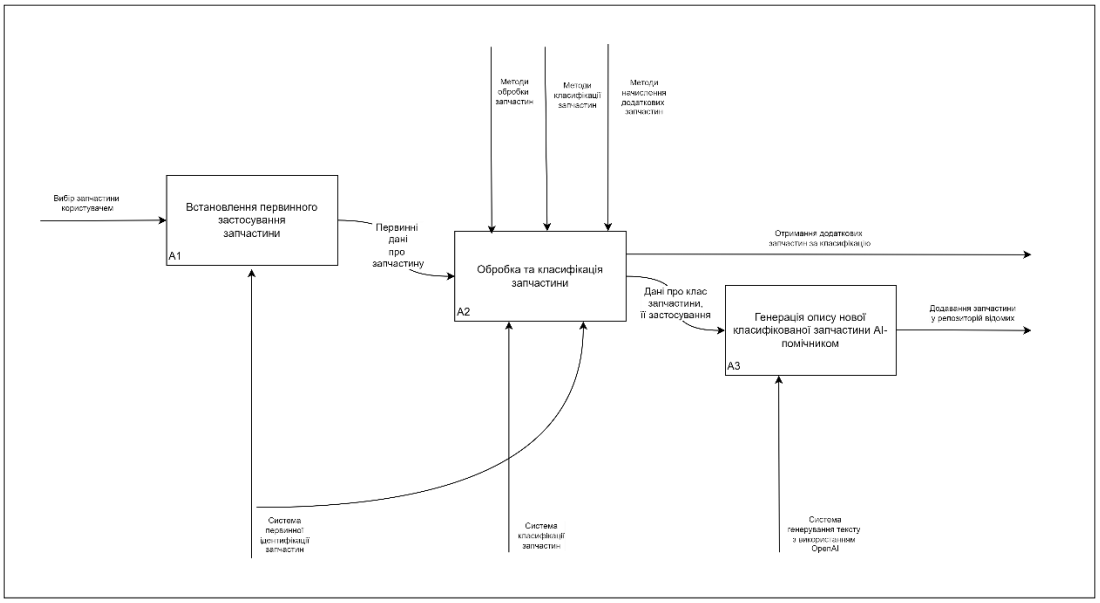


Рисунок 2.5 – IDEF1 діаграма з A1-A3 підблоками функції A0

На рисунку 2.5 показано 1 рівень декомпозиції функції аналізу невідомої запчастини, тут показано, які системи працюють в яких блоках та які методи застосуються для обробки невідомих запчастин.

2.3 Моделювання діаграм діяльності

Ігровий застосунок у вищеописаному стилі має немало сценаріїв діяльності, за якими можуть діяти гравці. Щоб відображувати основні дії, які може здійснити гравець зазвичай використовують діаграми діяльності, що часто є аналогом блок-схем.

Вони візуально описують логічну послідовність дій користувача та можливих розгалужень відповіді системи, що проєктується. Створено кілька діаграм діяльності, їх опис наведено нижче:

1) *Використання запчастин для покращення характеристик наземного дрону* (рис. 2.10). Дана діаграма відображає діяльність «Покращення дрону».

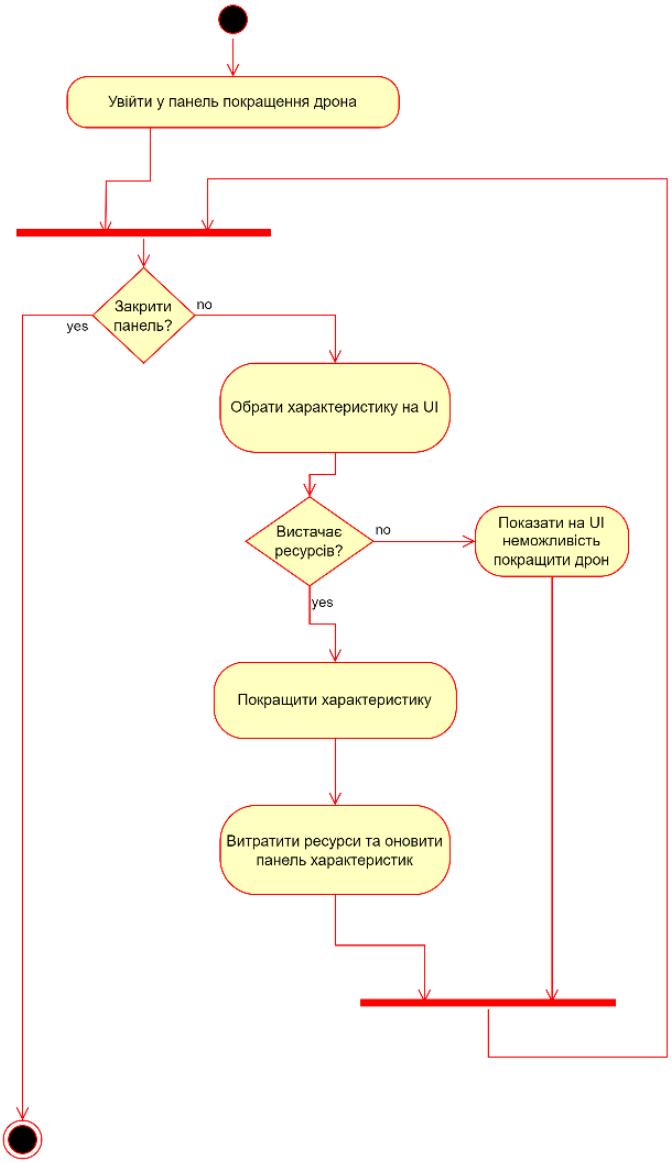


Рисунок 2.6 – Діаграма діяльності «Покращення дрону»

Коли гравець заходить у магазин йому відображаються можливі покращення характеристик дрону: кількість здоров’я, розмір батареї, максимальна ємкість вантажу, швидкість пересування.

Гравець може купувати ці покращення за зібрані запчастини, які можна отримати під час ігрової сесії (ескпедиції), AR-експедиції або отримуючи винагороди за завдання чи «класифікацію» у AI-помічника. Різні види запчастин впливають на різні покращення.

2) *Класифікація невідомої запчастини у AI-помічника.* Ця діаграма відображає діяльність, коли гравець хоче «класифікувати» невідому запчастину, яку він отримав під час «експедиції» (рис. 2.11).

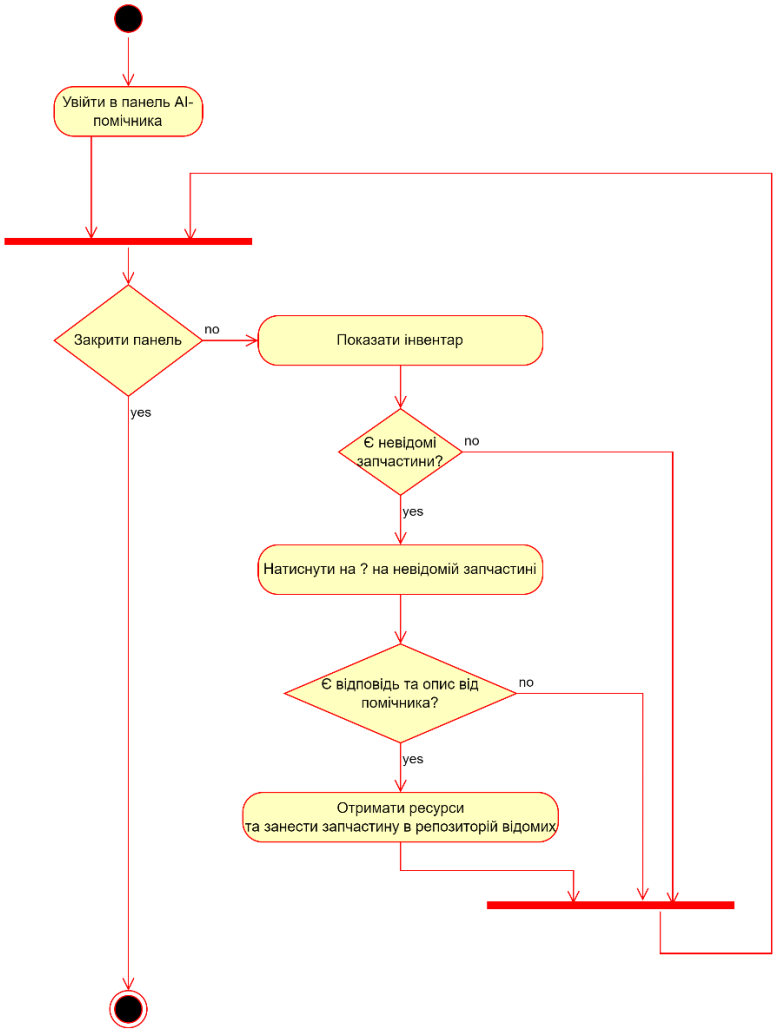


Рисунок 2.7 – Діаграма для діяльності «Класифікація невідомої запчастини»

Такий процес дозволяє гравцю отримати інформацію та додати цю запчастину до «репозиторію» відомих запчастин та отримати винагороду, якщо запчастина виявилась не звичайною.

3) *Відновлення ігрового прогресу після знищення дрона.* Діаграма показує діяльність гравця, коли той втратив дрон через ворожих роботів, пасивні пастки чи інші перешкоди. Гравець може витратити частину своїх зібраних запчастин для посилання в «експедицію» ще одного наземного дрона. Подібна дія доступна N-разів, що залежить від поточного стану розвитку

ангару (хабу), локації та наявних запчастин у гравця. Така діяльність англійською називається respawn action, яка часто використовується у іграх різних жанрів.

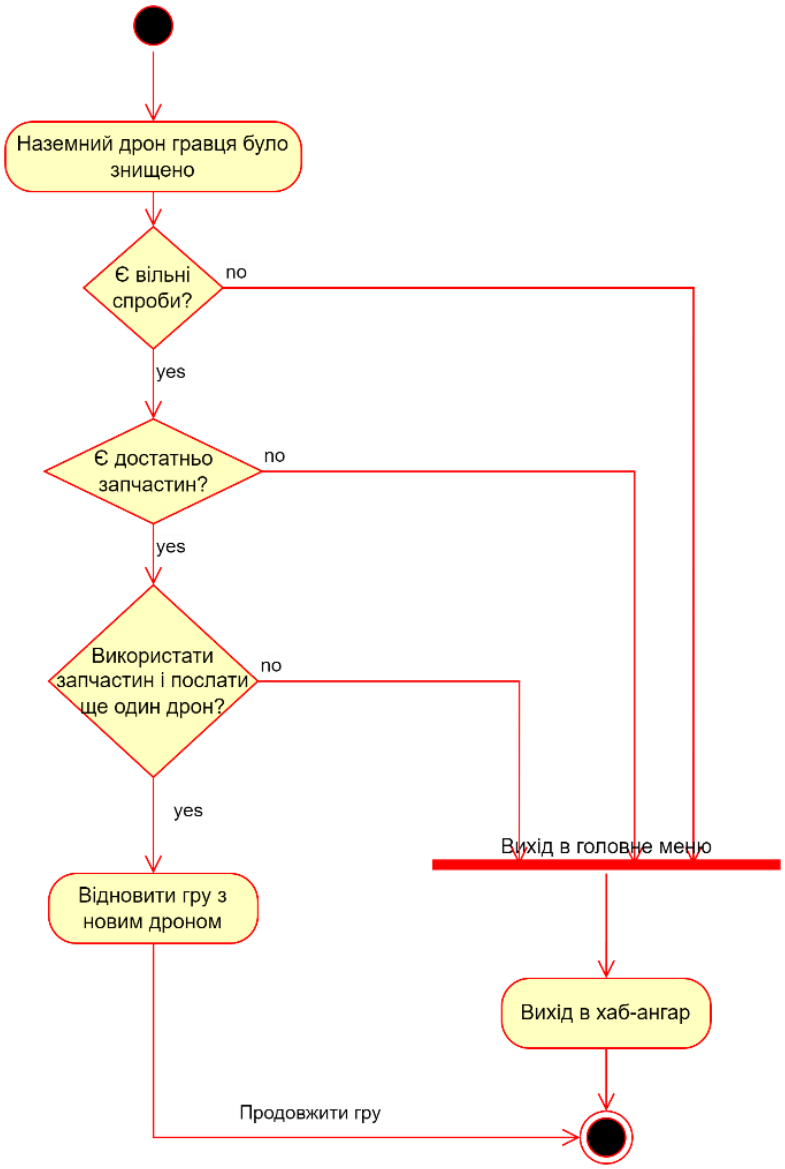


Рисунок 2.8 – Діаграма для діяльності «Відновлення гри після знищення дрона»

Діаграми були створені за допомогою спеціалізованого програмного вебзабезпечення моделювання та конструювання Draw.io для демонстрації деяких процесів дії користувача застосунку.

2.4 Огляд технологій

Для створення ігрового застосунку було обрано такий стек технологій:

– мова програмування C# (основна клієнтська мова платформи Unity);
– ігровий рушій (платформа) Unity;
– пакет Unity ML;
– пакет Unity для деформації мешу та розробки рівнів (ProBuilder);
– open-source плагін OpenAI Unity;
– інші open-source Unity плагіни для розробки ігор;

Рисунок 2.9 – Основний стек технологій

Рушій Unity безперечно є лідером у сфері розробки ігор. Unity Technologies постійно вдосконалюють свій продукт і пропонують передові рішення для повного циклу створення ігрових застосунків для різних платформ. Особливий попит Unity отримав саме в розробці мобільних ігор – понад 70% популярних ігор на мобільних платформах створені на його основі. Безкоштовний доступ, кросплатформність, детальна документація, багатий функціонал, інструменти аналітики та монетизації, а також велика спільнота розробників і готові рішення роблять Unity беззаперечним лідером серед інших рушіїв [7].

З огляду на ці переваги, вибір саме цього рушія був очевидним. Крім того, мова програмування C# є основною для створення скриптів у Unity, тому вона також була включена до обраного стеку технологій. Враховуючи тему кваліфікаційної роботи, пакети, які пропонує Unity також гарно вписуються у стек технологій.

Zenject (Extenject)

Фреймворк, розроблений для Untity з метою спрощення роботи із залежностями шляхом використання DI Container'а. Фактична імплементація патерну Dependency Injection.

Рисунок 2.10 – Лого Zenject

В Unity цей DI можна використовувати різними способами та методами. Підтримується:

- inject в методи;
- inject в конструктори;
- inject в у поля/властивості;

Контейнер представлений у вигляді Project Context'у на рівні проєкту та Scene Context`ів на рівні сцен, також є рівень ігрових об'єктів (рідко використовується). В свою чергу усі вони містять в собі «інсталлери».

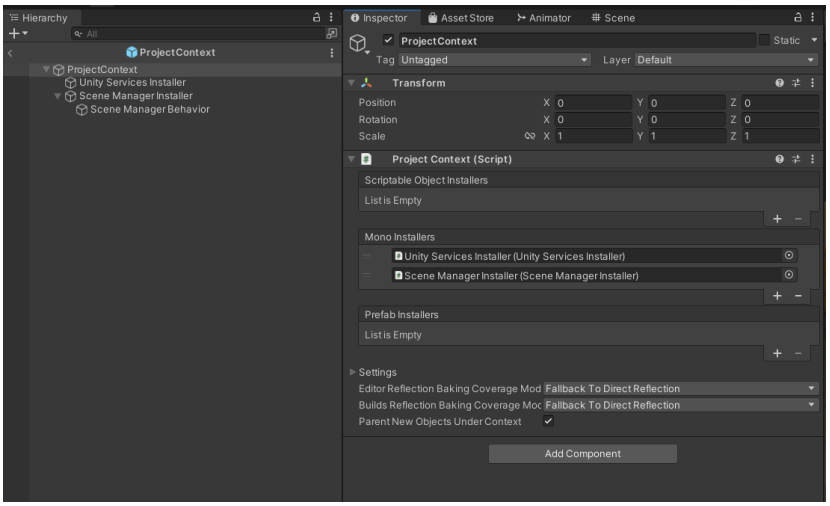


Рисунок 2.11 – Вигляд Project Context'у

На рисунку 2.11 вище показано яким чином інсталери додаються в контейнер.

Unity ML

Наступною технологією є підтримка Unity машинного навчання з використанням пакету ML Agents. Набір інструментів Unity Machine Learning Agents Toolkit (ML-Agents) — це проєкт із відкритим вихідним кодом, що дозволяє іграм служити середовищем для навчання агентів. Unity надає реалізацію (на основі PyTorch) найсучасніших алгоритмів, щоб дозволити розробникам ігор легко навчати ігрових агентів для 2D, 3D і VR/AR ігор. Розробники також можуть використовувати наданий простий у використанні API Python для навчання агентів за допомогою reinforcement learning, imitation learning, neuroevolution або будь-яких інших методів ML [23].

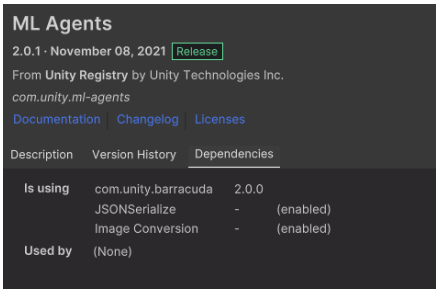


Рисунок 2.12 – Залежності пакету ML Agents

Unity ProBuilder

Цей пакет дозволяє будувати динамічні ігрові рівні, змінювати меш, об’єднувати чи роз’єднувати об’єкти, експортувати нові ассети, редагувати UV та ще багато інших корисних для будь-якого розробника речей.

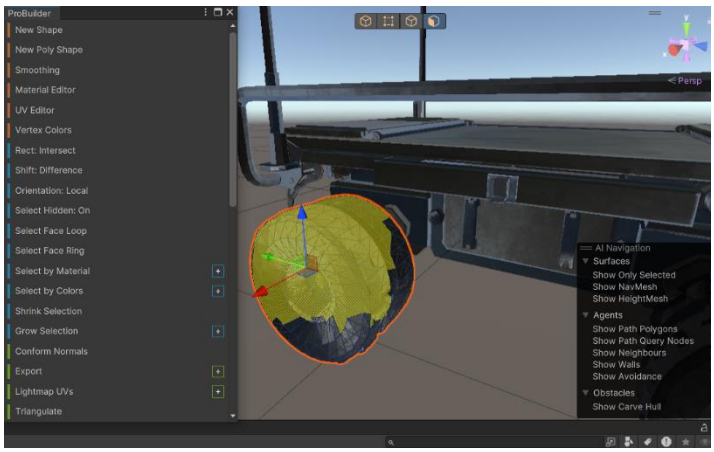


Рисунок 2.13 – Використання ProBuilder для виділення та відокремлення полігонів на меші

В застосунку, що розробляється він буде використаний для від’єднання деталей від «цільних» моделей 3D-мешів, з метою оптимізації та використання їх як інтерактивних об’єктів на ігровій мапі.

OpenAI Unity

Цей безкоштовний open-source плагін дає можливість напряму працювати із API OpenAI використовуючи при цьому Unity [25]. В даному плагіні присутні такі функції:

– взаємодія із ChatGPT – текстова взаємодія;
– Whisper – трасляція голосу в текст;
– DallE – генерація картинок за текстом;

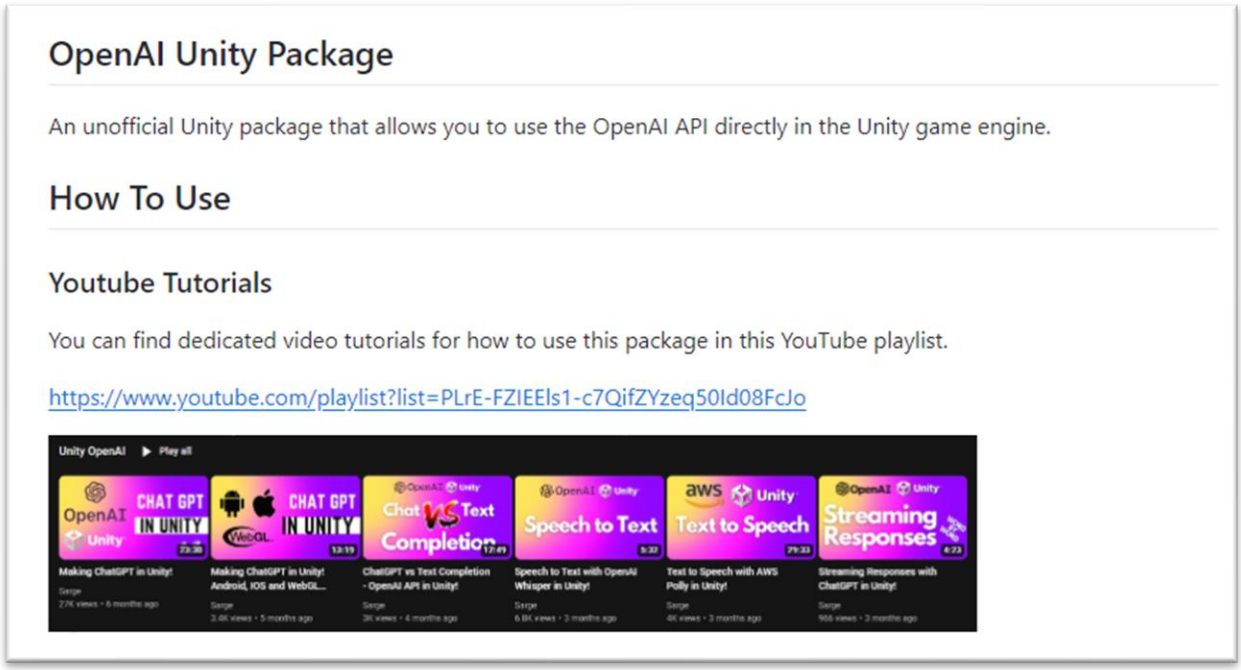


Рисунок 2.14 – Сторінка документації open-source плагіна

Cinemachine

Цей пакет дозволяє легко створювати катсцени, переходи, блендінги, зуми, не хвилюючись про імплементацію безлічі нюансів, які завжди виникають при роботі з камерою. Віртуальні камери, які пропонує Cinemachine значно спрощують життя програмісту, надають зручний інтерфейс для роботи, мають вбудовані можливості до розширення та автоматичне перемикання камер при деактивації попередніх.

