Пояснювальна записка
до курсової роботи

на тему: колекційна карткова гра зі штучним інтелектом

КПІ.ІП-1116.045480.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 4
ВСТУП 5
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7
1.1 Загальні положення 7
1.2 Змістовний опис і аналіз предметної області 9
1.3 Аналіз існуючих технологій та успішних ІТ-проєктів 10
1.3.1 Аналіз відомих алгоритмічних та технічних рішень 10
1.3.2 Аналіз допоміжних програмних засобів та засобів розробки 11
1.3.3 Аналіз відомих програмних продуктів 13
1.4 Аналіз вимог до програмного забезпечення 14
1.4.1 Розроблення функціональних вимог 22
1.4.2 Розроблення нефункціональних вимог 28
1.5 Постановка задачі 30
Висновки до розділу 31
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 33
2.1 Моделювання та аналіз програмного забезпечення 33
2.1.1 Опис процесу відображення карт колоди користувача в меню налаштування колод: 33
2.1.2 Опис процесу гри: 34
2.2 Архітектура програмного забезпечення 35
2.3 Конструювання програмного забезпечення 41
2.3.1 Опис алгоритму пошуку оптимального ходу ШІ 41
2.3.2 Зберігання даних 43
2.3.3 Взаємодія користувача з програмою 45
2.3.4 Використання класу MonoBehaviour 48
2.3.5 Опис додаткових утиліт, що використовуються в розробці 49
Висновки до розділу 49
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 51

3.1 Аналіз якості ПЗ 51
3.2 Опис процесів тестування 52
Висновки до розділу 59
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ . 60
4.1 Розгортання програмного забезпечення 60
4.2 Підтримка програмного забезпечення 64
Висновки до розділу 65
ВИСНОВКИ 67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ККІ	–	Колекційні карткові ігри
ККГ	–	Колекційна карткова гра
ДПМК	–	Дерево пошуку Монте-Карло
ШІ	–	Штучний інтелект
CSP	–	Constraint Satisfaction Problem
ПЗ	–	Програмне забезпечення
UI	–	User Interface

ВСТУП

Карткові ігри, беручи свій початок у глибині століть, завжди були популярними серед різних верств населення. В сучасному світі, з розвитком технологій, карткові ігри не лише не втратили своєї значимості, а й отримали можливість ще більше розкрити свій розважальний потенціал. В наш час існують безліч варіацій карткових ігор, серед яких кожен може знайти розвагу собі до душі. Особливий інтерес викликають колекційні карткові ігри, які поєднують в собі елементи стратегії, креативності, колекціонування та змагальності. Найчастіше карткові ігри призначені для гри у групі, це може становити собою велику проблему для людей, що не можуть знайти собі партнера для сумісної гри. Впровадження штучного інтелекту (ШІ), що імітує гравця, у такі ігри відкриває нові горизонти для гри наодинці.

Розвиток карткових ігор пройшов значний шлях від традиційних фізичних колод до сучасних цифрових форматів. Якщо раніше карткові ігри асоціювалися переважно з фізичними колодами, які гравці могли торкатися, мішати та розкладати на столі, то сьогодні все більша кількість цих ігор переходить у цифровий простір. Цей перехід відбувається на тлі загальної цифровізації розваг і відкриває нові можливості для інтерактивності, доступності та дозволяє розробникам частіше випускати оновлення для своїх ігор. Світова тенденція у сфері колекційних карткових ігор (ККІ) відзначається активною участю великих ігрових компаній, які створюють та підтримують свої ККІ. Це спостерігається у діяльності таких відомих компаній як Blizzard Entertainment з їхньою відомою грою "Hearthstone", Wizards of the Coast з легендарною "Magic: The Gathering", а також інших гравців на ринку, які розширюють уявлення про можливості та горизонти ККІ. Ці компанії, використовуючи свої ресурси та досвід, вносять значний вклад у розвиток жанру, постійно оновлюючи геймплей, вводячи нові механіки та елементи, що забезпечує зростання популярності ККІ серед широкої аудиторії гравців у всьому світі.

Ця курсова робота спрямована на створення колекційної карткової гри. Дослідження алгоритмів ігрових ШІ, створення ШІ, що імітуватиме дії суперника.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Колекційні карткові ігри (ККІ) [1] — різновид настільних ігор, для гри в які використовуються спеціально розроблені набори карток. Ці картки, крім грального, мають також самостійну колекційну цінність: вони продаються й обмінюються серед збирачів, через що власне і виникла назва «Колекційні карткові ігри». Хоча колекційні картки відомі дуже давно, логічні ігри з використанням таких карток з'явилися лише на початку 1990-х років, і дуже швидко набули популярності. Однією з найперших та найуспішніших ККГ є гра «Magic: The Gathering» [2], яка вважається родоначальницею всього цього класу ігор.