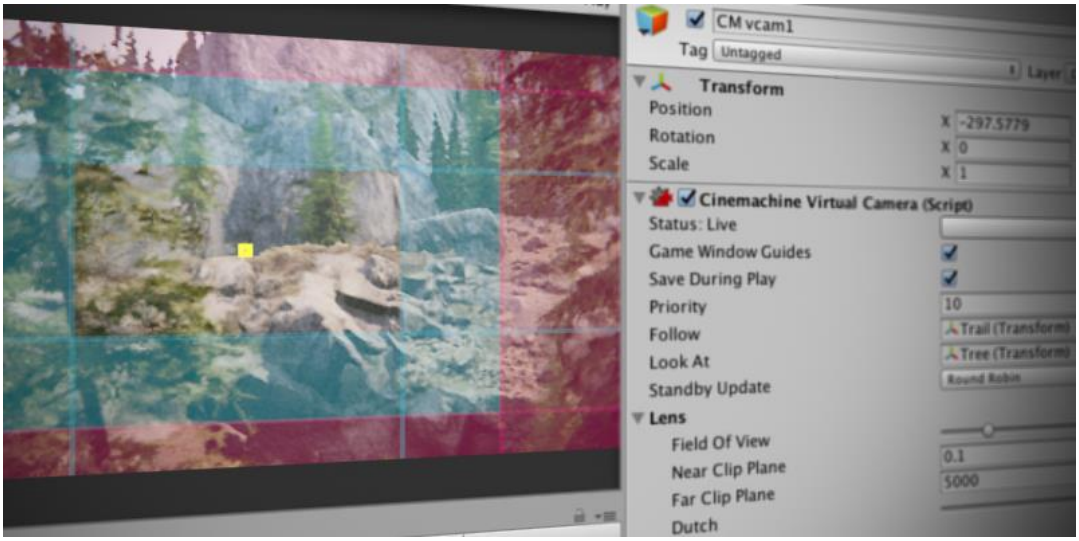


Рисунок 2.15 – Вигляд скрипта із віртуальною камерою Cinemachine в Unity

У проєкті віртуальні камери матимуть важливе значення саме на сцені головного меню – ангару, де камери допомогатимуть гравцю у навігації для потрібної йому панелі.

DOTween

Рисунок 2.16 – Лого DOTween

У розробці часто постає питання: «як плавно анімувати зміну позиції?». Першим на думку приходить DOTween – це безкоштовний опенсорс-пакет, який дозволяє анімувати безліч типів даних (в т.ч. вбудованих у Unity). Розмір мешу, позиція, поворот, колір, текст, затемнення та безліч іншого можна з легкістю імплементувати за допомогою цього інструмента.

Решта технологій, що використовуватимуться при розробці ігрового застосунку, та їх застосування будуть розглянуті у наступних розділах.

2.5 Пошук та інтеграція ресурсів для ігрового застосунку

Уміння підібрати моделі для ігрового застосунку високо цінується на ринку праці, особливо у компаніях, де немає змоги самостійно розробляти асети, моделювати меші або текстурувати. Сучасні виклики вимагають від розробників ігор сучасних рішень, одне з яких це купівля моделей, але, дякуючи дизайнерам, моделлерам та розробникам, які викладають власні портфоліо або напрацювання у відкритий доступ, студенти мають змогу розвиватись та розробляти свої проєкти з нуля без необхідності витрачати власні ресурси.

Для ігрового застосунку, що розробляється, а саме створення ігрового рівня та юнітів, варто враховувати сеттинг, що набагато складніше, коли безкоштовних доступних може і не знайтись. На щастя, Unity Asset Store допомогає у пошуках безкоштовних асетів за категоріями фільтрів [28].

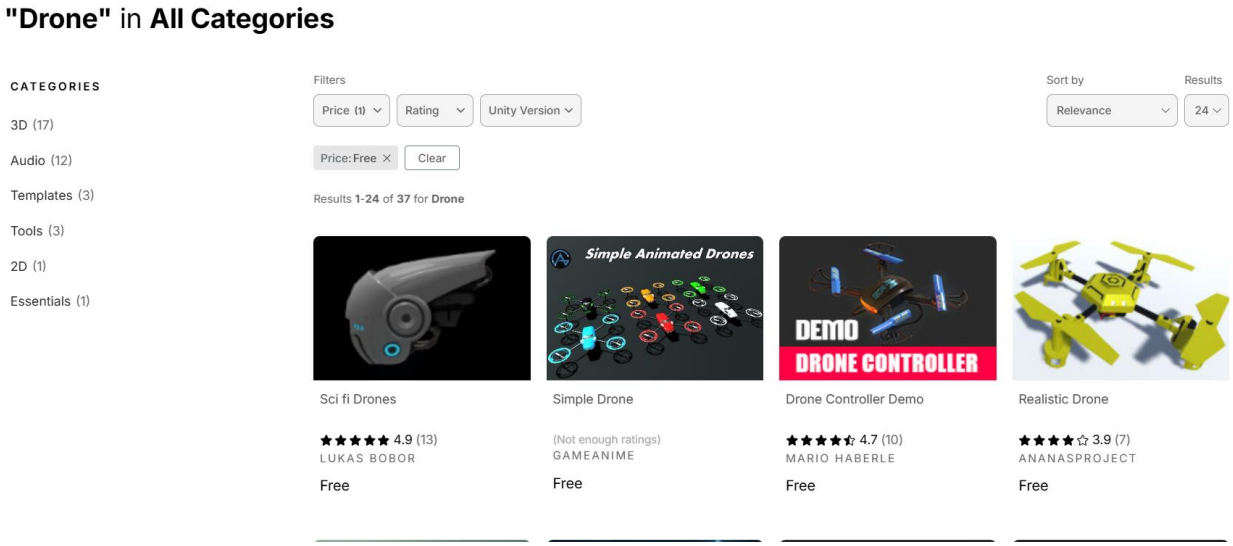


Рисунок 2.17 – Сторінка Unity Asset Store

Проте, не лише Unity Asset Store може стати в нагоді, чудові, а головне безкоштовні, асети (моделі, анімації, звуки, готові програмні рішення) можна знайти на таких сервісах як:

- Sketchfab (меші);
- Free3D (меші);
- itch.io (усе для геймдеву);

– Turbosquid (меші);
– Mixamo (анімації);
– CGTrader (меші) та інші.

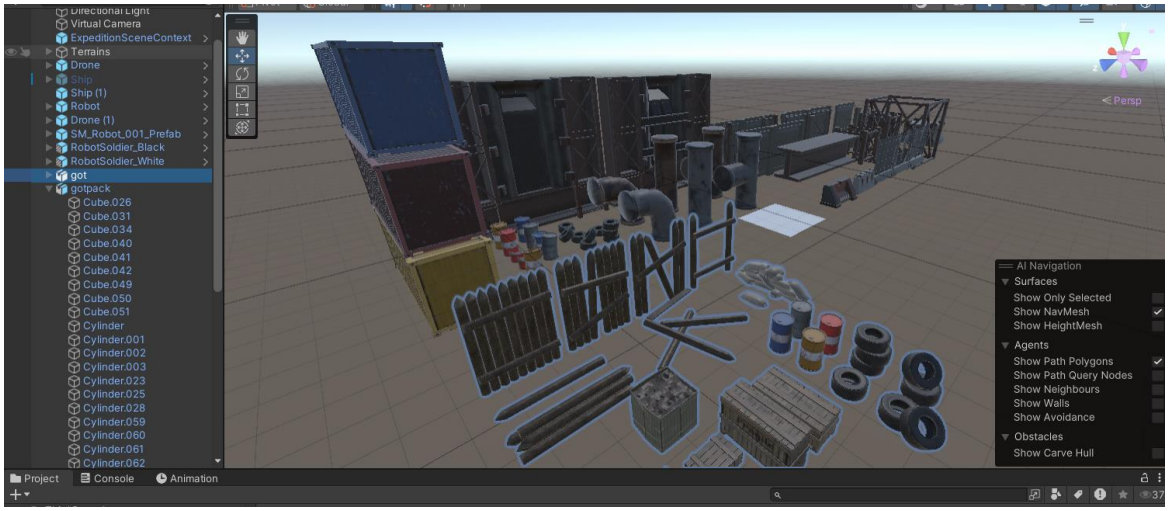


Рисунок 2.18 – Імпортовані об’єкти екстер’єру

Такі ресурси, на відміну від Unity пакетів, необхідно самостійно інтегровувати/імпортовувати в Unity, налаштовувати анімації, моделі та матеріали.

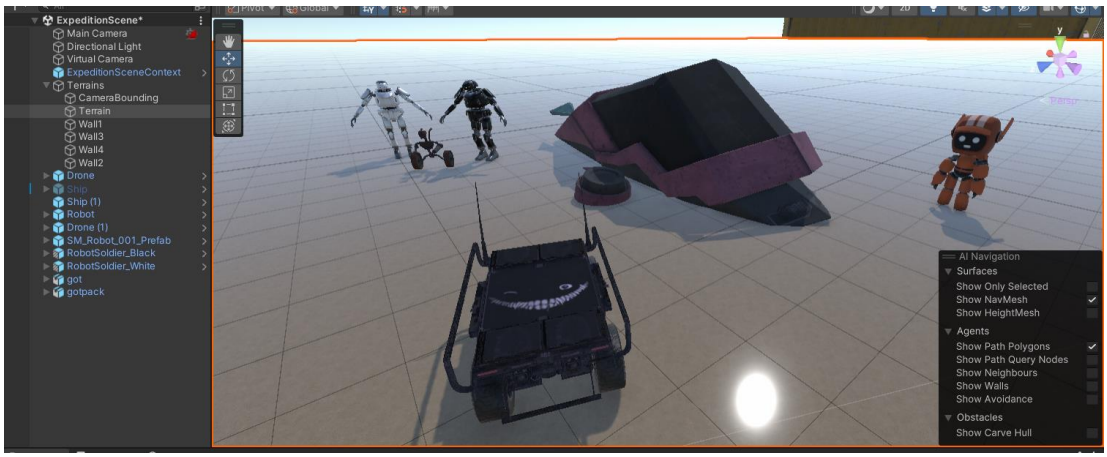


Рисунок 2.19 – Моделі юнітів, які будуть використовуватись у грі

З допомогою вищезазначених джерел було знайдено декілька якісних безкоштовних моделей наземного траспортного дрона, корабля для транспортування, роботів, які будуть виконувати роль ворогів, асети для гаража (головного меню), ресурсів (які можна буде збирати) та базові анімації роботам.

Висновки до розділу 2

В другому розділі було описано основні етапи, сценарії використання та алгоритми роботи ігрового застосунку. Розроблено DFD, IDEF та UML-діаграми, а саме DFD0, DFD1, IDEF0, IDEF1, діаграму використання, діаграми діяльності, завдяки чому закріплено навички описувати алгоритми та архітектуру застосунку за допомогою UML. Описано пошук ресурсів для розробки ігрових рівнів та юнітів, показано знайдені ресурси, що будуть застосовані при проєктуванні ігрових сцен та розробці ігрової логіки.

Обґрунтовано використання стеку технологій, а саме ігрового рушія Unity, мови програмування C#, модуля Unity ML та open-source плагінів для Unity.

3 ПРОЄКТУВАННЯ ІГРОВОГО ЗАСТОСУНКУ

3.1 Проєктування класів

UML (Unified Modeling Language) – це мова моделювання, що використовується для візуалізації, проєктування та документування програмних рішень. Для проєктування мобільних ігрових застосунків також використовують UML-діаграми, більшу частину з них вже було описано і використано у попередньому розділі у якості процесу моделювання застосунку, проте, найважливішою та найзрозумілішою залишається діаграма класів.

Діаграми класів (англ. Class Diagrams) відображають структуру системи шляхом описування класів, їх взаємозв'язків, атрибутів та методів.

Для застосунку, що проєктується важливо виокремити класи пов’язані із поведінкою головного «гравця» – наземного дрона, класів, що відповідають за поведінку ворожих роботів, класів для ресурсів та ряд інших, які вказані в таблиці 3.1, разом із основними атрибутами та методами і на, власне, діаграмі класів.

Таблиця 3.1 – Класи

Клас	**Параметри**	**Методи**	**Призначення класу**
UnityServicesInstaller	_unityServicesManager	InstallBindings()	Клас, що «інсталює» Unity ServicesManager, додаючи його у DI Container
UnityServicesManager		Initialize()	Клас, який ініціалізує сервіси Unity та дозволяє працювати із ними ззовні
		ApplyRemoteSettings()	

Продовження таблиці 3.1

SceneManagerInstaller	_sceneSettings	InstallBindings()	Клас, що інсталює і додає SceneManager Behaviour у DI Container
	_sceneManager Behavior		
SceneManagerBehavior	scenesRepository	StartLoadScene()	Клас, що відповідає за завантаження і вивантаження сцен
		UnloadScene()	
		LoadAsyncScene()	
SceneLoaderBehavior	_sceneManager Behavior	Awake()	Клас, використовує менеджер для завантаження потрібної сцени при натисненні кнопки
	sceneType	OnDestroy()	
	loadButton		
HangarCamera ControllerInstaller	_hangarCamera Controller	InstallBindings()	Клас, що інсталяє залежності для камер у ангарі, та додає контроллер у DI Container
	_panelBehaviors		
HangarCamera Controller	CurrentContainer	Initialize()	Клас, що відповідає за зміну камери та виду в ангарі
	panelContainers Repository	ChangeCamera View()	
	PanelChangedAction	OnDestroy()	

Кінець таблиці 3.1

DroneController	rigidbody	Awake()	Клас, який відповідає за керування наземним дроном, включає в себе методи пов’язані із фізикою руху
	leftWheel	Update()	
	rightWheel	FixedUpdate()	
	wheels	Steer()	
	maxSteeringAngle	Accelerate()	
	steeringSpeed		
	maxFrontMotorForce	Brake()	
	maxBackMotorForce	StartMove()	
	power	GetMaxSpeed()	

До таблиці були внесені класи, які взаємодіють із головним меню та дроном. Вони також включають у себе «біндинги» до DI контейнера, який надається вже описаним фреймворком Zenject. Такий DI контейнер включає в себе усі потрібні залежності та дозволяє легко і просто їх «резолвити» за потреби без необхідності мануального додавання залежностей через конструктори, властивості чи окремі ініціалізаційні методи. Також вони дозволяють спростити етап ініціалізації, викликаючи всю необхідну ініціалізаційну логіку зарання у InstallBindings() методах вбудованих класів-наслідників MonoInstaller’ах. Варто зазначити, що певні інсталлери повинні «існувати» лише у рамках життєвого циклу сцени Unity, тому їх потрібно додавати у SceneContext’и, а деякі можуть «жити» протягом усього життєвого циклу застосунку – такі мають додаватись у ProjectContext’і.

У таблиці 3.1 показані не усі класи, які будуть використовуватись при кодуванні у 4 розділі. Дані класи можна назвати найбільш простими у поясненні та легкими при проєктуванні. Решта класів вимагатиме додаткового опису власної логіки у наступному розділі. Ці та інші класи візуально наведені у додатку Б, на рисунках Б.1, Б.2 та Б.3.

3.2 Проєктування інтерфейсу та арт-стилю застосунку

Користувацький інтерфейс – найважливіша частина застосунку. Взаємодія із інтерфейсом, користувацький досвід мають бути простими та очевидними. Важливу роль грає інтерактивність ігрового інтерфейсу, саме тому Unity надає ряд стандартних компонентів інтерфейсу для швидкого прототипування та налаштування.

Для будь-якого ігрового застосунку і для швидкості прототипування, запроєктовані інтерфейси можна імплементувати вбудованими засобами Unity (Canvas UI). Проте, перед цим необхідно визначитись – який арт-стиль потрібен для застосунку? Беручи до уваги «сеттинг» застосунку, що розробляється, стиль має включати елементи «пост-апокаліпсису», роботизованих платформ та реалістичного спустошеного середовища.

У якості контенту (фонів та деяких іконок) було використано можливості безкоштовного сервісу для генерації зображень Leonardo AI [22]. Це зручний та потужний інструмент, який дозволяє безкоштовно та швидко згенерувати зображення по заданому «промту» – інструкції для нейромережі, яка генерує результуючий контент.

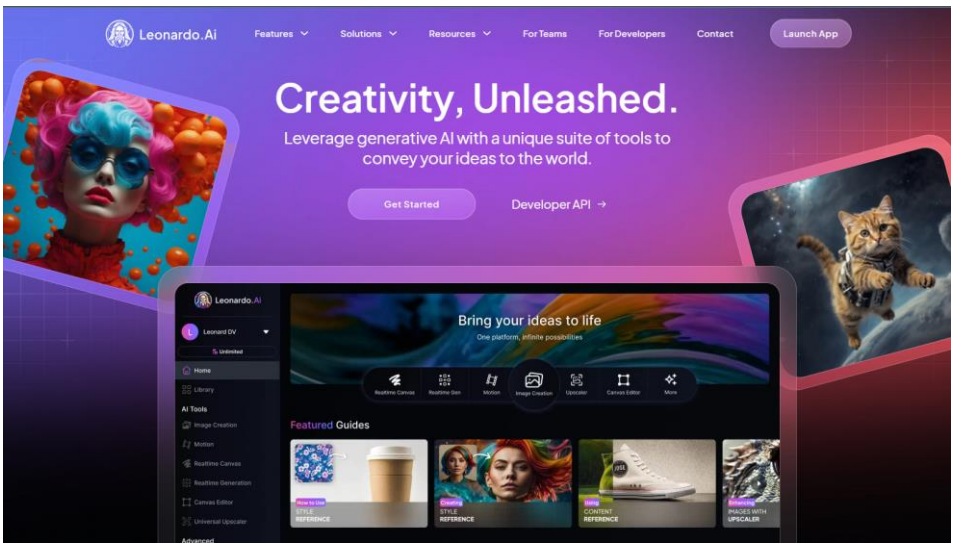


Рисунок 3.1 – Головна сторінка Leonardo AI сервісу

Для отримання кількох варіантів вікон завантаження ігрового застосунку було написано такий промт:

«Create a loading screen for a game in a post-apocalyptic style. The main setting includes a four-wheeled UAV drone with a flat, rectangular platform. The drone is constructed from metal and features a weathered, practical design with visible wear and tear. The back setting depicts devastated deserted land. There is a Logo on the top of the screen — "Rusty Boq."».

На рисунку 3.2 показано результати генерації зображень.

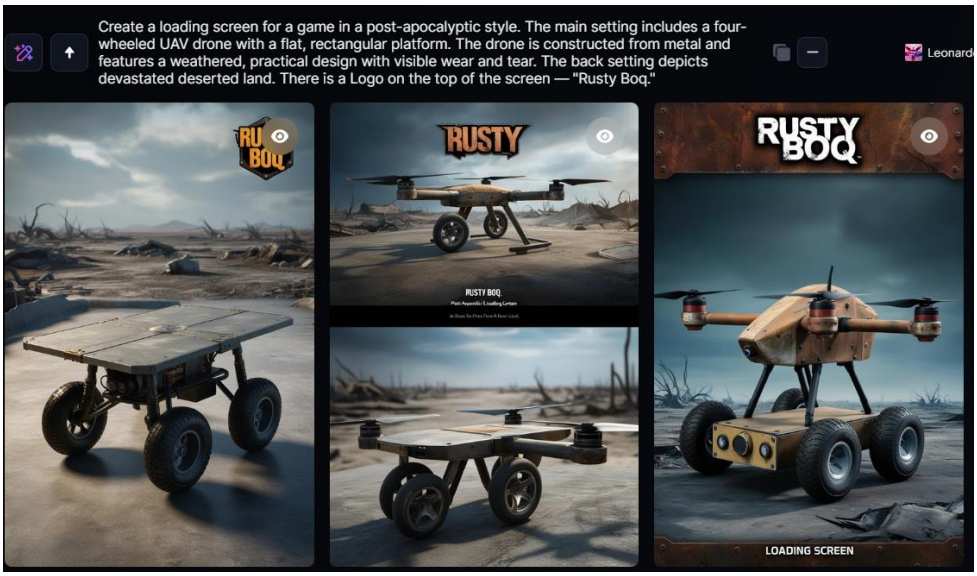


Рисунок 3.2 – Результат генерації зображень за допомогою Leonardo AI

На рисунку вище видно, що, на жаль, не всі результати влаштовують та підходять під головний задум прототипу.

Оскільки нейромережа не може генерувати одразу «чудовий» результат і вона постійно потребує «чіткого і зрозумілого промту», то довелось проітерувати кілька варіантів зображень фонів та використати найкращі з них.

Через ряд помилок, проб та виправлень «промтів» вдалось отримати кілька гарних зображень, які і були використані для сцени завантаження та переходів між сценами. Було обрано самі ті зображення, де чітко зрозуміло, що наш головний «персонаж» – наземний дрон, а не «квадрокоптер» чи якийсь незрозумілий «фентезійний» гібрид.

Одне і зображень було оброблено у графічному редакторі Photoshop, щоб виокремити логотип назви «Rusty Boq» як стандартної іконки (іконка на робочого столі телефона).

Отриманий результат зображено на рисунку 3.3 нижче, він був доданий в налаштуваннях ігрового проєкту та став «лого» самого проєкту.