ККІ дуже близькі до настільних рольових ігор [3]. Сюжети цих ігор базуються або на реальних спортивних змаганнях (автоперегони, баскетбол, бейсбол, футбол, хокей та інші популярні види спорту), або на уявних світах (це можуть бути будь-які сполучення фентезі, наукової фантастики, містики та інших пригодницьких жанрів). В останньому випадку сюжет зазвичай запозичується з відомих книг, коміксів, фільмів, а може й популярних комп'ютерних ігор (як-от «Pokemon»), та інколи це можуть бути і цілком оригінальні сюжети. Яскравим прикладом є гра «Magic: The Gathering», яка сама стала джерелом численних книжних новелізацій, супутніх товарів і різноманітних переспівів.

Кожна картка ККІ має власну назву, крім того, несе на собі оригінальну ілюстрацію та/або художній текст, а також може мати у грі відповідні спеціальні властивості. Усі ці елементи служать для підтримання атмосфери гри, що дасть гравцю змогу краще вжитися у свою роль, і повинні відповідати сюжету, на якому побудовано гру.

Колекційні карткові ігри — це досить складні стратегічні ігри. У грі беруть участь двоє (інколи і більше) гравців. На відміну від традиційних карткових ігор, таких як бридж, преферанс або покер, де всі гравці грають однією спільною

колодою, кількісний та якісний склад якої обмежений і визначений заздалегідь, в ККГ кожен з гравців має власну колоду карток, склад якої він визначає сам, вибираючи на свій смак з доступного йому набору карток. Тому в грі майже ніколи не побачиш двох однакових колод. Складання колоди — вкрай важливий елемент підготовки до гри, чи не важливіший за сам процес гри. Це дозволяє гравцю практично нескінченно варіювати стратегію гри, використовуючи свої улюблені комбінації карток, способи їх взаємодії, та інші елементи гри.

Кожна система ККІ має основний набір правил, які описують цілі гравців, типи карток, що використовуються у грі, та базові принципи взаємодії цих карток. Кожна картка може мати додатковий текст, що визначає особливі ефекти від цієї картки у грі.

Практично всі ККІ побудовані на використанні якоїсь системи ресурсів. Ресурси можуть самі бути певного типу картками гри, або визначатися іншими засобами (такими як особливі фішки, лічильники якогось резерву ресурсів, відмітки на картках тощо). Використання ресурсів є загальним методом контролювання темпу гри. Ресурси також служать для встановлення балансу відносної сили карток. Потужніші картки або ефекти потребують більше ресурсів для їх розігрування. Також контроль над темпом може здійснюватися потоком карток, що входять та виходять з гри (приміром, щоб ввести в гру сильну картку, треба вивести з неї одну чи дві менш сильні).

Під час гри зазвичай гравці по черзі роблять ходи, розігруючи свої картки та виконуючи пов'язані з грою дії. Хід гравця можна розділити на окремі кроки, на кожному з яких виконуються свої специфічні види ігрових дій. Порядок і назви цих кроків відрізняються у різних системах ККІ, однак типовою є така послідовність:

- Відновлення *(Restore)* — усі картки в грі приводяться у стан готовності до наступного ходу (може бути як на початку, так і наприкінці ходу гравця).
- Підготовка *(Standby)* — виконуються всі ефекти, обумовлені картками, що на цей час знаходяться у грі.
- Взяття карток *(Draw cards)* — беруться картки з колоди, для забезпечення циркуляції карток у руці гравця.

- Розігрування карток *(Play cards)* — картки з руки гравця вводяться у гру, впливаючи на ігрову ситуацію (цей крок може виконуватися двічі, до і після фази конфлікту).
- Конфлікт *(Conflict)* — виконуються основні дії, направлені на досягнення перемоги у грі; найпоширенішим методом таких дій є бій *(combat)*.
- Скидання карток *(Discard cards)* — у більшості ККІ максимальна кількість карток, що можуть знаходитися в руці гравця, обмежена, або необхідно їх оновити для наступного ходу.