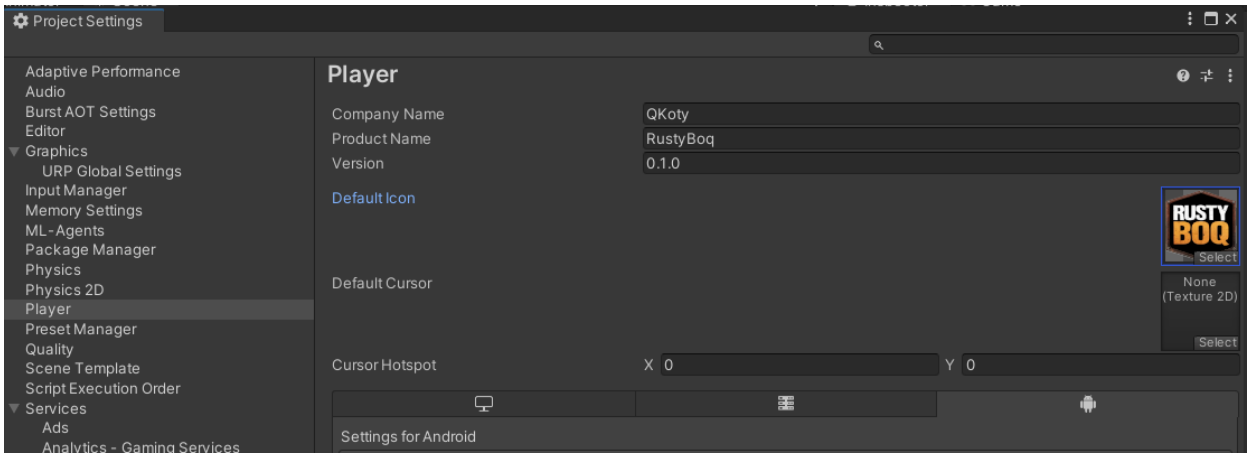
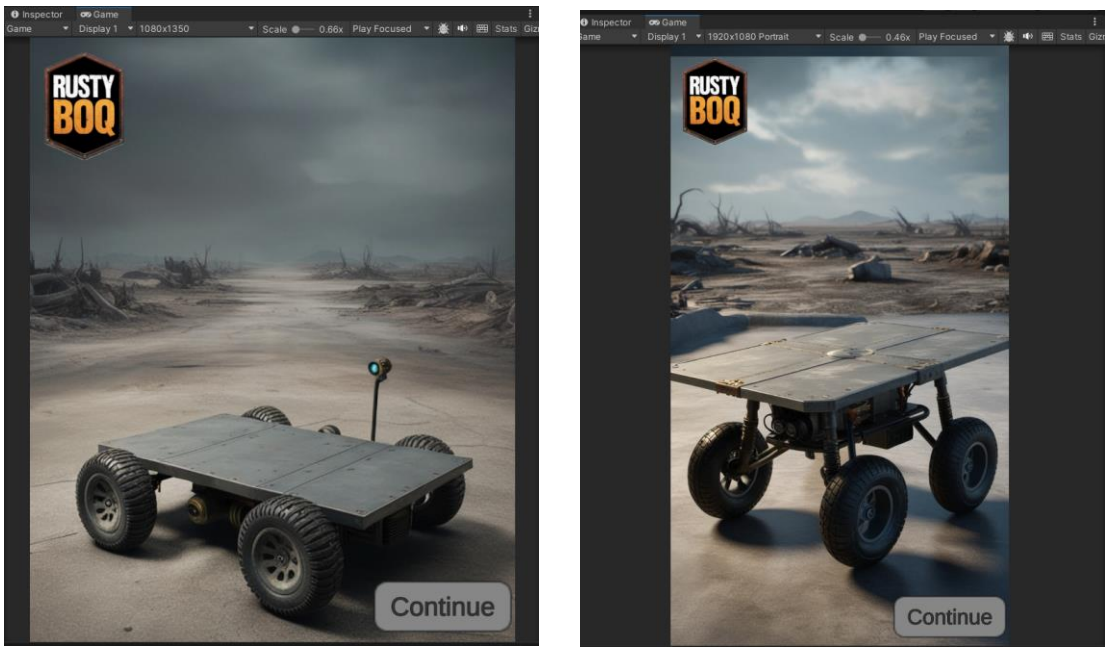


Рисунок 3.3 – Налаштування Unity проєкту із обраною іконкою

Опрацювавши решту зображень та вибравши найкращі – їх було оформлено та збережено фоновими для початкої сцени завантаженн.



а) б)

Рисунок 3.4 – а, б – варіанти вікна завантаження (початкової сцени)

Решта програмного інтерфейсу (у меню) наведена у додатку В (рис. В.1 та В.2).

3.3 Створення музичного супроводу (тематичний саундтрек)

Часто мобільні ігри не мають або не використовують можливість музичного (аудіо) супроводу, хоча це є невід'ємною складовою для покращення ігрового досвіду та атмосфери ігрового застосунку. Для цього було вирішено створити кілька музичних саундтреків (аудіо) за допомогою безкоштовного онлайн-сервісу Suno AI. Це сервіс, який використовує алгоритми ШІ (на базі Chat-GPT моделей) для генерації різноманітних аудіо із обраними жанрами, стилями музики. Підтримує також генерацію і озвучення слів.

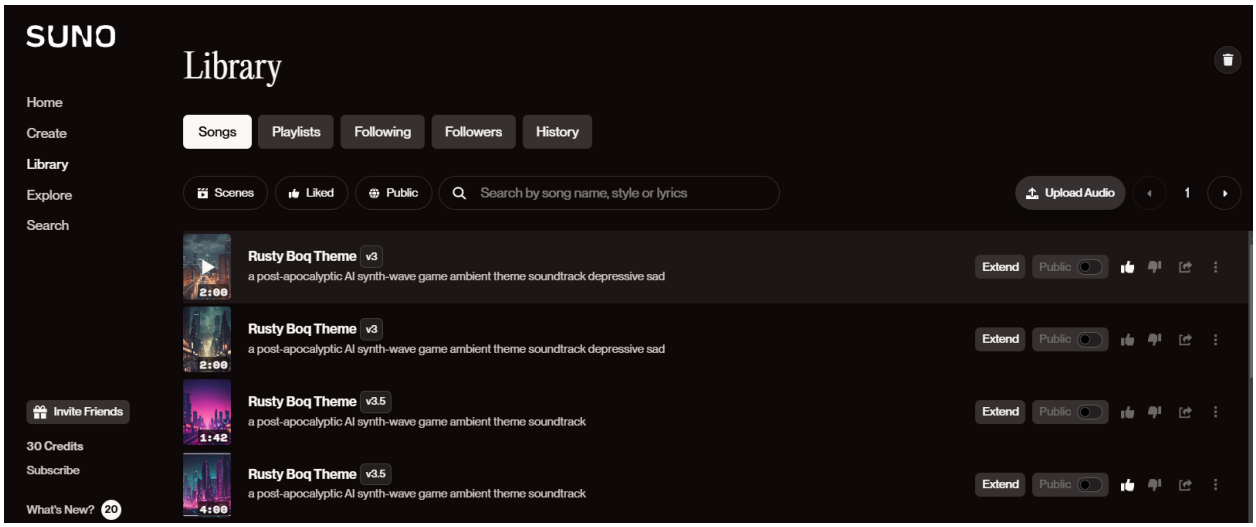


Рисунок 3.5 – Зовнішній вигляд онлайн-сервісу

Для створення безкоштовного акаунта нам потрібно зареєструватись та підв'язати пошту на цьому сервісі. Далі потрібно обрати жанр, стиль та версію GPT, яка використовуватиметься для генерації саунтреку. Було задано такий промт:

«*A post-apocalyptic AI synth-wave game ambient theme soundtrack*».

В результаті кількох проб отримано 4 варіанти саунтреків у форматі .mp3, з яких обрано два (версії v3.5 та v3), які й імпортовано до Unity та використано як ambient аудіотеми гри.

3.4 Створення та організація Unity проєкту

Для розробки ігрового застосунку спочатку необхідно встановити версію рушія Unity (було обрано 2022.3.48f1), використовуючи Unity Hub, та створити проєкт з репозиторієм, підв’язавши його до системи контролю версій GitHub.

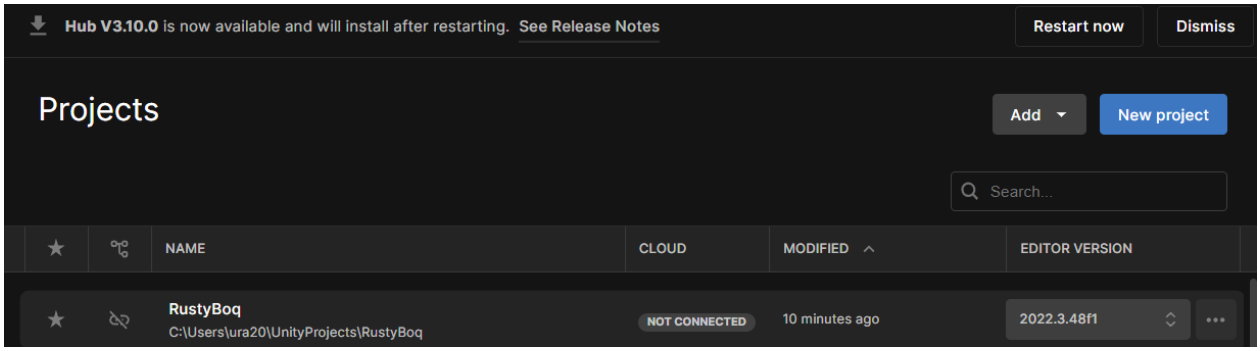


Рисунок 3.6 – Створений проєкт у Unity Hub

Оскільки проєкт мобільний, необіхідно встановити додаткові залежності, а саме Android Build Support, його можна додати у налаштуваннях версії Unity у вкладці Modules.

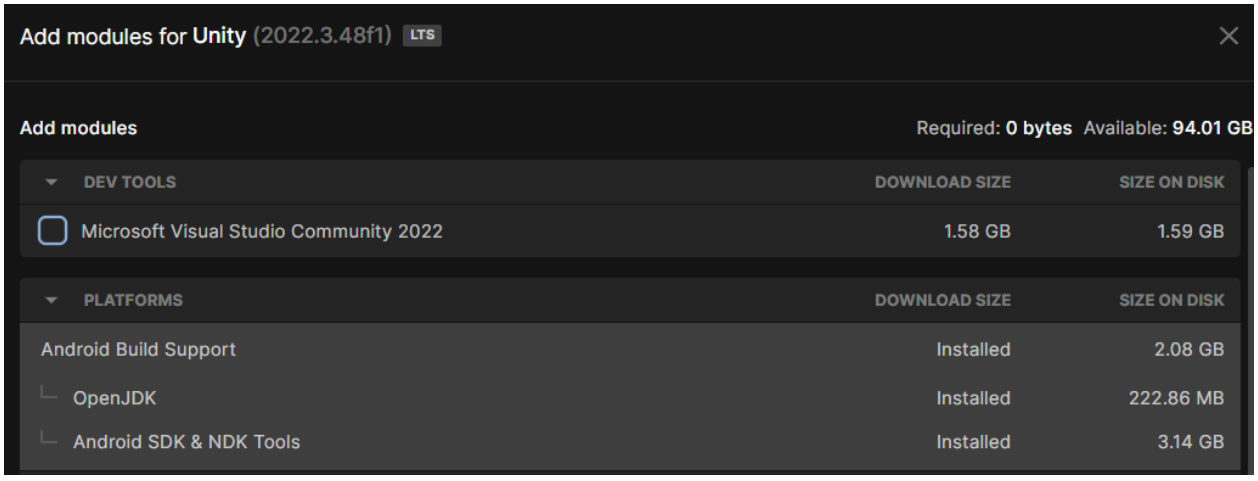


Рисунок 3.7 – Встановлений Android модуль для проєкту

Дотримуючись стандартів версіонування було створено репозиторій для проєкту та додано його на власний акаунт у GitHub.

Проєкт створено, а отже треба розподілити його на папки, щоб організація файлів була зручною, доступною та зрозумілою.

Рисунок 3.8 – Стуктура папок в проєкті

Нижче наведено опис папок:

1) 2D – папка двовимірних ресурсів, таких як вікна (скріни);

2) Prefabs – папка для ігрових об’єктів (префабів), які готові до використання на ігровій сцені;

3) Resources – папка для збереження важливих ресурсів (файлів, текстур, спрайтів, моделей тощо);

4) Scenes – папка для збереження сцен;

5) Scripts – папка зі скриптами (файли з програмним кодом, які містять ігрову логіку);

6) Settings – папка із налаштуваннями проєкту;

7) Third Party Assets – папка для ресурсів або сторонніх бібліотек, які можуть використовуватись для розробки.

8) Packages – папка із залежностями UPM.

Для інтеграції пакетів у Unity використовується Unity Package Manager (UPM), який має кілька опцій інтеграції:

– додавання пакету з диску;
– додавання пакету в архіві типу .tar.gz;
– додавання залежності по git’у;
– додавання пакету по імені.

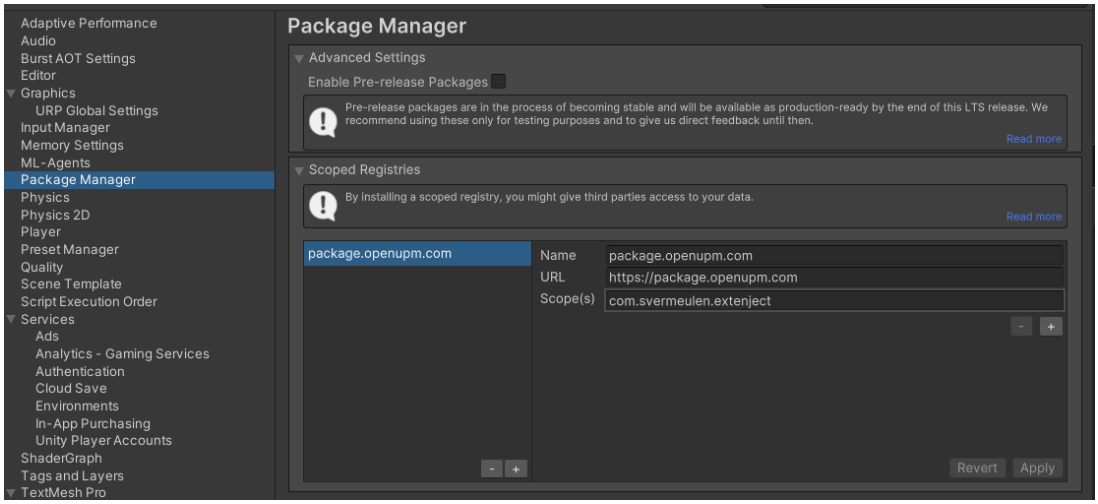


Рисунок 3.9 – Посилання на openupm.com для Extenject (Zenject) пакета

Для додавання пакетів, які «хостяться» на інших сервісах або пакедж менеджерах, або ж мають специфічний спосіб розгортання, в Unity необхідно додати scoped registry та вказати необхідне посилання як на рис. 3.9.

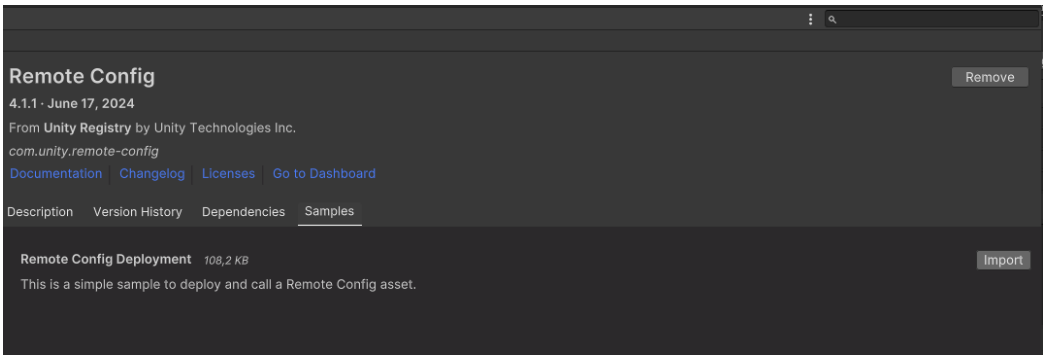


Рисунок 3.10 – Пакет Remote Config

Деякі пакети офіційно оновлюються Unity. Вони знаходяться в registry Unity, та, частіше за все, містять семпли для демонстрації роботи із функціоналом пакетів (рис. 3.10). Семпли не обов’язкові для встановлення.

3.5 Проєктування сцен

Кожна гра оперує поняттям «ігрової сцени», де гравець взаємодіє із ігровим світом, пересувається, отримує ресурси, щось купує, знищує противників, заробляє очки тощо. Порядок сцен та їх проєктування – важлива складова при розробці мобільного ігрового застосунку. Було прийнято рішення розділити ігровий цикл на три основні сцени:

1. Bootstrap scene (сцена завантаження).
2. Hangar scene (головне меню – ангар у даному випадку).
3. Expedition scene (ігрова сцена, де буде виконуватись головна логіка гри).

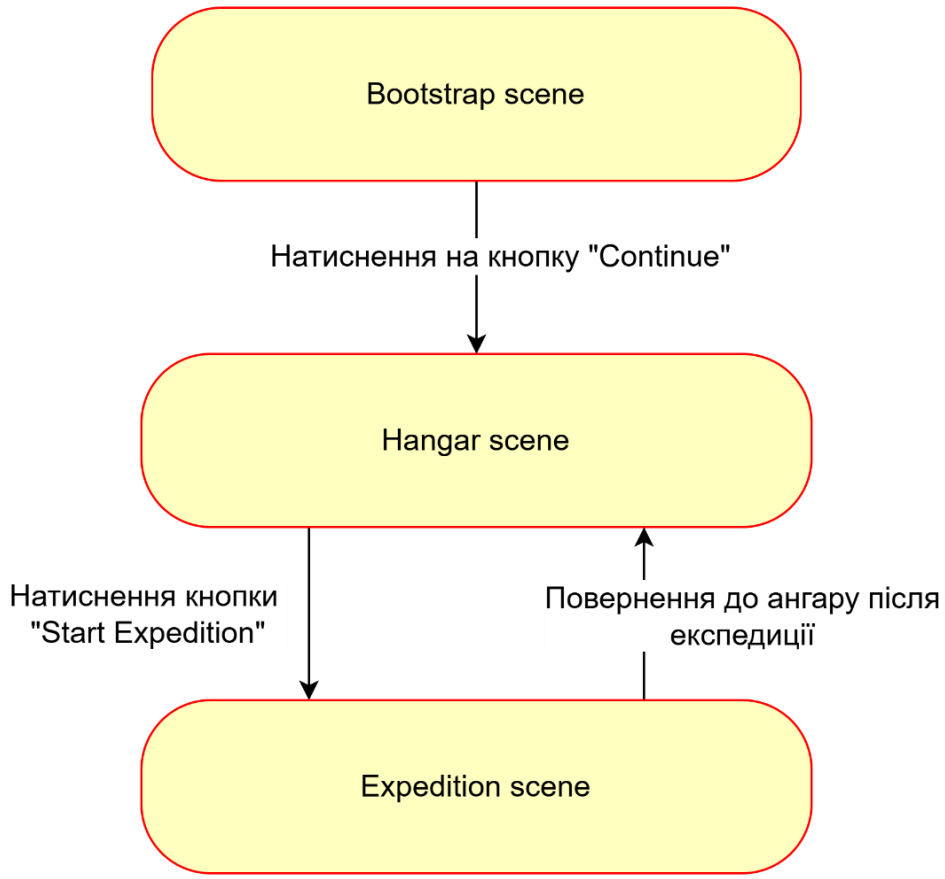


Рисунок 3.11 – Схема ігрового циклу сцен

Для переключення сцен в останньому розділі буде описано код зміни сцен. На рисунку 3.11 зображено верхньорівнева схема ігрового циклу сцен.

3.5.1 Проєктування сцени головного меню

Головне меню будь-якого ігрового застосунку є одним із ключових місць, де гравець проводить свій ігровий час. Зацікавити гравця, дати йому інтерактивну складову у меню, а саме: взаємодію із об’єктами, камерою, приміщеннями – найкращий спосіб досягтити гарного користувацького досвіду.

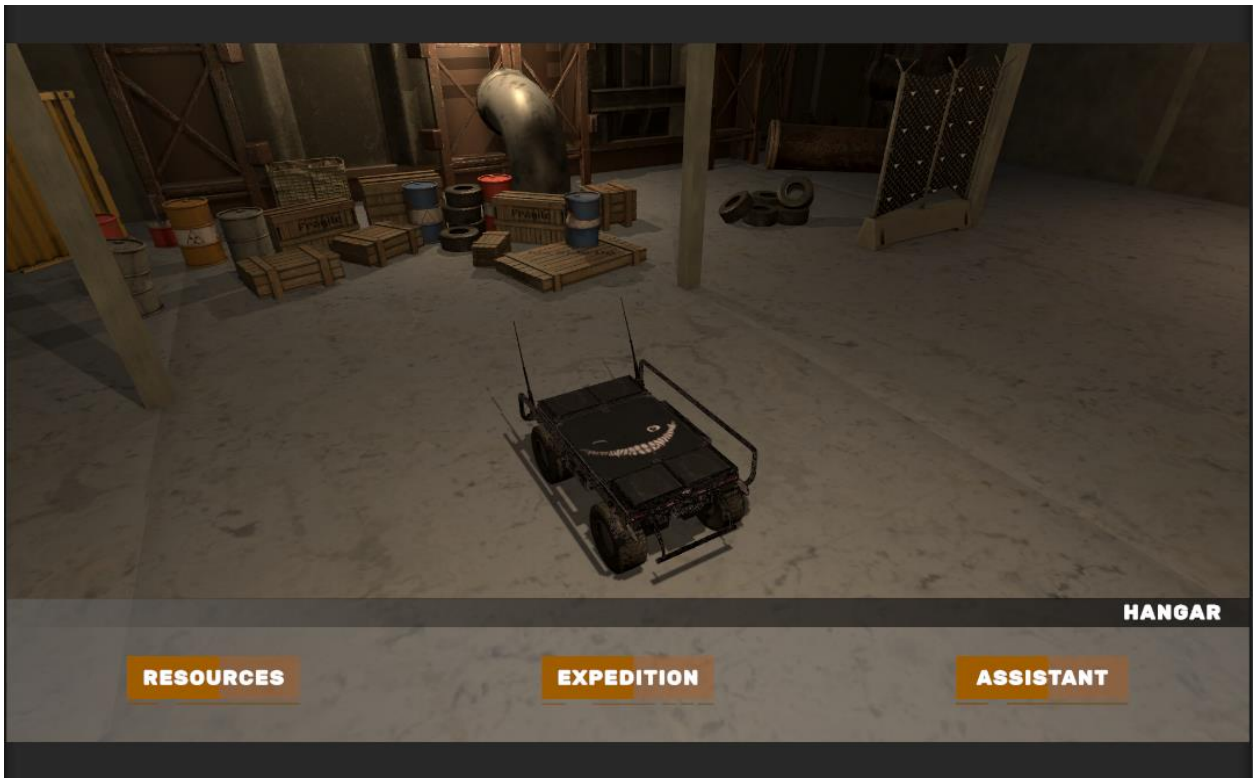


Рисунок 3.12 – Панель із дроном (без меню покращення)

Прийнято рішення реалізувати головне меню у стилі вибору «панелі», де будуть присутні такі:

– панель із наземним дроном (тут можна буде перейти і покращити його, або придбати потім новий);
– панель із ресурсами;
– панель із конвертопланом (для старту експедиції);
– панель із AI-помічником (для класифікації ресурсів).

Для цього знадобиться додати кілька віртуальних камер (Cinemachine). Налаштування цих камер не складне, включає безліч налаштувань, з допомогою яких можна створити «плавні» переходи між камери та кінцевими точками. Кодування таких переходів описано у наступному розділі.

Панель із дроном (рис. 3.12) демострує вигляд ангару, проте камера сфокусована на наземному дроні. В цій панелі буде присутню можливість покращити або придбрати інший дрон за наявності для цього відповідних ресурсів.

Рисунок 3.13 – Панель «експедиції»

Камера на рисунку 3.13 сфокусована на «кораблі», який буде транспортувати наземний дрон до «точки висадки» та евакуйовувати його назад, після закінчення «експедиції». Ця панель включає можливість почати експедицію з обраним дроном.

При натисненні на відповідні кнопки камера буде змінювати свій кут та погляд на відповідну зону в ангарі, таким чином користувач матиме розуміння з чим він взаємодіє.

3.5.2 Проєктування ігрової сцени

Ігрова сцена – найактивніше середовище, де гравець напряму взаємодіє і/або змінює оточення. Для прототипу достатньо розробити один рівень, наповнивши його вже знайденими безкоштовними моделями та ресурсами.

Оскільки стиль прототипу – постапокаліптичний, то рівень має бути доволі «темним» та «брудним». Для цього гарно вписуються сірі та коричневі тони кольорів, які нагадуватимуть пустельний регіон. У безкоштовних асетах також присутні моделі засохлих дерев, воно чудово підійдуть під стилістику рівня.

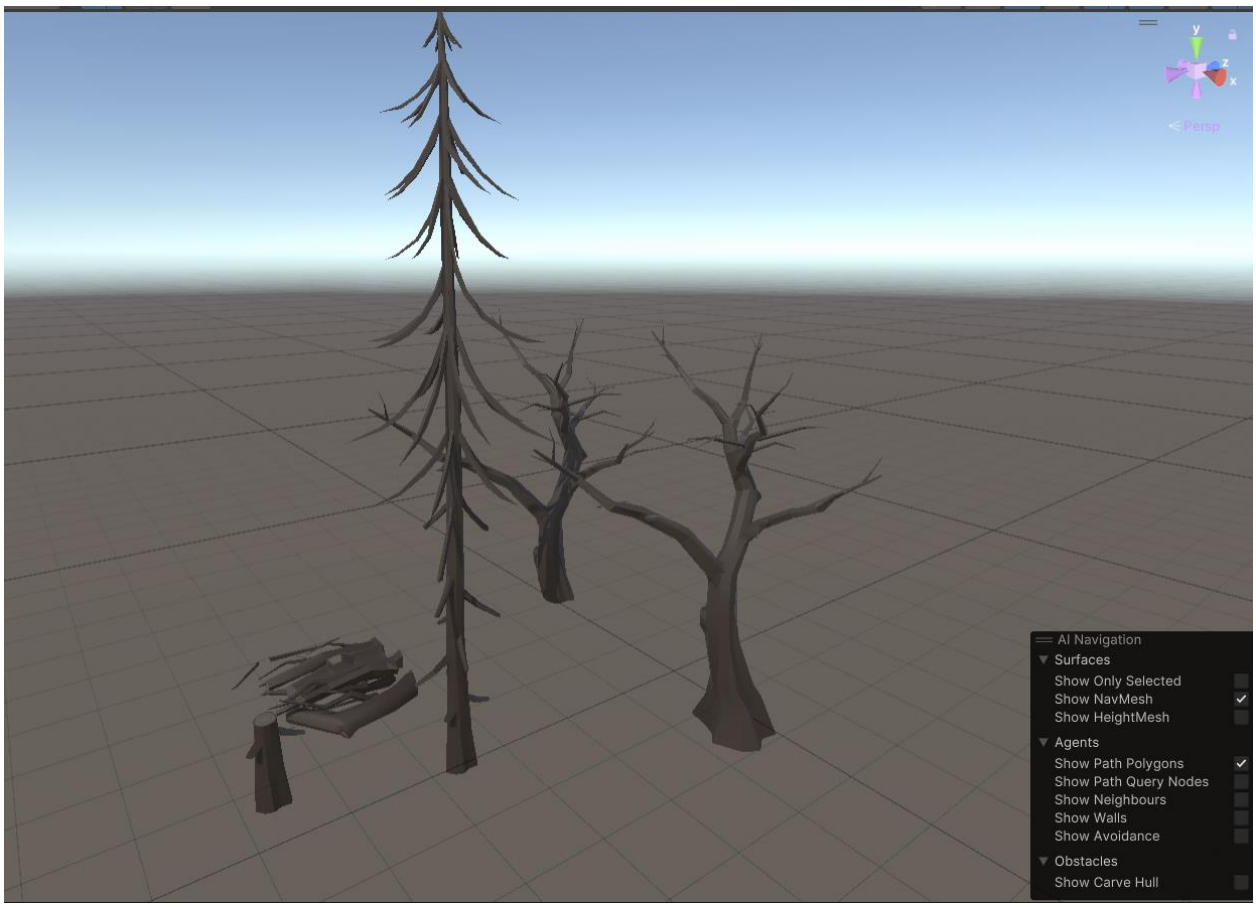


Рисунок 3.14 – Засохлі дерева

Перш ніж додавати асети на мапу ігрової, треба створити Terrain об’єкт, який в собі містить текстури, меш та інструменти для спавну об’єктів на мапі. Це вбудований асет Unity, який швидко та просто масштабується, має безліч налаштувань та можливостей для прототипування.

Створивши террейн, замінимо йому матеріал та налаштуємо текстури із вже імпортованих у проєкт. До базової текстури також треба включити normal map текстуру, щоб террейн мав певний візуальний рельєф.

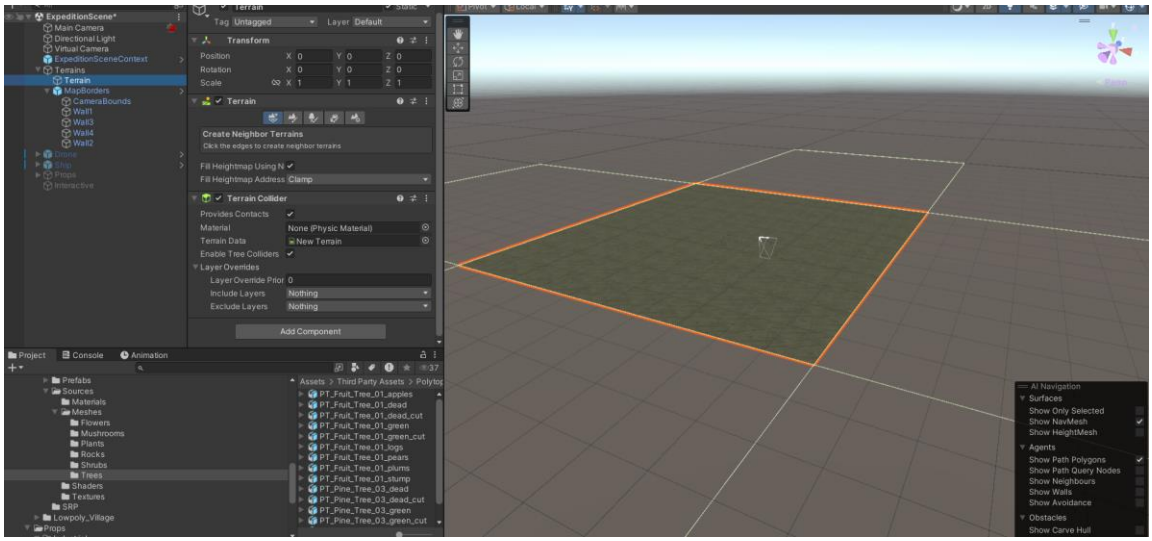


Рисунок 3.15 – Новий террейн

Цей асет має велику кількість налаштувань в інспекторі, які включають в себе:

- малювання текстурою по террейну;
- зміна висоти террейну;
- спавн дерев та трави;
- спавн спеціальних мешів;
- загальні налаштування террейну.

Окрім вищеописаного функціоналу, террейн автоматично генерує власний колайдер (фізичну форму), яку можна використовувати при зчитуванні колізій, та пересуванні юнітів, а в даному випадку – наземного дрона та роботів-NPC.

Хоч террейн і виглядає великим, але він все одно вимагає обмежень на мапі, а саме – невидимих колайдерів-стін, щоб дрон не мав змоги виїхати і впасти за межі террейну. Також додамо обмеження для віртуальної камери, щоб вона не могла виходити за межі власного Bounding Box. У Cinemachine для цього є спеціальний компонент – Cinemachine Confiner, який приймає

Collider як поле і автоматично обмежує пересування камери у межах цього колайдера.

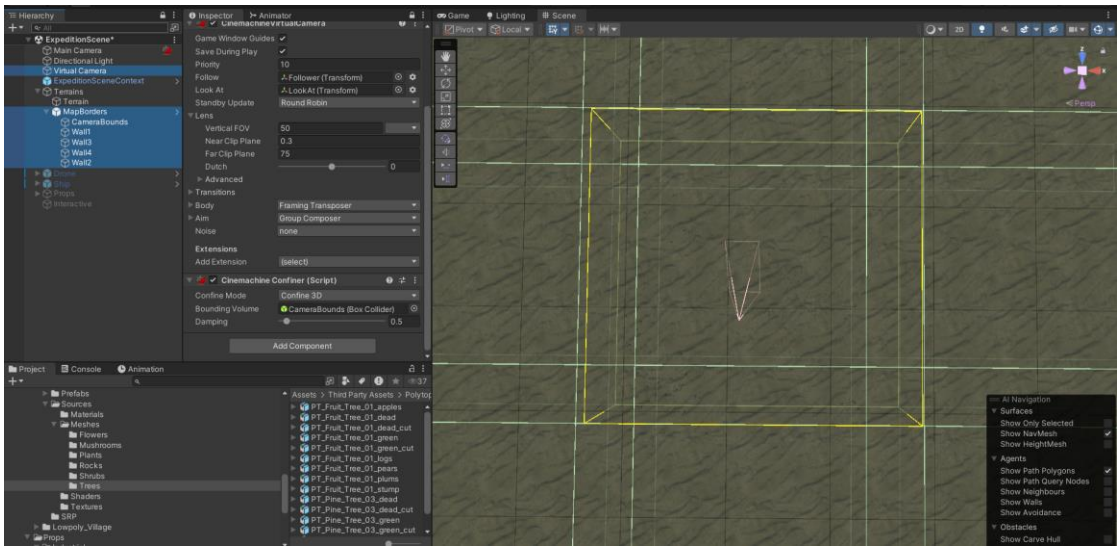


Рисунок 3.16 – Обмеження ігрового рівня та камери за допомогою колайдерів

Обмеження та камера готові, наступний крок – додавання об’єктів та пропсів на карту, щоб створити та урізноманітнити потрібні відчуття від геймплею. Для цього використано вищезгадані асети засохлих дерев та інструменти террейну для спавну спеціальних об’єктів та дерев. Додаткового до цього на мапі розмістщено такі об’єкти як:

– ящики;
– розбиті дошки;
– запчастини від роботів;
– колеса;
– покинуте авто;
– бруси, дошки та ін.

Для деяких з них додано колайдери, щоб вони теж не могли бути «прохідними» для дрона. Це додає реалістичності середовищу, оскільки воно матиме перешкоди, які заважитимуть гравцю пересуватись вільно. Ігровий рівень починає набувати потрібної форми та візуальної складової. На ньому окрім статичних об’єктів також будуть присутні інтерактивні, з якими наземний дрон зможе взаємодіяти. Гравець може використовувати перешкоди

для власного захисту від ворожих роботів, які намагатимуться його переслідувати, тому дослідження рівня стає цікавим квестом для пошуку цікавих та корисних місць.

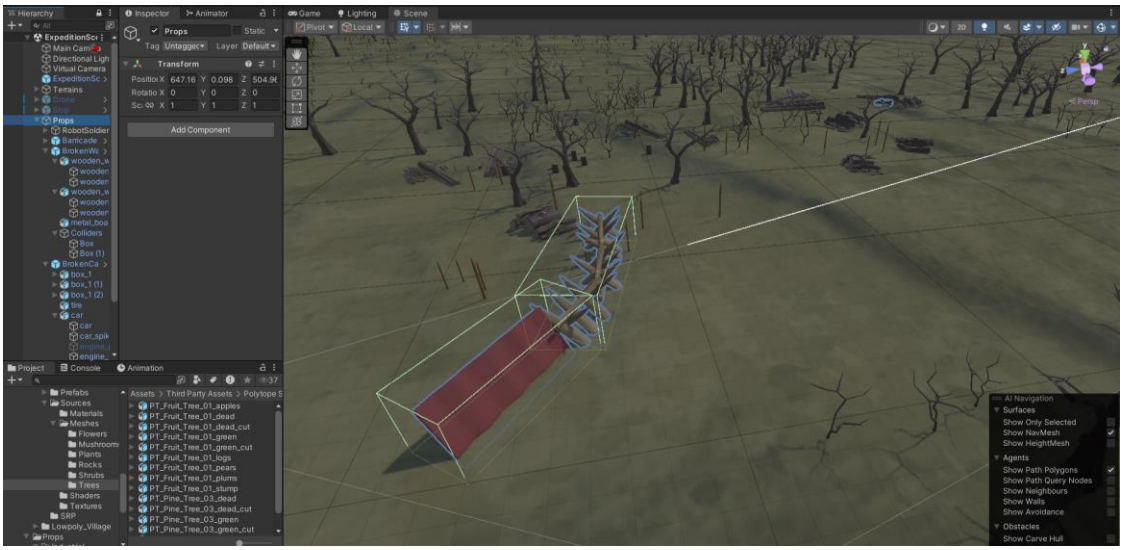


Рисунок 3.17 – Розташування додаткових об’єктів та перешкод

Наприкінці додано дрон, щоб побачити, яким чином буде виглядати камера під час справжньої гри. Трохи підналаштувавши кути, обмеження, точки слідкування, отримано ось такий результат (рис. 3.18).

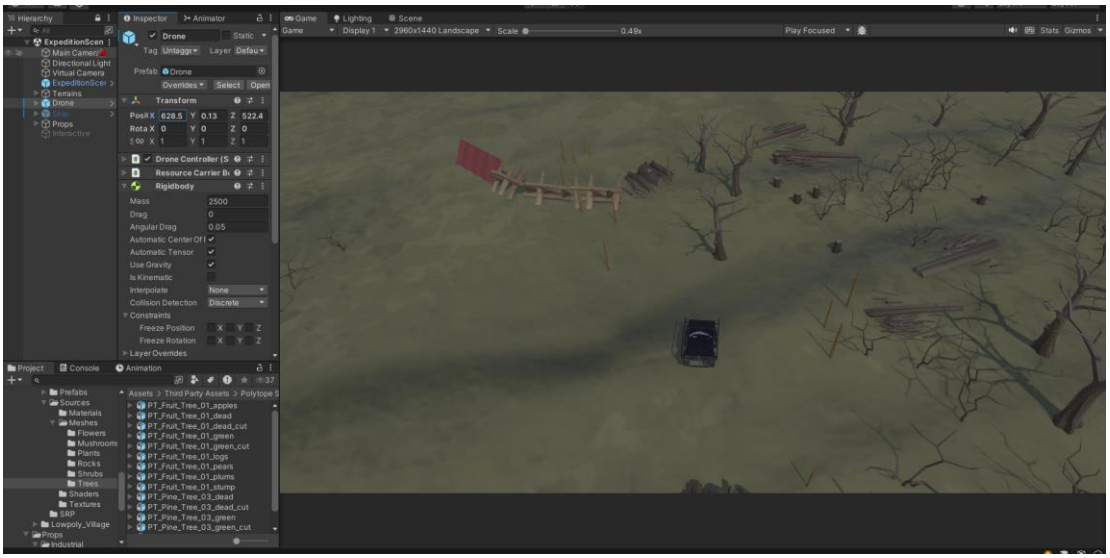


Рисунок 3.18 – Вигляд камери на ігровій сцені від POV дрона

Налаштування сцени майже завершене, наступні кроки описані в останньому розділі роботи.

Висновки до розділу 3

В третьому розділі описано проєктування класів, що будуть імплементовуватись у четвертому розділі, додано таблицю із класами, показано діаграму класів.

Представлено візуальну складову інтерфейсу ігрового застосунку. Наведено скріншоти фонових зображень сцени завантаження, згенерованих за допомогою безкоштовного сервісу Leonardo AI. Показано принцип роботи безкоштовного онлайн-сервісу генерації аудіо Suno AI. У підрозділах, пов’язаних із проєктуванням сцен продемонстровано цикл ігрових сцен, показано зовнішний вигляд головного меню та ігрової сцени з використанням віртуальних камер Cinemachine та інтегрованих безкоштовних асетів. Описано процес створення та налаштування ігрового рівня з використанням вбудованого Terrain’у.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ТА ТЕСТУВАННЯ

4.1 Кодування сцен завантаження та головного меню

Будь-яка гра починається із головного меню. Тут можна змінити налаштування, переглянути статистику, зайти в магазин – усе, що потрібно гравцю, щоб відчувати, що він впливає на гру та має можливість вибрати або придбати саме те, що йому потрібно на даний момент.

Розробка головного меню вимагає розуміння, що саме потрібно для задоволення потреб та вимог гри. У попередньому розділі було описано процес створення сцен, наведено таблицю класів та показано діаграми класів, які, в тому числі, можна віднести до логіки головного меню. Розпочати варто із менеджера завантаження сцен, адже більшість ігор починаються із, так званої, «Bootstrap scene» – сцени, яка передує головному меню, та слугує сценою для ініціалізації базових класів та залежностей і, в свою чергу, відповідає за подальше завантаження головного меню.

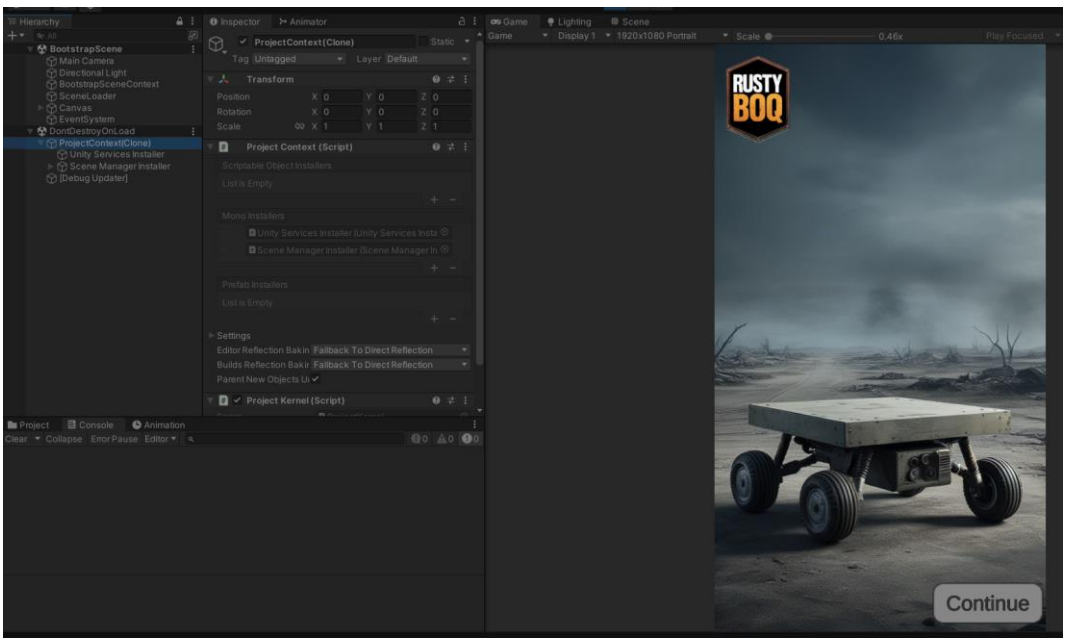


Рисунок 4.1 – Bootstrap сцена із базовими залежностями у Project Context’і

Для цього була створена така сцена, на якій і впроваджуються такі залежності за допомогою інсталерів Zenject DI фреймворку. На рис. 4.1

показано базові залежності, а саме Unity Services Installer та Scene Manager Installer. На рис. 4.2 наведено код класу SceneManagerInstaller.

```
1 asset usage  YriyQKoty
public class SceneManagerInstaller : MonoInstaller
{
    [SerializeField] private SceneSettings _sceneSettings;  Scene Settings.asset

    [SerializeField] private SceneManagerBehavior _sceneManagerBehavior;  Scene Manager Behavior (SceneManagerBehavior)

    YriyQKoty
    public override void InstallBindings()
    {
        var sceneRepository = new Dictionary<ESceneType, SceneBinding>();

        foreach (var sceneBinding in _sceneSettings.SceneBindings)
        {
            if (!sceneRepository.ContainsKey(sceneBinding.ESceneType))
                sceneRepository.Add(sceneBinding.ESceneType, sceneBinding);
        }

        Container // DiContainer
            .Bind<IReadOnlyDictionary<ESceneType, SceneBinding>>() // ConcreteIdBinderGeneric<IReadOnlyDictionary<...>>
            .WithId(identifier: "ScenesRepository") // ConcreteBinderGeneric<IReadOnlyDictionary<...>>
            .FromInstance(sceneRepository) // ScopeConcreteIdArgConditionCopyNonLazyBinder
            .AsSingle();

        Container // DiContainer
            .Bind<SceneManagerBehavior>() // ConcreteIdBinderGeneric<SceneManagerBehavior>
            .FromInstance(_sceneManagerBehavior) // ScopeConcreteIdArgConditionCopyNonLazyBinder
            .AsSingle();
    }
}
```

Рисунок 4.2 – Код класу SceneManagerInstaller

У цьому класі відбувається «встановлення» залежностей та налаштувань, пов’язаних із взаємодією сцен. Так поле **IReadOnlyDictionary<ESceneType, SceneBinding> sceneRepository** було додано до DI Container’а, та може використовуватись будь-де у коді згодом за потреби, аналогічно для **SceneManagerBehavior**. Цей клас відповідає за перемиканням між сценами, він містить кілька методів, які здійснюють завантаження потрібної сцени (приймаючи відповідні параметри про сцену) за допомогою вбудованого Unity SceneManager`а.