1.2 Змістовний опис і аналіз предметної області

Колекційні карткові ігри (ККІ), які починалися з фізичних карток, з часом трансформувались у цифрові формати. Перехід відбувався поступово, з появою інтернету та розвитком комп'ютерних технологій. Цифрові версії ККІ дозволили гравцям з усього світу змагатися між собою онлайн, розширюючи спільноту та роблячи карткові ігри більш доступними. Перші спроби цифровізації ККІ полягали в створенні простих онлайн-версій традиційних карткових ігор, які дозволяли гравцям змагатися через інтернет. Однак, з часом технології розвивалися, і ці ігри отримали більш складні графічні та ігрові можливості, включаючи віртуальні світи, складні стратегії та інтерактивність. Сучасні версії ККІ, такі як "Hearthstone" [4] або онлайн-версії "Magic: The Gathering", вже набагато більше, ніж просто перенесення фізичних карток у цифровий простір. Вони включають в себе елементи соціальної взаємодії, глибоких стратегій, а також можливості для індивідуалізації гри гравцем.

На сьогоднішній день, ККІ використовуються не тільки на персональних комп'ютерах, але й на мобільних пристроях, забезпечуючи гравцям можливість змагатися де і коли вони забажають. Мобільні додатки та онлайн-платформи стали основними середовищами для цифрових ККІ. Вони включають в себе різноманітні інструменти та функції, такі як рейтингові системи, чати для спілкування між гравцями, а також можливість проведення турнірів. Штучний інтелект, що застосовується в цих іграх, не тільки симулює опонентів, але й аналізує стратегії гравців для покращення ігрового досвіду. Це дозволяє створювати більш збалансоване і захоплююче ігрове середовище.

Однією з головних проблем сучасних колекційних карткових ігор є їх збалансованість. У деяких випадках, цифрові версії ККІ можуть бути занадто схильні до випадковості або, навпаки, занадто прогнозованими, що погіршує ігровий досвід. Крім того, високі ціни на цифрові картки створюють фінансовий бар'єр для нових гравців, а також можуть сприяти створенню нерівних умов для гравців, які не можуть дозволити собі витрачати значні суми на гру.

Розробники повинні зосередитися на створенні більш справедливих та збалансованих ігрових умов. Це може включати в себе постійне тестування та налаштування геймплею, залучення спільноти гравців для отримання зворотного зв'язку та використання даних про ігровий процес для ідентифікації та коригування незбалансованості. Також необхідне забезпечення більш прозорих та справедливих систем монетизації. Наприклад, встановлення обмежень на витрати в грі або введення моделей, які не дають непереборної переваги гравцям, які витрачають більше коштів.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

У розробці штучного інтелекту (ШІ) для колекційних карткових ігор, ключовим завданням є створення алгоритму, який ефективно оцінює можливі ходи та стратегії. Серед популярних алгоритмів, що здатні забезпечити роботу ШІ в картковій грі, мною було виділено метод Монте-Карло.

Загалом для реалізації штучного інтелекту в комп’ютерних іграх використовується дерево пошуку Монте-Карло. У інформатиці дерево пошуку Монте-Карло (ДПМК) [5] – це евристичний алгоритм пошуку, що широко використовується для процесів прийняття рішень в контексті покрокових ігор з кінцевою кількістю ходів. Основна ідея ДПМК полягає в аналізі найбільш перспективних ходів, розширенні дерева пошуку на основі випадкової вибірки даних простору пошуку. Застосування дерева пошуку Монте-Карло в іграх базується на багатьох симуляціях, які також називаються розгортаннями. У кожній симуляції гра розгортається до самого кінця шляхом вибору ходу

випадковим чином. Кінцевий результат гри кожного розігрування потім використовується для побудови ваги вузлів у дереві гри, щоб обирати вузли, які призведуть до кращих результатів у майбутніх розігруваннях.

На основі проведеного аналізу було вирішено використати спрощену версію методу Монте-Карло для розробки штучного інтелекту. Цей підхід передбачає симуляцію гри після кожного потенційного ігрового рішення велику кількість разів, що дозволяє оцінити ефективність різних ігрових рішень у поточному ігровому положенні. Рішення для ШІ буде обиратися на основі найбільшої кількості виграшних завершень гри у цих симуляціях.

Спрощення методу Монте-Карло дозволяє зменшити обчислювальні витрати, зберігаючи при цьому ефективність аналізу ігрових сценаріїв. Такий підхід забезпечує баланс між точністю прогнозування та швидкістю обчислень, що є критично важливим для динамічних ігрових умов.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Для розробки Колекційної карткової гри, мною було вирішено використати розповсюджений для такого роду ігор ігровий рушій під назвою Unity.