Методи **StartLoadScene** та **LoadAsync** показані на рис. 4.3.

Логіка доволі проста – прийняти як параметр тип сцени та завантажити її по знайденому імені, проте, щоб це спрацювало, сцени необхідно додати як активні у Build Settings вікні у проєкті (рис. 4.4).

```
1 usage  YriyQKoty
public void StartLoadScene(ESceneType sceneType)
{
    if (!scenesRepository.ContainsKey(sceneType))
    {
        Debug.LogError( message: $"There is no scene with type {sceneType} in repository!");
        return;
    }

    var sceneName :string  = scenesRepository[sceneType].SceneName;

    if (string.IsNullOrEmpty(sceneName))
    {
        Debug.LogError( message: $"Scene name is not correct!");
        return;
    }

    StartCoroutine( routine: LoadAsyncScene(sceneName));
}

Frequently called  1 usage  new *
IEnumerator LoadAsyncScene(string sceneName)
{
    AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(sceneName);

    while (!asyncOperation.isDone)
    {
        // Check if the load has finished
        if (asyncOperation.progress >= 0.9f)
        {
            asyncOperation.allowSceneActivation = true;
        }

        yield return null;
    }
}
```

Рисунок 4.3 – Методи StartLoadScene та LoadAsyncScene

Метод **StartLoadScene** перевіряє чи вказана у параметрах сцена справді існує (знаходиться у репозиторії сцен, що були встановлені в інсталері із налаштувань), чи ім’я не нульове та розпочинає Coroutine. Coroutine (карутина) – механізм Unity, який дозволяє виконувати асинхронну логіку, не блокуючи головного потоку, метод StartCoroutine вбудований та доступний лише у MonoBehavior класах та їх нащадках. У методі **LoadAsyncScene** вбудований у Unity менеджер сцен починає процес вантаження сцени та активує її при досягненні 90% готовності, після цього обрана сцена показується на екрані.

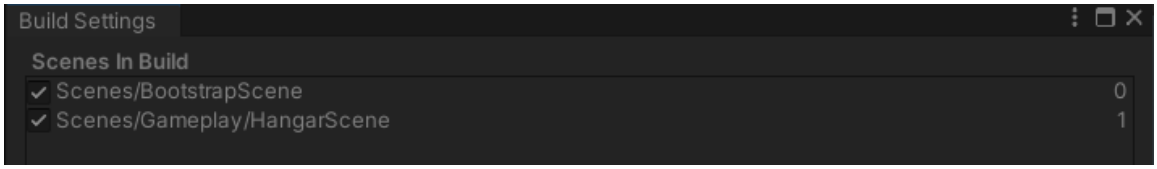


Рисунок 4.4 – Налаштування активних сцен у проєкті

Для виклику методу початку завантаження сцени було також додано **MonoBehavior** клас **SceneLoaderBehavior**, код наведено на рис. 4.5.

Логіка проста, для роботи потрібна кнопка (вбудований елемент Unity), тип сцени, підписка та відписка на подію кліку.

```
1 asset usage  YriyQKoty
public class SceneLoaderBehavior : MonoBehaviour
{
    [Inject] private SceneManagerBehavior _sceneManagerBehavior;

    public ESceneType sceneType;  Hangar
    public Button loadButton;  ContinueBtn (Button)

    public TextMeshProUGUI loadingPercentageTxt;  Unchanged

    Event function  YriyQKoty
    private void Awake()
    {
        loadButton.onClick.AddListener( call: () =>
        {
            _sceneManagerBehavior.StartLoadScene(sceneType);
        });
    }

    Event function  YriyQKoty
    private void OnDestroy()
    {
        loadButton.onClick.RemoveAllListeners();
    }
}
```

Рисунок 4.5 – Клас для перемикання сцени

Цей клас можна перевикористовувати, тому його буде додано як для переходу в головне меню з початкової сцени, так і в ігрову сцену для початку експедиції, ну і звісно ж для повернення назад у головне меню після закінчення експедиції. MonoBehavior класи «навішуються» на GameObject`и – ігрові об’єкти в Unity, які є контейнерами для ігрових скриптів, які містять ігрову логіку.

На рис. 4.6 показано ігровий об’єкт, на якому додано клас SceneLoaderBehavior, туди додані залежності від кнопки та вказано тип сцени, яку необхідно завантажити по натисненню на кнопку.

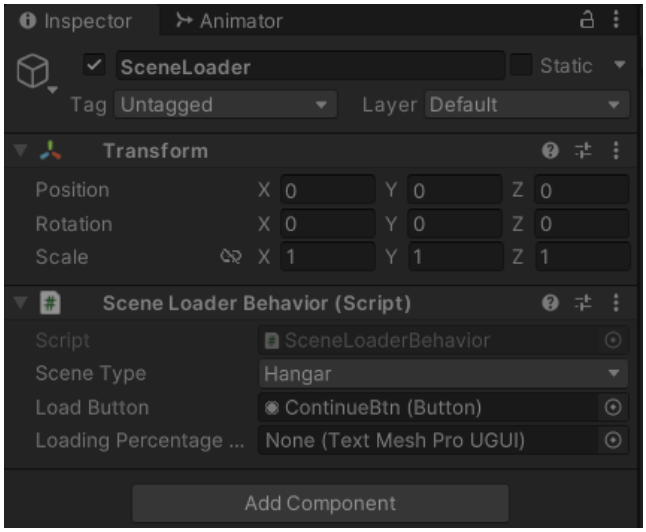


Рисунок 4.6 – Ігровий об’єкт зі скриптом

Переходимо до головного меню. В меню, як вже було описано в попередньому розділі, присутні панелі для дрона, експедиції, ресурсів та асистента. Для них було створено ряд класів, деякі з них будуть наведені нижче. Основна логіка головного меню – переміщення камери до панелей та зміна сцени на ігрову. На рис. 4.7 показано клас-інсталер залежностей контролера камери в ангарі.

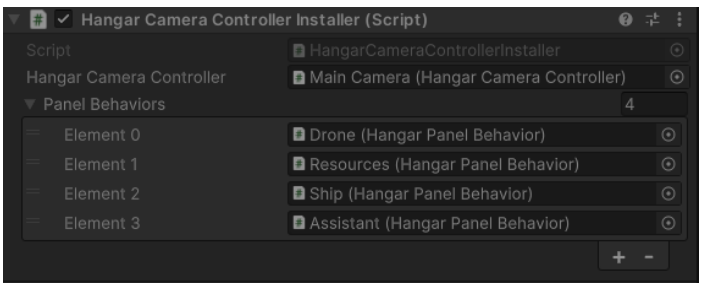


Рисунок 4.7 – Інсталер контролера камери в ангарі

Камера повинна «розуміти» на які позиції їй потрібно «переїджати» та під якими кутами «дивитись» на сцену, тому для кожної з панелей були додані

позиції прив’язки для камери, які, в свою чергу, зберігаються як посилання у класі **HangarCameraPointsContainer**.

```
[Serializable]
4 usages  YriyQKoty  1 exposing API
public class HangarCameraPointsContainer
{
    public CinemachineVirtualCamera VirtualCamera;  Serializable

    [field: SerializeField]
    1 usage
    public Transform CameraLookAt { get; private set; }

    [field: SerializeField]
    1 usage
    public Transform CameraFollower { get; private set; }
}
```

Рисунок 4.8 – Клас, який зберігає позиції для камери

Зміна позицій відбувається по натисненню на кнопки панелей, які прив’язані до конкретних об’єктів на сцені, логіка зміни прописана у **HangarCameraController**`і. Метод **ChangeCameraView** (рис. 4.9) перевіряє чи є контейнер за вказаним типом у репозиторії панелей та змінює поточний контейнер з позиціями.

```
1 usage  YriyQKoty
public void ChangeCameraView(EHangarPanelType panelType)
{
    if (!panelContainersRepository.ContainsKey(panelType))
        return;

    var container = panelContainersRepository[panelType];

    if (container != CurrentContainer)
    {
        CurrentContainer?.VirtualCamera.gameObject.SetActive(false);

        CurrentContainer = container;
        CurrentContainer.VirtualCamera.gameObject.SetActive(true);
    }

    container.VirtualCamera.Follow.position = container.CameraFollower.position;
    container.VirtualCamera.LookAt.position = container.CameraLookAt.position;

    PanelChangedAction?.Invoke(panelType);
}
```

Рисунок 4.9 – Метод ChangeCameraView

Також метод містить Action делегат, який підписується на зміну панелей по натисненню на відповідну кнопку на користувацьому інтерфейсі.

Камера змінюється та працює автоматично завдяки вже описаному пакету від Unity Cinemachine. Налаштування двох віртуальних камер зроблені на сцені та підв’язані під усі позиції. На рис. 4.10 показано як виглядає віртуальна камера, зафіксована на панелі із асистентом.

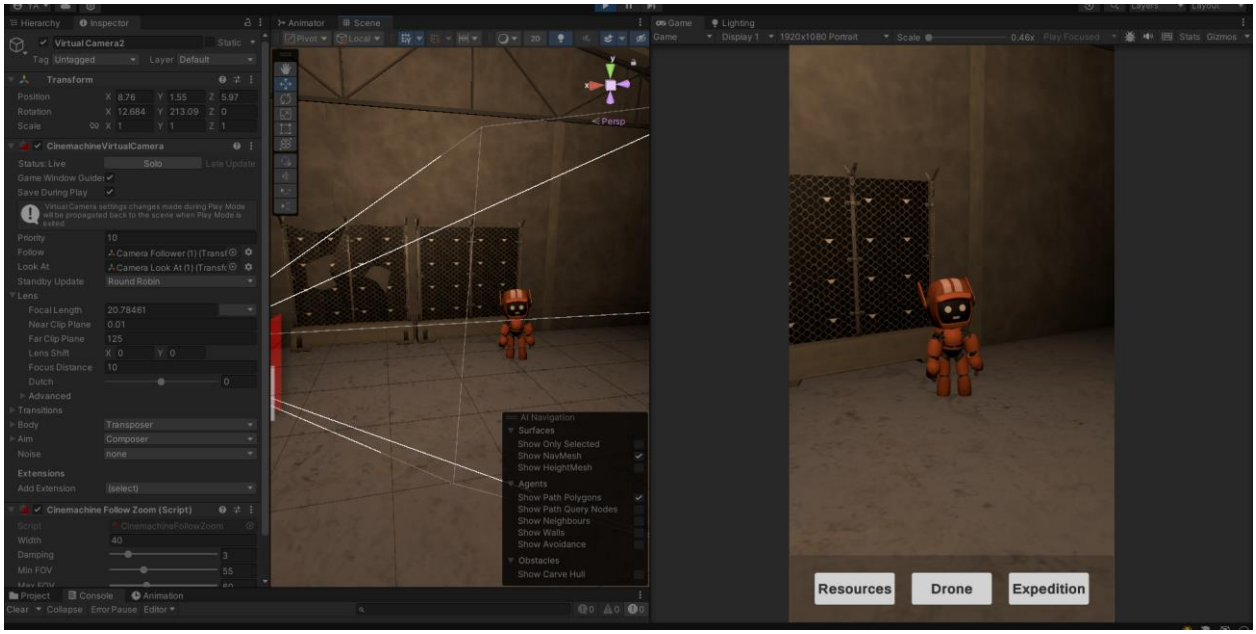


Рисунок 4.10 – Налаштування віртуальної камери та центрування на асистенті

Аналогічні налаштування зроблені для панелі з ресурсами, дроном та кораблем для експедицій. На корабель експедиції також був доданий скрипт для зміни сцени на ігрову.

4.2 Кодування пересування наземного дрона

Пересування наземного дрона є основною механікою гри, адже саме завдяки її є можливість збирати ресурси, тому її реалізація є першочерговим завданням. Оскільки обрана модель дрона має 4-ьох колісну базу, то реалізація і буде включати 2 рульових колеса та 2 задні. Для цього використаємо вбудований в Unity Wheel Collider, який автоматично взаємодіє із террейном, проте вимагає попереднього налаштування. На рис. 4.11 показано параметри, які дозволяють колесам правильно колайдитись із террейном.

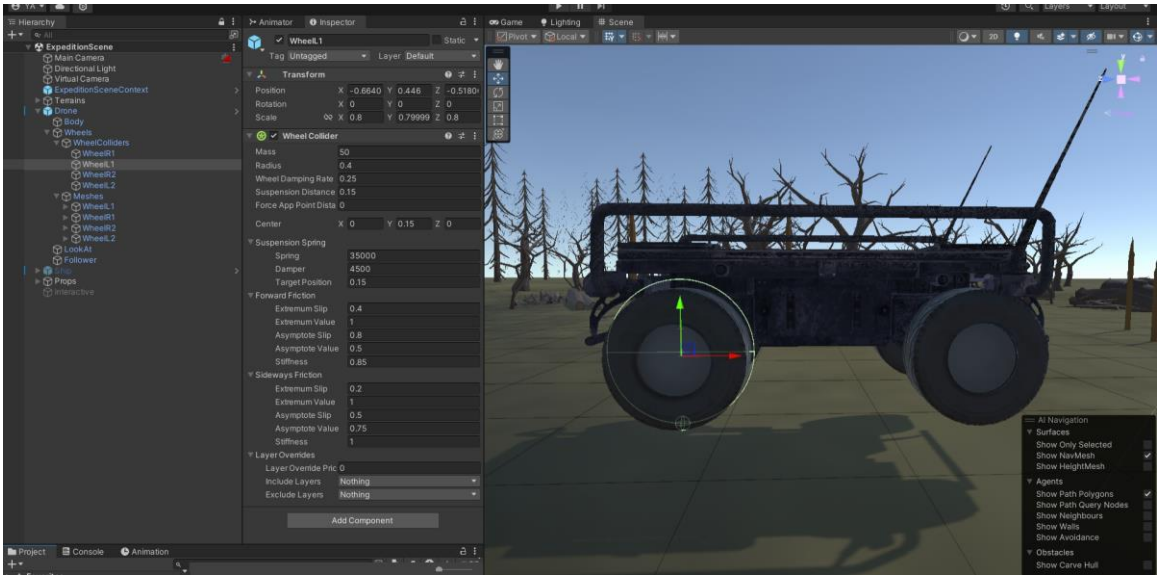


Рисунок 4.11 – Налаштування Wheel Collider

Колайдери самі по собі не можуть пересуватись, тому необхідно імплементувати використання фізики. Для цього написаний клас **DroneController**, який також «навішений» на модель наземного дрона. Він включає багато методів та параметрів, які вже описані у таблиці в 3 розділі, та присутні на діаграмі класів у додатку Б. Основа логіка полягає у наданні колесам «сили», яка передається користувацьким вводом, після цього дрон розганяється та підпадає під обмеження по максимальній швидкості. Також реалізовано задній хід, для нього швидкість нижча, ніж для переднього.

```
Frequently called  1 usage  YriyQKoty
public void Steer(float horizontalInput, float verticalInput)
{
    var maxSpeedKph:float = GetMaxSpeed(verticalInput);

    float speedFactor = rigidbody.velocity.sqrMagnitude / maxSpeedKph ;

    var steeringAngle:float = Mathf.Lerp(a: maxSteeringAngle, b: 0, t: Time.deltaTime * speedFactor) * horizontalInput;

    leftWheel.Collider.steerAngle = Mathf.Lerp(a: leftWheel.Collider.steerAngle, b: steeringAngle,
        t: Time.fixedDeltaTime * steeringSpeed);
    rightWheel.Collider.steerAngle = Mathf.Lerp(a: rightWheel.Collider.steerAngle, b: steeringAngle,
        t: Time.fixedDeltaTime * steeringSpeed);
}
```

Рисунок 4.12 – Метод Steer для повороту дрона

Поворот дрона імплементовано за допомогою **Steer** методу, який також включає користувацький ввід (рис. 4.12).

Користувацький ввід в свою чергу зчитується в Update методі контроллера, після чого з його використанням оброблюються методи гальмування, пересування, повороту та прискорення (рис. 4.13).

```
Event function  YriyQKoty  More
private void Update()
{
    var horizontalInput float = Input.GetAxis("Horizontal");

    var verticalInput float = Input.GetAxis("Vertical");

    if (verticalInput == 0)
        Brake();
    else
        StartMove();

    Steer(horizontalInput, verticalInput);
    Accelerate(verticalInput);

    foreach (var wheel in wheels)
    {
        wheel.UpdateWheel();
    }
}
```

Рисунок 4.13 – Update метод із логікою пересування дрона

У методі Fixed Update застосовується «стабілізаційна» сила, щоб дрон «не перекидався» при різких поворотах під час пересуванні по террейну (рис. 4.14). Ця сила вираховується із точок колізій колес із террейном, та прикладається до центрів коліс, щоб вони не могли перевертатись при швидких поворотах.

```
Event function  YriyQKoty
private void FixedUpdate()
{
    WheelHit hit;
    float travelL = 1.0f;
    float travelR = 1.0f;

    bool groundedL = leftWheel.Collider.GetGroundHit(out hit);
    if (groundedL)
        travelL = (-leftWheel.Collider.transform // Transform
            .InverseTransformPoint(hit.point).y - leftWheel.Collider.radius) / leftWheel.Collider.suspensionDistance;

    bool groundedR = rightWheel.Collider.GetGroundHit(out hit);
    if (groundedR)
        travelR = (-rightWheel.Collider.transform // Transform
            .InverseTransformPoint(hit.point).y - rightWheel.Collider.radius) /rightWheel.Collider.suspensionDistance;

    float antiRollForce = (travelL - travelR) * antiRoll;

    if (groundedL)
        rigidbody.AddForceAtPosition( force: leftWheel.Collider.transform.up * -antiRollForce,
            leftWheel.Collider.transform.position);
    if (groundedR)
        rigidbody.AddForceAtPosition( force: rightWheel.Collider.transform.up * antiRollForce,
            rightWheel.Collider.transform.position);
}
```

Рисунок 4.14 – Fixed Update метод із логікою стабілізації коліс

Додатково, на кожне з коліс також прикріплений спеціальний клас Wheel, який оновлює їх «rotation», адже при пересуванні колеса прокручуються, та вимагають зміни їхнього кута навколо осі.

```
[Serializable]
3 usages  YriyQKoty
public class Wheel
{
    public Transform WheelMesh;  Serializable
    public WheelCollider Collider;  Serializable

    Frequently called  1 usage  YriyQKoty
    public void UpdateWheel()
    {
        Vector3 pos = WheelMesh.position;
        Quaternion rot = WheelMesh.rotation;

        Collider.GetWorldPose(out pos, out rot);

        WheelMesh.position = pos;
        WheelMesh.rotation = rot * Quaternion.Euler(x: 0, y: 90, z: 0);
    }
}
```

Рисунок 4.15 – Код класу Wheel

4.3 Інтеграція Open AI плагіну

У грі заплановано мати асистента, який буде допомагати описувати недосліджені предмети та видавати дані про їх характеристики. Для цього нам знадобиться інтегрувати плагін OpenAI Unity, налаштувати dashboard та використати наявний функціонал інтегрованого плагіну.

4.3.1 Open AI

OpenAI – американська організація із дослідження штучного інтелекту, створена у 2015 році у Сан-Франциско, Сполучені Штати Америки. Її ціль – пошук та розробка безпечних, стабільних та автономних моделей ШІ з метою широкого використання у всіх сферах інформаційних технологій. Станом на 2024 рік до них належать такі:

- сімейство GPT-моделей;
- DALL-E генерація зображень з тексту
- Sora генерація відео з тексту та інші.

Найвідомішим, звісно ж, став ChatGPT. Сьогодні він є невід'ємною частиною при розробці застосунків, у ролі помічника, консультанта, оброблювача текстів, перекладача – усього, що ChatGPT здатен пришвидшити та спростити як процесу обробки тексту.

В ігрових застосунках так само можна використовувати подібні моделі машинного навчання та штучного інтелекту для ігрових механік. У проєкті, що розробляється, запланована імплементація логіки AI-асистента (дружнього робота в ангарі, вже показаного на рис. 4.10) з використанням OpenAI API, а саме, за допомогою безкоштовного плагіну OpenAI Unity. Цей плагін містить усі необхідні методи та класи для взаємодії із більшістю функціоналу, який OpenAI надає розробникам через їх власну API. Проте, немає гострої потреби у використанні усіх наявних функцій, лише у генерації описів «невідомих» гравцю предметів, які той буде збирати під час експедиції на ігровій мапі. Для такої генерації достатньо звичайної GPT-3.5 моделі OpenAI.

4.3.2 Dashboard Open AI

OpenAI, як і будь-який інший надавач послуг, пропонує широкий функціонал та менеджемент проєктів (рис. 4.11) з дуже детальною документацією [29]. Під проєкт треба створити спеціальний API-ключ, який буде зберігатись в окремому хмарного сховищі (Unity Remote Config), щоб не втратити доступ до проєкту.

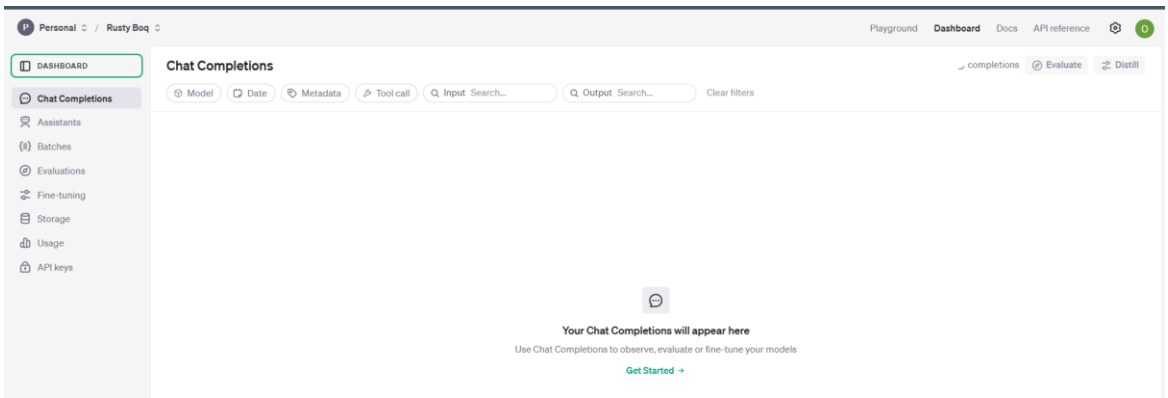


Рисунок 4.16 – Дашборд OpenAI зі створеним Rusty Boq проєктом

GPT оперує поняттям **токенів. Токени –** набори символів, які зазвичай складають більшу частину слова, які GPT опрацьовує, і, відповідно, повертає результат користувачу. Фактично, кожен запит (промт), який задає користувач складається із певної кількості токенів, які вже класифіковані у моделі. На рис. 4.17 показано яким чином модель розбиває речення на токени.

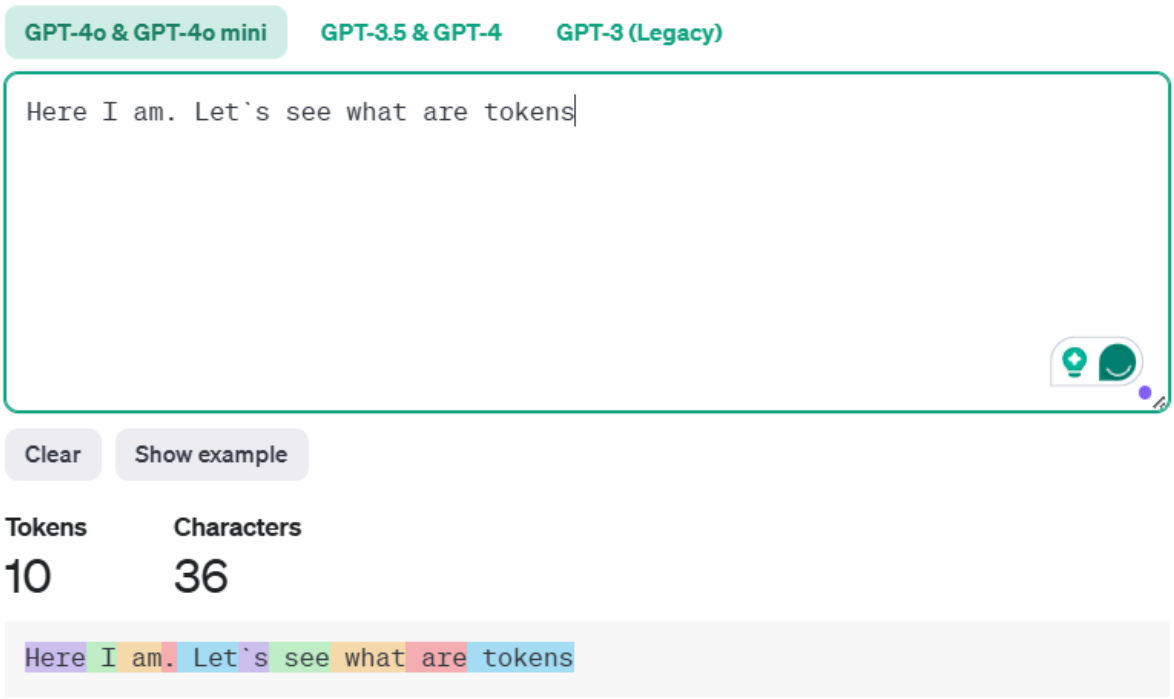


Рисунок 4.17 – Демонстрація «розбиття» речення на токени

Токени враховуються API як у складі промту, так і в складі відповіді, тобто речення промту + речення відповіді сумуються, і результуюча кількість токенів оцінюється API. На жаль, у 2024 році, OpenAI припинила безкоштовно надавати кредити для досліджень функціоналу API, тому довелось підвищити рівень User Tier до «платного», закинувши на платформу 5$. Цього достатньо для поточних потреб проєкту, деталі по витратах грошей та токени та запити можна дізнатись у вкладці Usage на дашборді.

В залежності від рівня організації, API виставляє певні обмеження (ліміти) на використання, а саме:

– TPM – токенів на хвилину
– RPM – запитів на хвилину

– RPD – запитів на день
– TPD – токенів на день.

Кожна із моделей сімейства GPT (GPT-3.5 turbo, GPT-4.0, GPT-4.0-mini тощо) мають власні ліміти, частково показані на рис. 4.18, їх також можна кастомізувати за потреби проєкту.

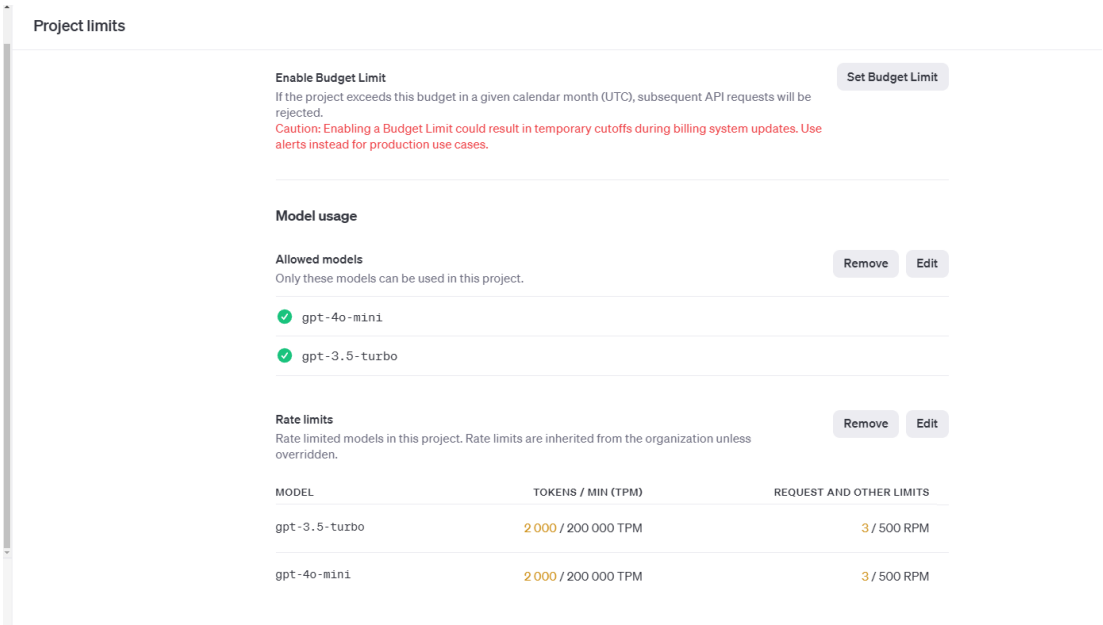


Рисунок 4.18 – Ліміти запитів та токенів за хвилину

Для застосунку буде використовуватись модель gpt-3.5-turbo, адже вона має більший ліміт на к-ть токенів та в цілому достатня для досить тривіальної задачі, як генерації опису предметів.

4.3.3 Використання плагіну OpenAI Unity

Інтеграція цього плагіну полягатиме у надсиланні запитів на Open AI у «потрібному» для застосунка місці, а саме – при «класифікації» невідомих запчастин/предметів. Надсилання цих запитів буде відбуватись у панелі AI-асистента, який знаходиться в меню-ангарі.

Проте, для того, щоб почати інтеграцію, необхідно створити репозиторій для інвентарю гравця, аби запам’ятовувати відповіді Chat`у GPT вже у зрозумілому для застосунку форматі та перевикористовувати отриманий опис предметів та їх статистик повторно для уникнення додаткових запитів на Open AI. Для цього створимо Scriptable об’єкт та менеджер для інвентарю, щоб

взаємодіями із предметами, додавати нові, прибирати та інші стандартні дії із репозиторіями. Клас InventoryItem (рис. 4.19) містить усі необхідні дані для запитів у Open AI, даних відображення (іконка, текст), поля для кешу (після першого запиту дані будуть зберігатись і перевикористовуватись щоразу надалі).

```
[Serializable]
13 usages  YriyQKoty *  2 exposing APIs
public class InventoryItem
{
    public string ItemId;  Serializable

    public string Name;  Serializable

    public bool NeedsAIAssistantClassification;  Serializable

    [TextArea]
    public string ChatPromt;  Serializable

    public Sprite Icon;  Serializable

    public bool Cached;  Serializable
    public string CachedDescription;  Serializable
    public string CachedStatsText;  Serializable

    public uint Count;  Serializable
}
```

Рисунок 4.19 – InventoryItem клас

У класі InventoryManager (рис. 4.20) присутні методи для отримання, додавання та видалення предметів інвентарю. Окремо додано словник типу рядок-предмет для прототипів предметів. Це зроблено з метою розмежування усіх можливих предметів та тих предметів, які гравець отримав під час гри в експедиції. Також клас містить метод для зміни стану предмета із невідомого на класифікований – тут змінюватиметься стан на закешований та зберігатиметься у Unity Cloud Save для збереження прогресу та даних, отриманих від Chat GPT.

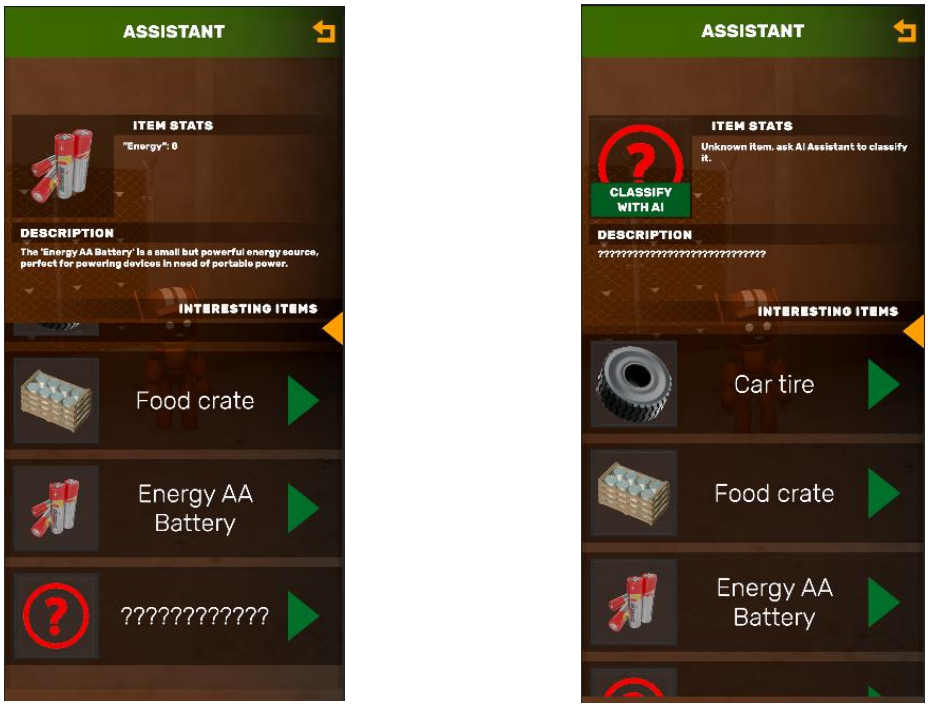
Усі знайдені предмети будуть з’являтись в панелі AI-асистента і матимуть два стани: потребує класифікації та класифікований. Для того, щоб використовувати предмет для, наприклад, покращення поточного дрону

(купівлі нового) або обміну предметів у панелі ресурсів, треба спершу дізнатись що це за предмет і які характеристики він може покращити.

```
3 usages
public class InventoryManager
{
    private Dictionary<string, InventoryItem> prototypeItemsRepository;

    private Dictionary<string, InventoryItem> collectedItemsRepository;

    public List<string> collectedItemsPrototypes = new List<string>();

    1 usage
    public InventoryManager(Dictionary<string, InventoryItem> prototypeItemsRepository){...}

    2 usages
    public bool TryGetInventoryItem(string itemId, out InventoryItem result){...}

    public InventoryItem GetItem(string itemId) => collectedItemsRepository[itemId];

    2 usages
    public bool TryAddItemToInventory(string itemId, uint count = 1){...}

    1 usage
    public bool TryRemoveItemFromInventory(string itemId, uint count = 1){...}

    1 usage
    public void CacheDiscoveredItem(InventoryItem item){...}

    1 usage
    public uint GetInventoryItemCount(string itemId) => collectedItemsRepository[itemId].Count;
}
```

Рисунок 4.20 – Клас InventoryManager

Для цього треба перейти до панелі асистента, обрати «невідомий» предмет та натиснути «Classify with AI». Після цього предмет отримає свій опис, буде збереженим та доступним для вищеописаних операцій (рис. 4.21).



а) б)

Рисунок 4.21 – а) класифікований предмет; б) предмет, який треба класифікувати

4.4 Інтеграція Unity ML пакета

Окрім стандартних бібліотек та функціональності (таких як фізика, математика тощо), Unity пропонує ряд додаткових пакетів та бібліотек, які можна користовувати за потреби. Unity ML Agents є таким пакетом, який дозволяє використовувати алгоритми машинного навчання для розробки ігрових застосунків.

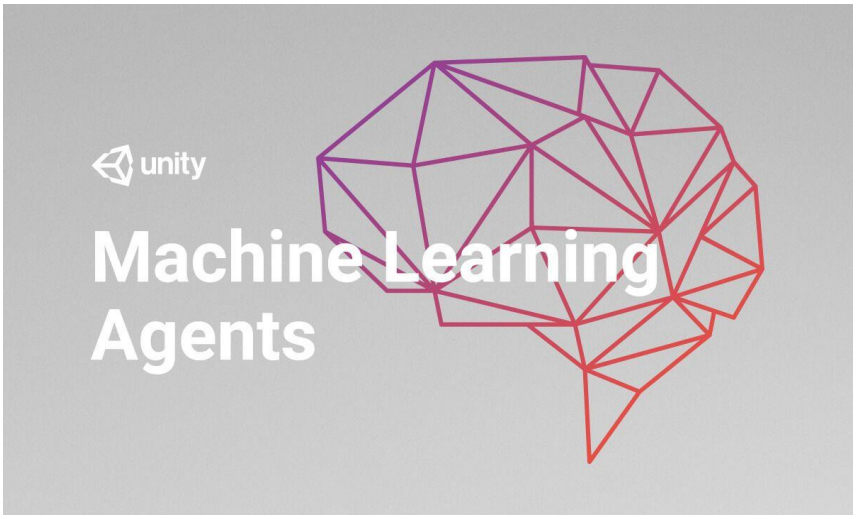


Рисунок 4.22 – Логотип ML Agents

У третьому розділі було показано встановлення цього пакету, проте окрім нього треба ще розгорнути локальне середовище для навчання агентів. Для цього треба встановити мову програмування Python та бібліотеку ML PyTorch для налаштування локального сервера під навчальні потреби. Встановити Python можна через офіційний сайти.

```
C:\Users\ura20\UnityProjects\RustyBoq>py
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Рисунок 4.23 – Встановлена версія Python

Перевіримо, що Python справді встановлено через виклик у консолі (рис. 4.23). Далі необхідно встановити PyTorch, проте для цього треба мати готове віртуальне середовище та менеджер пакетів. На рис. 4.24 показно команди, за

допомогою якихстворено віртуальне середовище та додано pip (пакетний менеджер Python).

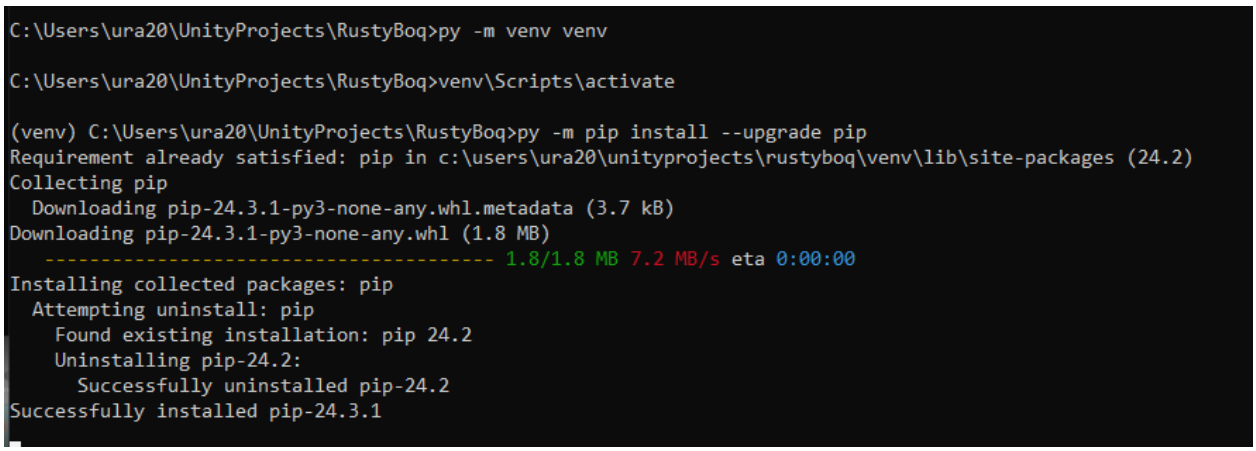


Рисунок 4.24 – Налаштування віртуального середовища та pip

Наступний крок – встановлення пакетів mlagents та PyTorch. Команди схожі (рис. 4.25-26). Дочекавшись встановлення mlagents пакета, встановлюємо і PyTorch.

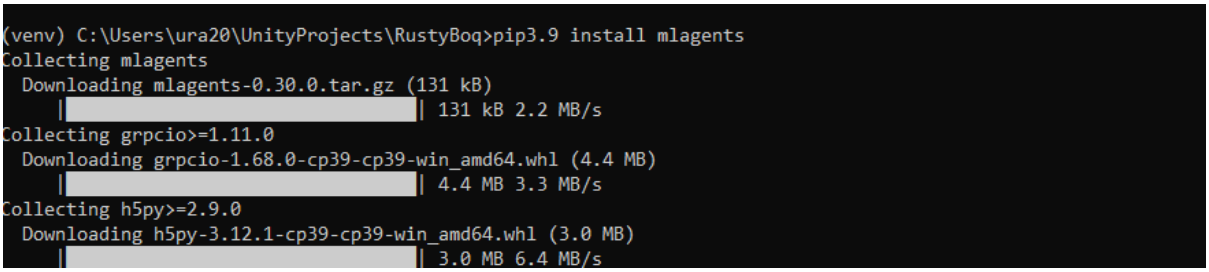


Рисунок 4.25 – Встановлення mlagents пакета

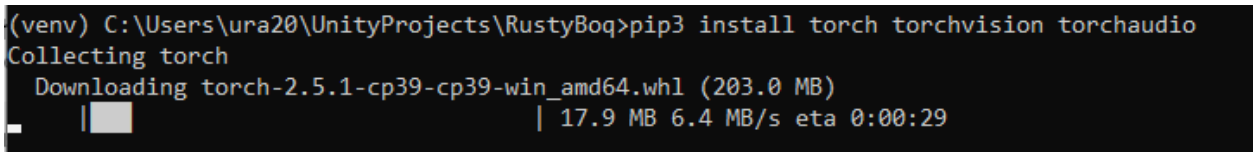


Рисунок 4.26 – Встановлення PyTorch

Ці бібліотеки використовуються Unity ML Agents для розгортання локального середовища машиного навчання для агентів. У випадку проєкту, що розробляється, такими агентами будуть роботи-NPC. Для перевірки коректного встановлення треба ввести команду *mlagents-learn -h* (рис. 4.27).

```
(venv) C:\Users\ura20\UnityProjects\RustyBoq>mlagents-learn -h
c:\users\ura20\unityprojects\rustyboq\venv\lib\site-packages\torch\__init__.py:1144: UserWarning: torch.set_default_tensor_type() is deprecated as of PyTorch 2.1, please use to
et_default_dtype() and torch.set_default_device() as alternatives. (Triggered internally at C:\actions-runner\_work\pytorch\pytorch\builder\windows\pytorch\torch\csrc\tensor\py
tensor.cpp:434.)
  _C._set_default_tensor_type(t)
usage: mlagents-learn [-h] [--env ENV_PATH] [--resume] [--deterministic] [--force] [--run-id RUN_ID] [--initialize-from RUN_ID] [--seed SEED] [--inference] [--base-port BASE_PO
                      [--num-envs NUM_ENVS] [--num-areas NUM_AREAS] [--debug] [--env-args ...] [--max-lifetime-restarts MAX_LIFETIME_RESTARTS]
                      [--restarts-rate-limit-n RESTARTS_RATE_LIMIT_N] [--restarts-rate-limit-period-s RESTARTS_RATE_LIMIT_PERIOD_S] [--torch] [--tensorflow]
                      [--results-dir RESULTS_DIR] [--width WIDTH] [--height HEIGHT] [--quality-level QUALITY_LEVEL] [--time-scale TIME_SCALE]
                      [--target-frame-rate TARGET_FRAME_RATE] [--capture-frame-rate CAPTURE_FRAME_RATE] [--no-graphics] [--torch-device DEVICE]
                      [trainer_config_path]

positional arguments:
  trainer_config_path

optional arguments:
  -h, --help            show this help message and exit
  --env ENV_PATH        Path to the Unity executable to train (default: None)
  --resume              Whether to resume training from a checkpoint. Specify a --run-id to use this option. If set, the training code loads an already trained model to initiali
                        the neural network before resuming training. This option is only valid when the models exist, and have the same behavior names as the current agents in
                        your scene. (default: False)
  --deterministic       Whether to select actions deterministically in policy. `dist.mean` for continuous action space, and `dist.argmax` for deterministic action space (default
                        False)
  --force               Whether to force-overwrite this run-id's existing summary and model data. (Without this flag, attempting to train a model with a run-id that has been us
                        before will throw an error. (default: False)
  --run-id RUN_ID       The identifier for the training run. This identifier is used to name the subdirectories in which the trained model and summary statistics are saved as w
```

Рисунок 4.27 – Відображення вікна help пакету mlagents

4.5 Навчання NPC

Для навчання NPC необхідно створити відповідне середовище, агентів та провести ряд «навчань» агентів, щоб отримати в результаті «мізки», тобто модель нейронної мережі, які і будуть використовуватись для пересування NPC агента.

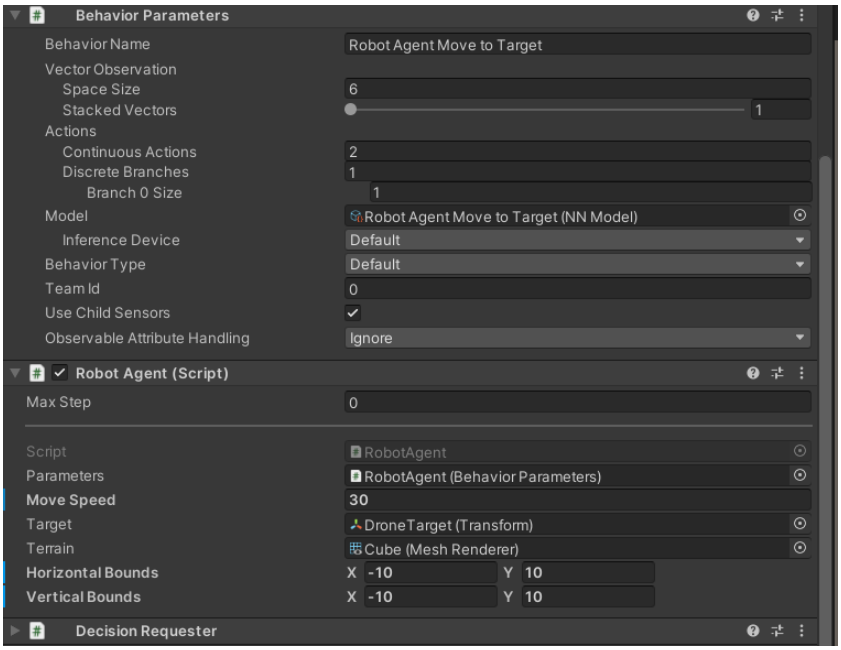


Рисунок 4.28 – ML скрипти Robot агента в інспекторі

Для налаштування агента, треба створити скрипт, який буде наслідуватись від загального класу Agent та перевизначати логіку деяких методів, які вказані на рисунках нижче (рис. 4.28).

```
public override void CollectObservations(VectorSensor sensor)
{
    var position:Vector3 = transform.localPosition;
    var targetPos:Vector3 = target.localPosition;

    sensor.AddObservation(new Vector3(position.x, y: 0, position.z));
    sensor.AddObservation(new Vector3(targetPos.x, y: 0, targetPos.z));
}

public override void Heuristic(in ActionBuffers actionsOut)
{
    ActionSegment<float> continuousActions = actionsOut.ContinuousActions;

    continuousActions[0] = Input.GetAxis("Horizontal");
    continuousActions[1] = Input.GetAxis("Vertical");
}
```

Рисунок 4.29 – Методи CollectObservation та Heuristic

Метод CollectObservation записує у сенсор останні дані про ціль та самого агента, які згодом і використовуються для посиленого навчання. Метод Heuristic використовується переважно для тестування логіки в самому середовищі перед початком навчання.

```
public override void OnActionReceived(ActionBuffers actions)
{
    float moveX = actions.ContinuousActions[0];
    float moveZ = actions.ContinuousActions[1];

    var direction = new Vector3(moveX, y: 0, moveZ);
    transform.localPosition += direction * Time.deltaTime * moveSpeed;

    if (direction != Vector3.zero)
    {
        transform.rotation =  Quaternion.Slerp (
            a: transform.rotation,
            b: Quaternion.LookRotation (direction),
            t: Time.deltaTime * 15f);
    }

    float distanceToTarget = Vector3.Distance(a: this.transform.position,
        b: target.position);

    if (distanceToTarget < 3f)
    {
        SetReward(15.0f);
        if (parameters.BehaviorType != BehaviorType.InferenceOnly)
            terrain.material.color = Color.green;
        EndEpisode();
    }
}
```

Рисунок 4.30 – Метод OnActionReceived

Метод OnActionReceived оновлює позицію та напрямок повороту агента, також додатково перевіряє, чи треба агенту дати винагороду за близьке місцезнаходження із ціллю.

Далі необхідно зібрати середовище для тестування – окрему сцену, із кілька десятками інстансів роботів та їх цілей для навчання, та запустити це середовище через консольну команду mlagents-learn –run-id=(назва_тесту).

Рисунок 4.31 – Середовище для тестування

Чим нижче стандартне відхилення (Std) та вище середнє значення (mean) – тим краще агенти навчаються досягати бажаного результату. По завершенню навчання отриманий файл перетягуємо в Unity та додаємо як «мізки» агенту.

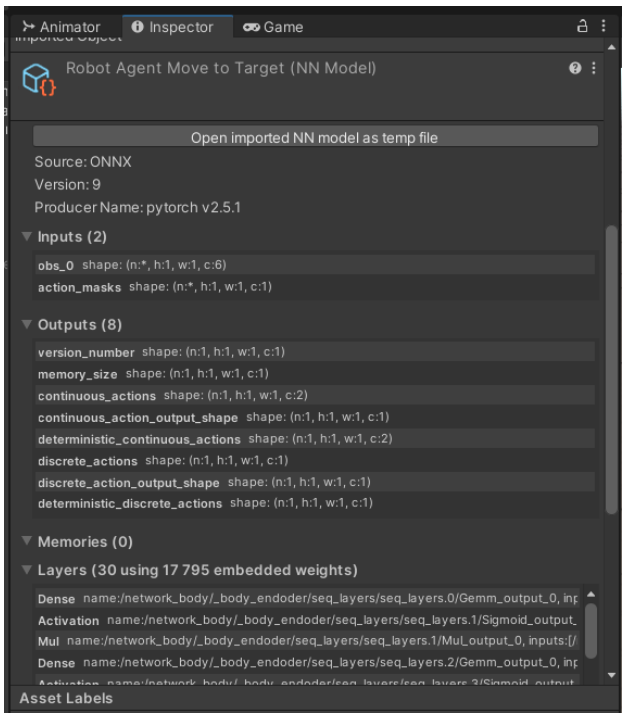


Рисунок 4.32 – Навчена нейромережева модель

4.6 Тестування прототипу

Усі ігрові застосунки вимагають тестування. Нижче показано деякі із основних перевірених тест-кейсів.

Таблиця 4.1 – Завантаження панелі асистеста у головному меню

<table>
<tr><td>Актори</td><td colspan="2">Гравець</td></tr>
<tr><td>Мета</td><td colspan="2">Відкрити застосунок, перейти до головного меню «Ангар» та відкрити панель асистента</td></tr>
<tr><td>Передумова</td><td colspan="2">Користувач повинен мати встановлений застосунок на мобільному пристрої та надти доступ до інтернету</td></tr>
<tr><td colspan="3">Успішний сценарій:
1. Гравець встановив застосунок, надав доступ до інтернету
2. Гравець заходить до гри
3. Гравець натискає кнопку Continue та переходить до головного меню
4. Гравець натискає кнопку Assistant
5. Камера показує робота-асистента
6. Гравець натискає на асистента
7. Відкривається панель асистента</td></tr>
<tr><td colspan="3">Сценарій пройдено успішно, гравець перейшов до панелі асистента</td></tr>
<tr><td colspan="3">Розширення. Успішно виконані</td></tr>
<tr><td>1а</td><td colspan="2">Гравець після заходу в ангар обрав будь-яку іншу панель замість Assistant, але повернувшись назад все одно зміг відкрити цю панель.</td></tr>
<tr><td>2а</td><td colspan="2">Гравець не увімкнув інтернет. Результат: повідомлення по помилку та неможливість відкрити панель асистента.</td></tr>
</table>

Таблиця 4.2 – «Класифікація» запчастини AI-асистентом

Актори	Гравець
Мета	«Класифікувати», тобто описати запчастину, додати її до репозиторію відомих
Передумова	Користувач має відкрити панель AI-асистента, мобільний пристрій має бути під'єднаний до інтернет-мережі.
Успішний сценарій: 1. Гравець зайшов до панелі AI-асистента 2. Гравець бачить власний інвентар із запчастинами 3. Гравець натискає на «невідому» запчастину 4. Гравець бачить кнопку «Класифікувати» та натискає на неї 5. Гравець бачить текстове вікно, де показується запитання (промт) до асистента 6. Асистент «класифікує» запчастину, відповідає у текстовому вікні 7. Запчастина змінює свій вигляд, інформація про неї зберігається в інвентарі	
Сценарій пройдено успішно, гравець «класифікував» предмет	
Розширення. Успішно виконані	
1а	Мобільний пристрій гравця не має інтернет зв'язку. «Класифікація» предмета неможлива, виводиться повідомлення про відсутність зв'язку
2а	Гравець натискає на вже «класифікований» предмет, кнопка відсутня
3а	Пристрій має інтернет-зв'язок, але OpenAI API повертає помилку, користувач отримує повідомлення про проблеми у асистента та прохання звернутись пізніше.

Таблиця 4.3 – Збереження даних застосунком

Актори	Гравець, застосунок
Мета	Перевірка збереження даних (ресурси, стати, описи)
Передумова	Користувач має витратити або отримати ресурси (запчастини), вийти зі гри, повернутись у гру
Успішний сценарій: 1. Гравець зайшов у панель покращення дрона 2. Гравець бачить панель із цінами у запчастинах 3. Гравець покращує будь-які характеристики дрона 4. Гравець отримує покращення (бачить візуально збільшені статистики) 5. Гравець виходить зі гри або згортає застосунок 6. Гравець повертається у гру 7. К-ть ресурсів та статистии лишились такими ж, як і були після покупки	
Сценарій пройдено успішно, гравець покращив дрон	
Розширення. Успішно виконані	
1а	Гравець замість покращення дрона вирушує до панелі ресурсів, де їх можна покращити або комбінувати. Гравець покращує обрану запчастину, витрачені запчастини зникають. Після перезапуску баланс той же.
2а	Гравець використовує механіку «класифікації» запчастини, із певним шансом отримує на ігровий баланс додаткові запчастини. Після перезапуску баланс той же.

Висновки до розділу 4

У четвертому розділі продемонстровано кодування основної ігрової логіки мобільного ігрового застосунку. Описано особливості інтеграції open-source плагіну Unity Open AI та Unity ML. Імплементовано механіки пересування наземного дрона, реалізовано логіку головного меню. Зазначені у попередніх розділах технології використані на практиці для ігрових сцен та механік.

Проведено тестування ігрового застосунку, описано успішні сценарії. Завдяки проведеному аналізу знайдено деякі незначні баги (помилки) в логіці механік та виправлено їх у кодовій базі.

ВИСНОВКИ

Під час написання кваліфікаційної магістерської роботи було отримано цінний досвід з конструювання та проєктування програмного забезпечення, а саме мобільного ігрового застосунку із застосуванням алгоритмів штучного інтелекту. Оформлено 4 розділи, що влючають аналітичну частину, частину з моделювання ПЗ (DFD, IDEF, UML діаграми), частину з проєктування (діаграма класів, проєктування сцен) та частину кодування (інтеграція плагінів, імплементація ігрової логіки).

Для досягнення визначеної мети було виконано поставлені завдання:

– здійснено аналіз предметної області;

– визначено основні функції застосунку, що проєктується;

– створено специфікацію вимог до ПЗ;

– проведено моделювання сценаріїв використання, описано основні та альтернативні use cases;

– створено діаграми DFD 0 і 1 рівня, IDEF0, IDEF1, діаграми діяльності;

– обґрунтовано використання технологій ШІ при створенні мобільних ігрових застосунків;

– створено Unity проєкт з інтегрованими ресурсами;

– імплементовано ігрову логіку відповідно до описаних сценаріїв з використанням алгоритмів ШІ;

– розроблено прототип мобільного ігрового застосунку.

Завдяки проведеному аналізу аналогів було визначено актуальність та доцільність створення мобільного ігрового застосунку з використанням алгоритмів штучного інтелекту. Згідно з технічним завданням було обрано зручні інструменти та технології розробки, а саме: ігровий рушій Unity, мову програмування C# та open-source плагіни для використання ШІ-технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Афонін Ю. С., Давиденко Є.О. Мобільний застосунок з використанням алгоритмів штучного інтелекту. *Моделі, методи та засоби програмної інженерії* : тези доп. Всеукр. наук.-пр. конф. «Могилянські читання – 2024» : Технічні науки. Миколаїв, 06–10 лист. 2024 р.: Чорном. нац. ун-т ім. Петра Могили, 2024. С. 24–26.

2. Urmanov M., Alimanova M., Nurkey A. Training unity machine learning agents using reinforcement learning method. *15th International Conference on Electronics, Computer and Computation, ICECCO,* 2019. DOI:10.1109/ICECCO48375.2019.9043194.

3. Adenan A. R., Kassim M., Kamaluddin N. A. The Design of Mobile 3D Augmented Reality on Marker Visual Inspection Mask. *International Journal of Interactive Mobile Technologies*. 2022. Vol. 16, no. 21. P. 97-113. DOI:10.3991/ijim.v16i21.33843.

4. Chaudhry T., Juneja A., Rastogi S. AR Foundation for Augmented Reality in Unity. *International Journal of Advances in Engineering and Management (IJAEM)*. 2021. Vol. 3, no. 1. P. 662-667.

5. Kim, S.L., Suk, H.J., Kang, J.H., Jung, J.M., Laine, T.H., and Westlin, J. Using Unity 3D to facilitate mobile augmented reality game development. *IEEE World Forum on Internet of Things*. 2014, P. 21–26. URL: https://doi.org/10.1109/WF-IoT.2014.6803110 (date of access: 14.09.2024).

6. Inkarbekov M., Monahan R., Pearlmutter B. A. Visualization of AI Systems in Virtual Reality: A Comprehensive Review. *International Journal of Advanced Computer Science and Application*. 2023. Vol. 14. no. 8, P. 33-42. DOI:10.14569/IJACSA.2023.0140805.

7. Cossu S. M. Beginning Game AI with Unity: Programming Artificial Intelligence with C#, 2020. DOI:10.1007/978-1-4842-6355-6.

8. Takoordyal, K. Beginning Unity Android Game Development: From Beginner to Pro. Apress Berkeley, CA, 2020. 255 p.

9. Florian N., Florian A. D., Florian G. et al. Augmented reality in the free fall study. *AIP Conference Proceedings*, American Institute of Physics Inc. 2023. Vol. 2843. DOI:10.1063/5.0150826.

10. Єфімов Д. В. Використання доповненої реальності (AR) в освіті. *Вісник Запорізького національного університету*. 2021. Вип. 2, ч. 1, С. 217-225. DOI:10.26661/2522-4360-2021-1-2-34.

11. Roberts, P. Game AI Uncovered. 1st ed. Boca Raton : CRC Press, 2024, 216 p.

12. Majumder A. Deep Reinforcement Learning in Unity: With Unity ML Toolkit. 1st ed. Apress Berkeley, CA, 2020, 564 p. DOI:10.1007/978-1-4842-6503-1.

13. Singla N. A Comparison of the Effectiveness of Unreal and Unity Engines in Constructing Virtual Environments. *Manufacturing Technologies and Production Systems: Principles and Practices* / ed. by Neeraj Singla. CRC Press, 2023. P. 217-223. DOI:10.1201/9781003367161-21.

14. AI in Gaming. 5 Biggest Innovations. *Engati*. URL: https://www.engati.com/blog/ai-in-gaming (date of access: 17.09.2024).

15. The Combination of Artificial Intelligence and Extended Reality: A Systematic Review. *Frontiers*. URL: https://www.frontiersin.org/articles/10.3389/frvir.2021.721933/full (date of access: 23.09.2024).

16. Singh R., Mehra N., Dhamija A. Natural Selection Simulator using Machine Learning. *Proceedings - 2022 5th International Conference on Computational Intelligence and Communication Technologies, CCICT*. 2022. P. 257-261. DOI:10.1109/CCiCT56684.2022.00055.

17. Forte J. L. B., Fernández J. A. P. Haifu (hybrid artificial intelligence for unity). *18th International Conference on Intelligent Games and Simulation, GAME-ON*. 2017.

18. Nandy A., Biswas M. Neural Networks in Unity: C# Programming for Windows 10. 1st ed. Apress Berkeley, CA, 2018, 158 p. 2018. DOI:10.1007/978-1-4842-3673-4.

19. Smith, M., Maiti, A., Maxwell, A.D., Kist, A.A. Using Unity 3D as the Augmented Reality Framework for Remote Access Laboratories. *Smart Industry & Smart Education. Lecture Notes in Networks and Systems* / ed. by Auer, M., Langmann, R., 2019., Vol 47. P. 581-590. DOI:10.1007/978-3-319-95678-7_64.

20. Top 6 Machine Learning Use Cases in Gaming for 2024. *Codiste*. URL: https://www.codiste.com/machine-learning-in-gaming (date of access: 23.09.2024).

21. OpenAI-Unity. URL: https://medium.com/@achinthabandara1/openai-unity-509fb6173b87 (date of access: 23.09.2024).

22. Leonardo AI. URL: https://leonardo.ai/ (date of access: 24.09.2024).

23. Unity ML-Agents Toolkit. URL: https://github.com/Unity-Technologies/ml-agents (date of access: 24.09.2024)

24. OpenAI Unity Package. URL: https://github.com/srcnalt/OpenAI-Unity (date of access: 24.09.2024).

25. Пчелянський Д. П., Воінова С. А. Штучний інтелект: перспективи та тенденції розвитку. *Automation of technological and business processes.* 2019. DOI:10.15673/atbp.v11i3.1500.

26. Smith, M., Maiti, A., Maxwell, A.D., Kist, A.A. Using Unity 3D as the Augmented Reality Framework for Remote Access Laboratories. *Smart Industry & Smart Education. Lecture Notes in Networks and Systems* / ed. by Auer, M., Langmann, R., 2019., Vol 47. P. 581-590. DOI:10.1007/978-3-319-95678-7_64.

27. Unity Asset Store. URL: https://assetstore.unity.com/search# (date of access: 25.10.2024).

28. Open AI platform. URL: https://platform.openai.com/docs/overview (date of access: 25.10.2024).

ДОДАТОК А

Діаграма сценаріїв використання

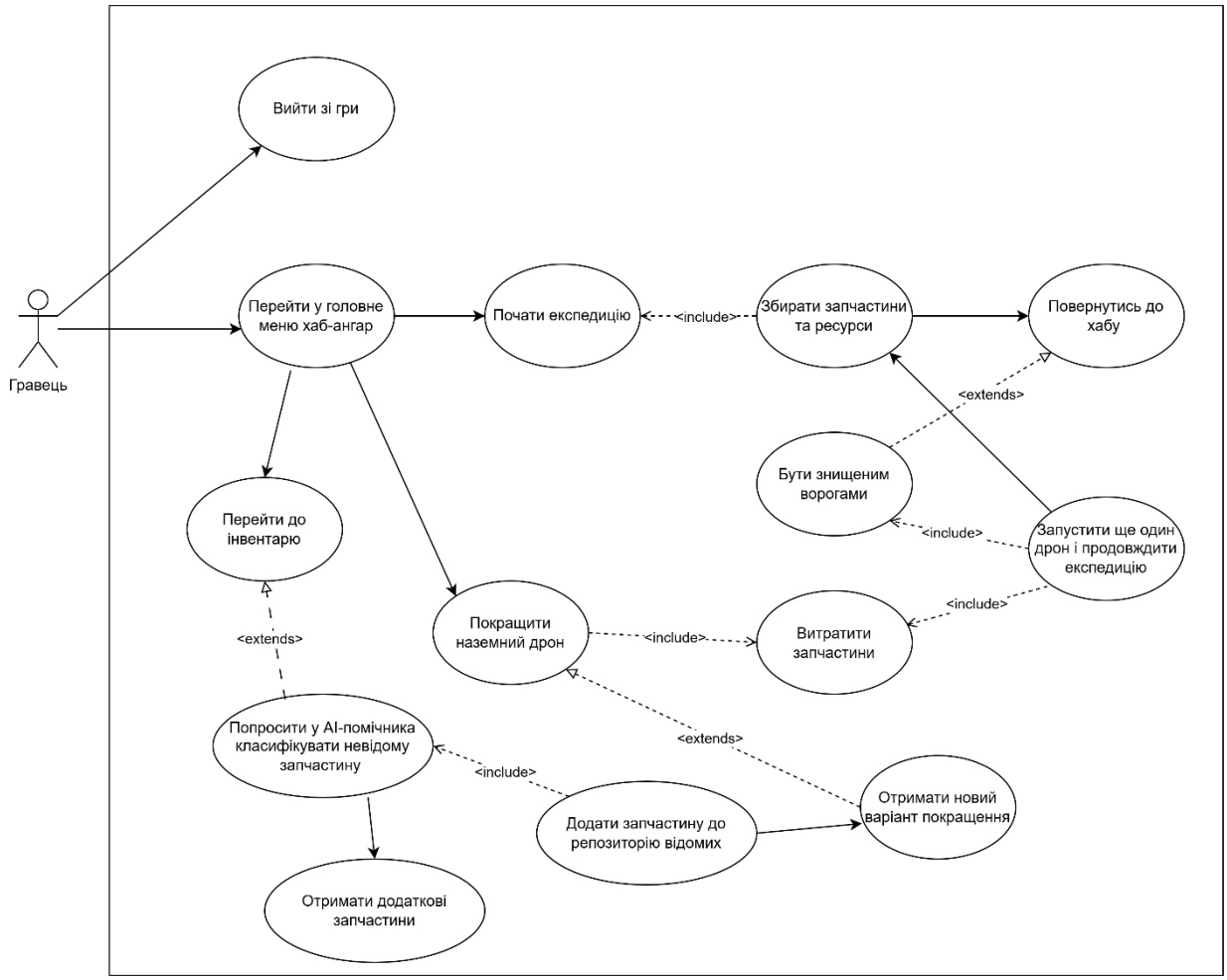


Рисунок А.1 – Use-case діаграма

ДОДАТОК Б

Діаграми класів

Діаграма класів 1

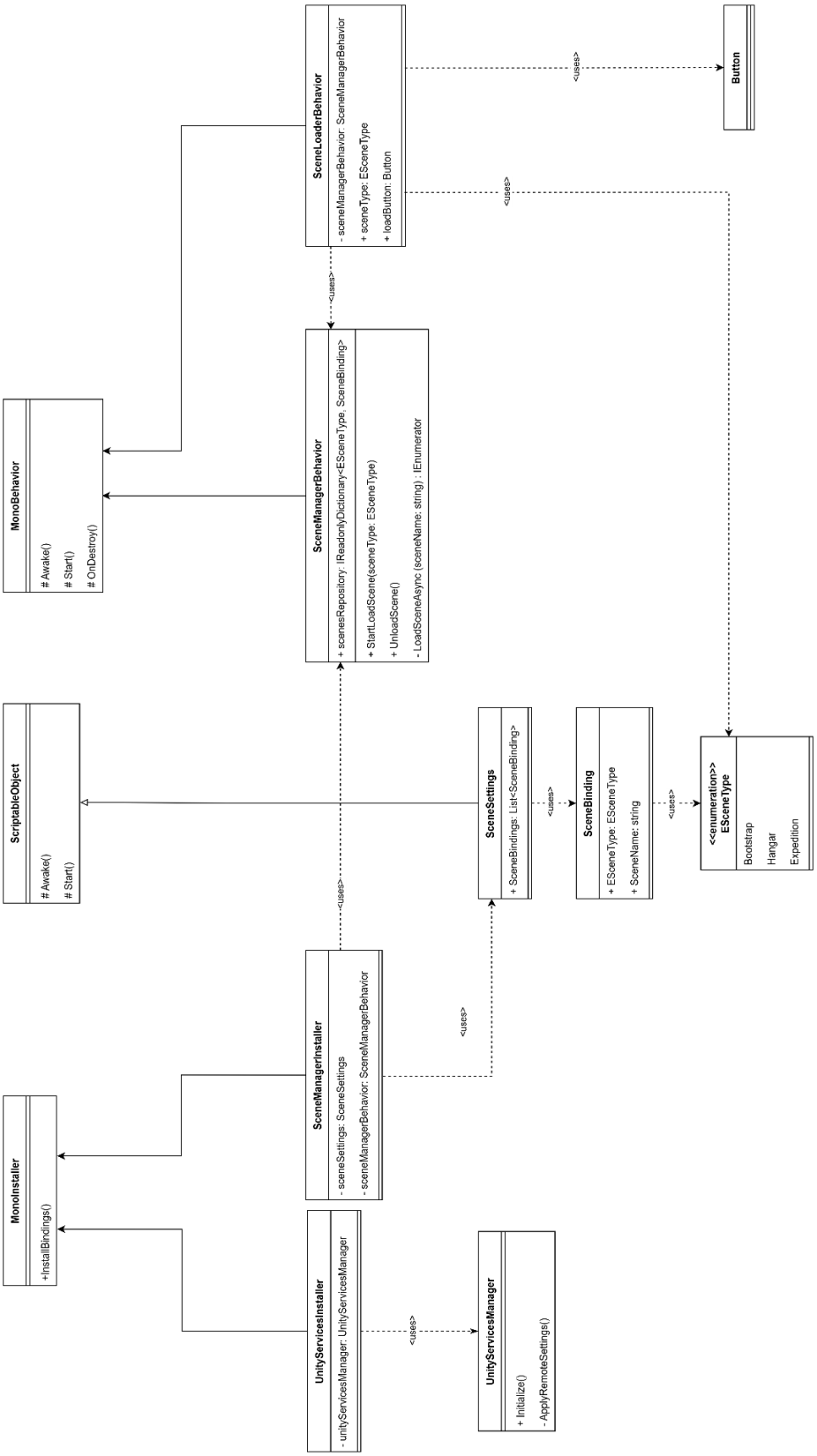


Рисунок Б.1 – Діаграма класів 1

Діаграма класів 2

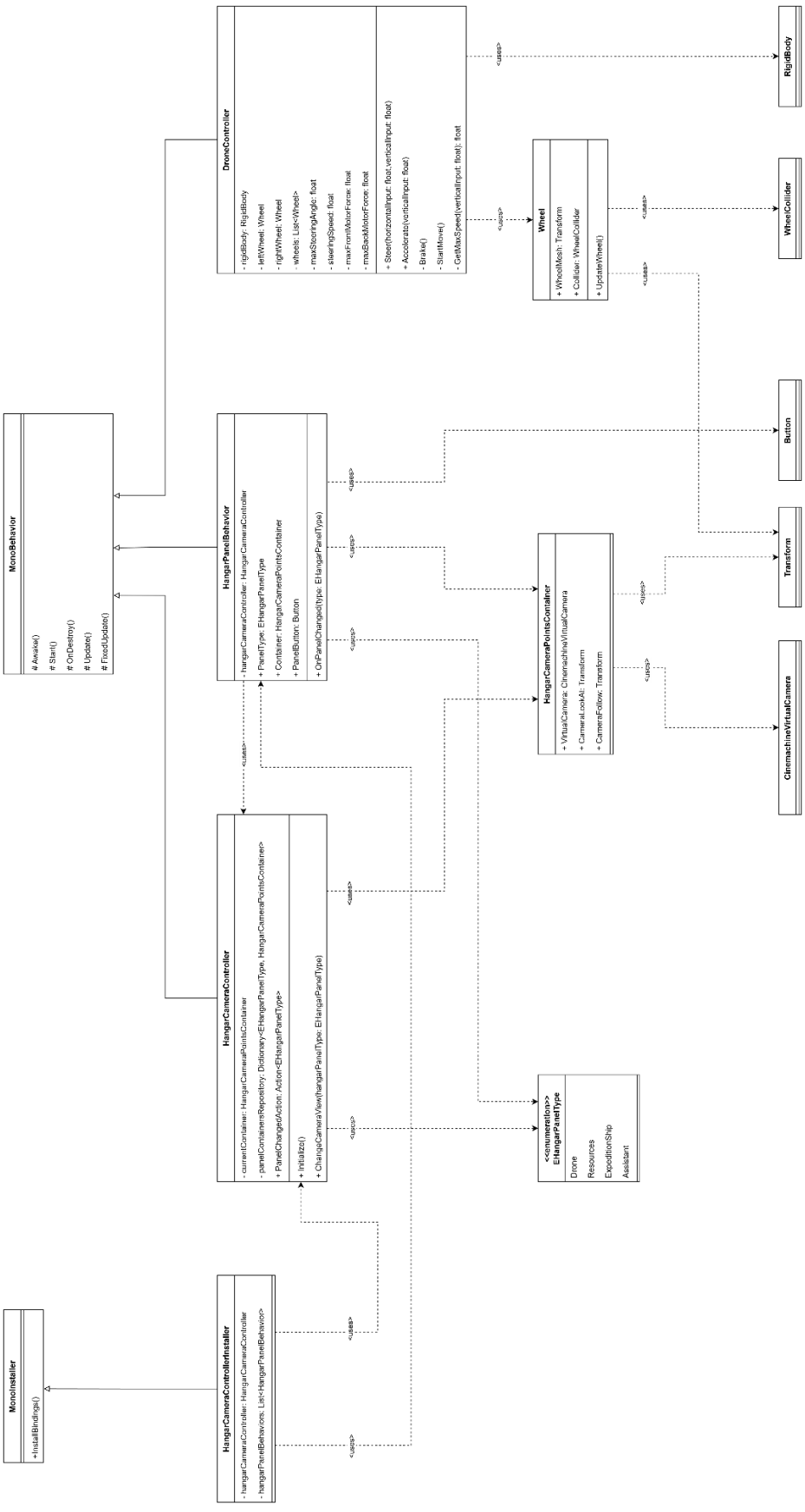


Рисунок Б.2 – Діаграма класів 2

ДОДАТОК В

Інтерфейс застосунку

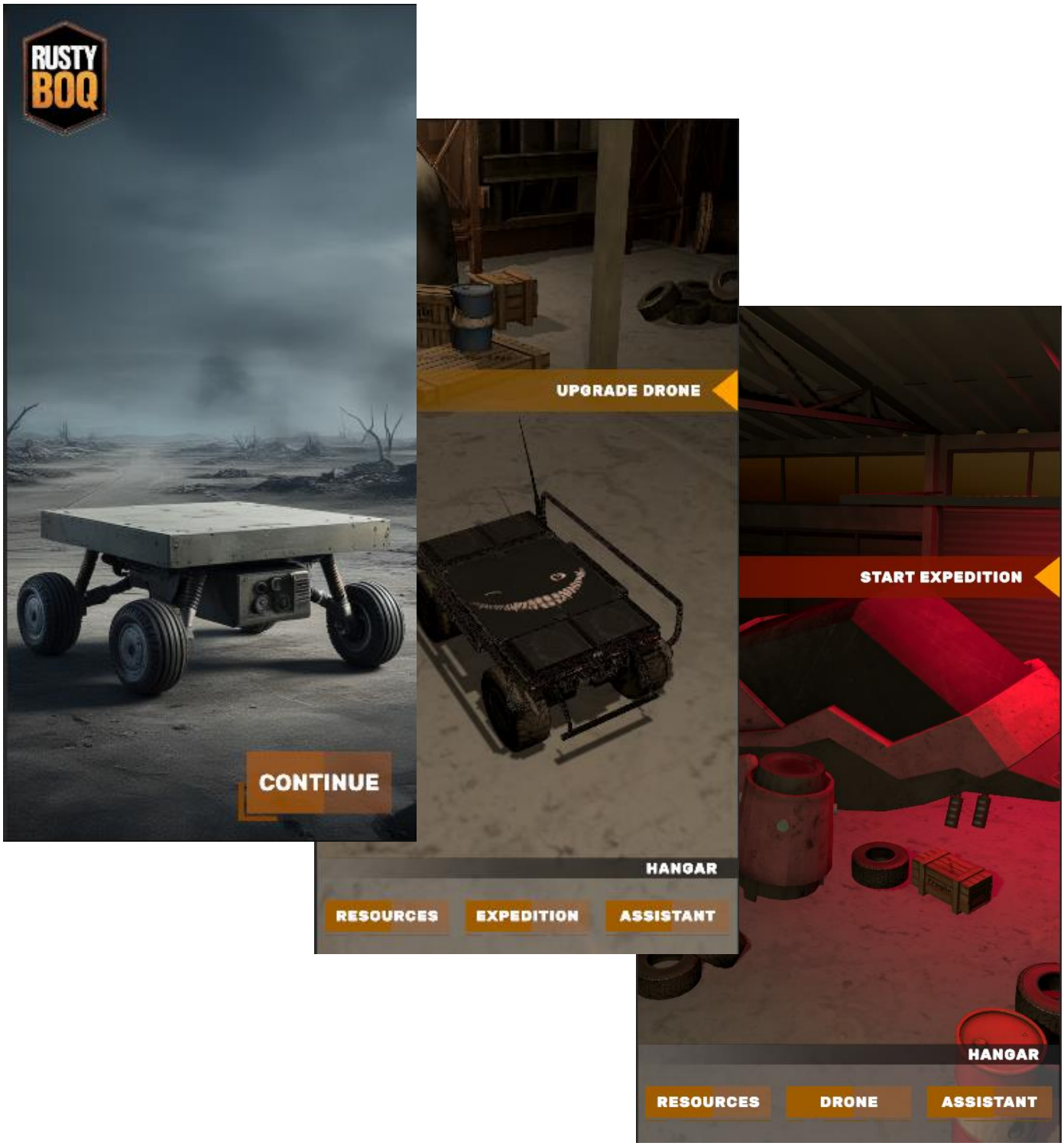


Рисунок В.1 – Зовнішній вигляд застосунку

Інтерфейс застосунку

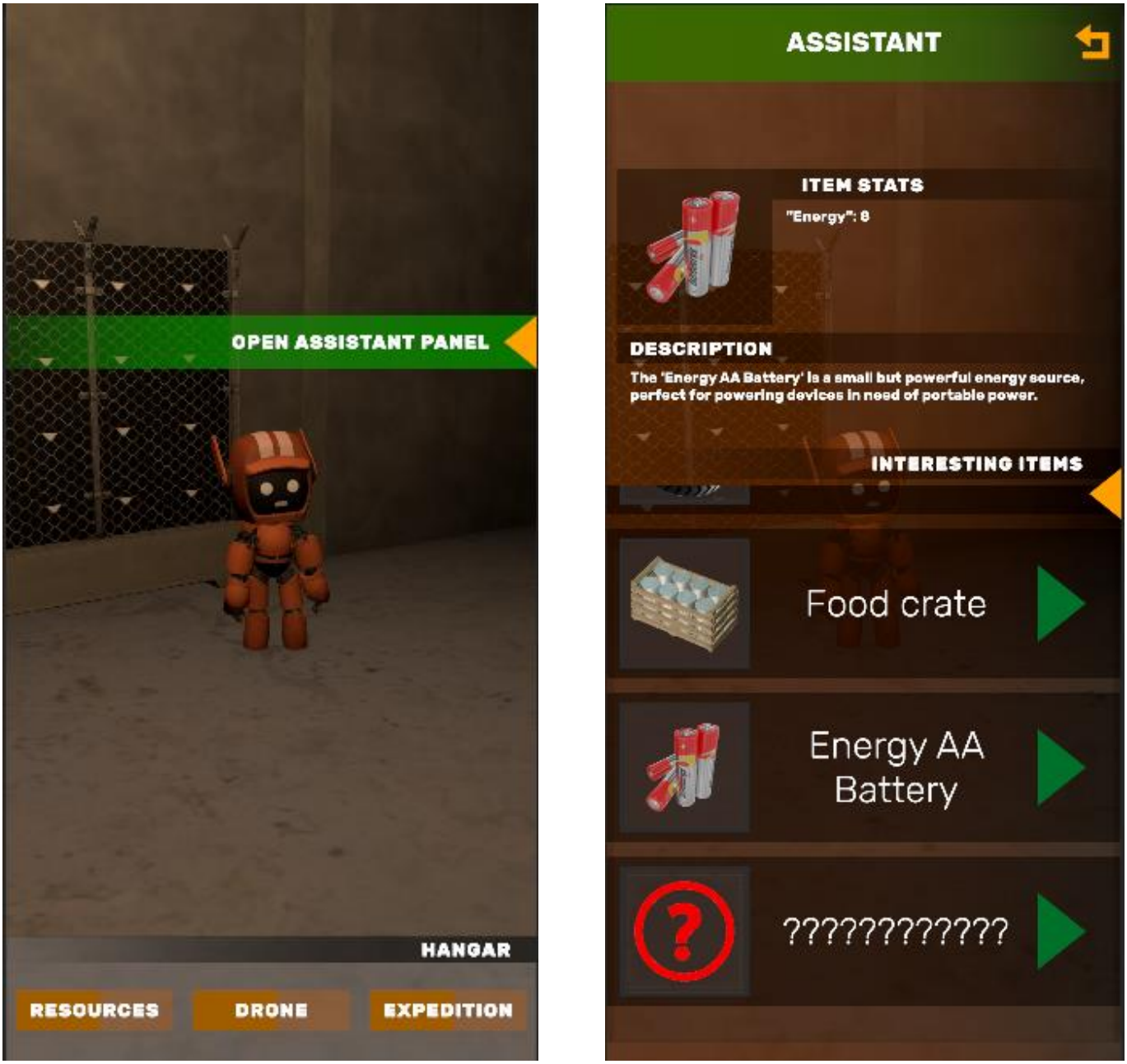
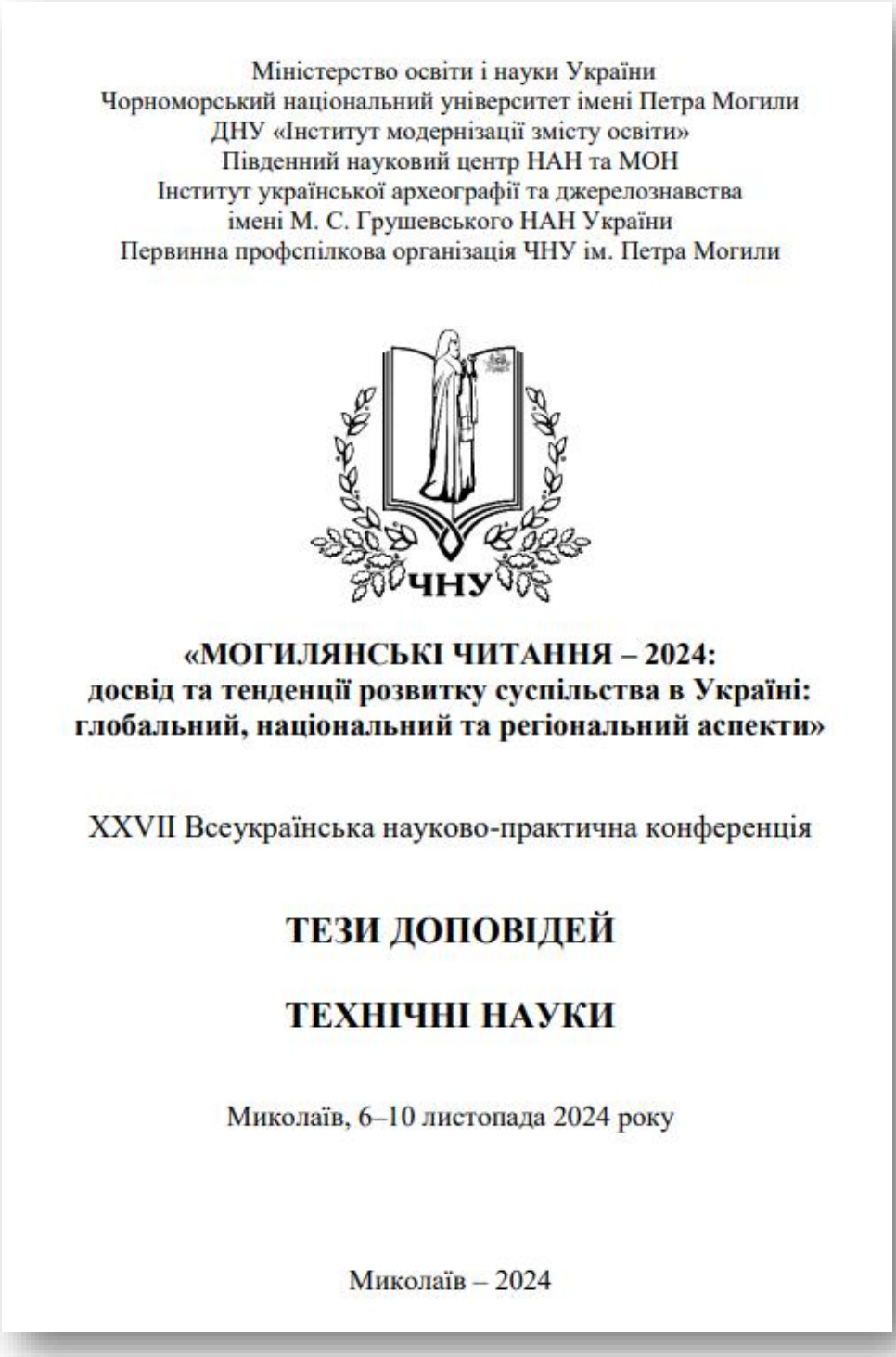


Рисунок В.2 – Вигляд інтерфейсу застосунку

ДОДАТОК Г

Апробація результатів



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 Г.1 – Обкладинка збірника тез доповідей конференції

ЗМІСТ

СЕКЦІЯ: КОМП'ЮТЕРНІ НАУКИ

Підсекція:
- **АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ**

Підсекції:
- **МОДЕЛІ, МЕТОДИ ТА ЗАСОБИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ**

269

Рисунок Г.2 – Сторінка змісту із тезами