Unity[6] — багатоплатформовий інструмент для розроблення відеоігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX[7] та OpenGL[8].

На Unity написані тисячі ігор, програм, візуалізації математичних моделей, які охоплюють безліч платформ і жанрів. При цьому Unity використовується як великими розробниками, так і незалежними студіями.

Редактор Unity має простий Drag&Drop[9] інтерфейс (Рисунок 1.1), що складається з різних вікон, завдяки чому можна проводити налагодження гри

прямо в редакторі. Двигун використовує для написання скриптів мову програмування C#[10].

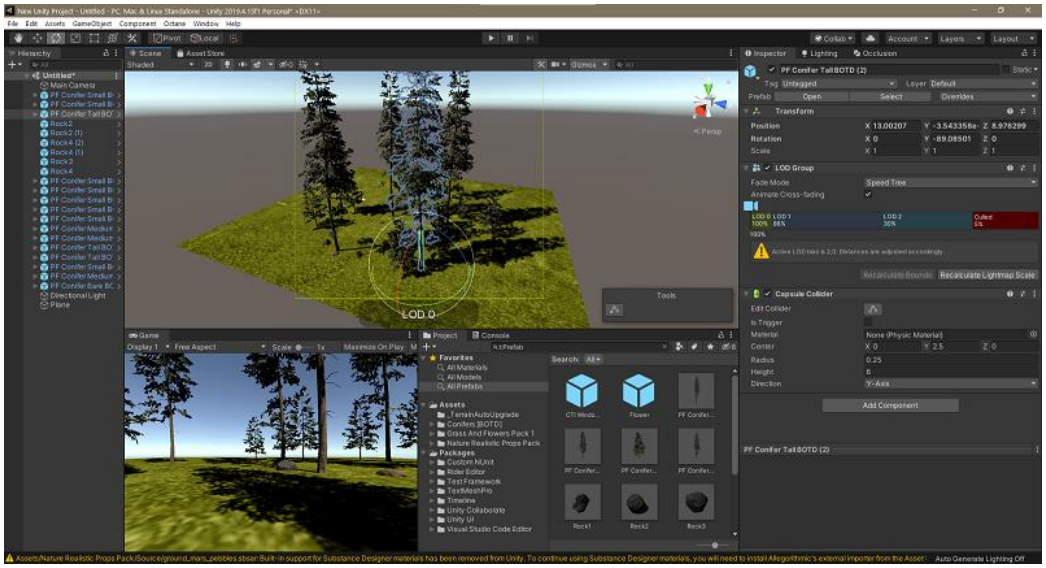

Рисунок 1.1 – Інтерфейс Unity

Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв та налаштувань. Сцени можуть містити як об'єкти (моделі). Об'єкти, у свою чергу, містять набори компонентів, з якими і взаємодіють скрипти. У будь-якого об'єкта на сцені обов'язково присутній компонент Transform — він зберігає координати розташування, повороту і розмірів об'єкта по всіх трьох осях. У об'єктів з видимою геометрією також за умовчанням присутній компонент Mesh Renderer, що робить модель видимою. Різні моделі можуть об'єднуватися в набори (ассети) для швидкого доступу до них. У редакторі є система наслідування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту та масштабу батьківського об'єкта. Скрипти у редакторі прикріплюються до об'єктів як окремі компоненти.

На Unity написані сотні ігор, додатків та симуляцій, Unity використовується як великими розробниками (наприклад, Blizzard[8]), так і при створенні інді-ігор. Комп'ютерні ігри на Unity охоплюють безліч платформ та жанрів. Характерними прикладами колекційних карткових ігор, написаних з допомогою Unity є:

- Hearthstone – колекційна карткова онлайн-гра створена за мотивами всесвіту Warcraft, розроблена компанією Blizzard Entertainment

- Inscryption [11] – колекційна карткова гра з елементами roguelike, розроблена компанією Daniel Mullins Games і видана Devolver Digital.
- Gwent: The Witcher Card Game [12] – колекційна карткова гра, створена за мотивами всесвіту The Witcher, розроблена компанією CD Projekt RED.

Отже, вибір Unity для розробки колекційної карткової гри обумовлений його гнучкістю та здатністю адаптуватися до різних платформ та умов розробки. Його зручний редактор з Drag&Drop інтерфейсом, потужні можливості для 2D графіки, а також широкі можливості для написання скриптів на C# роблять Unity ідеальним інструментом для створення як великих, так і невеликих проектів. Він також дозволяє легко інтегрувати різноманітні асети та ресурси, що є важливим для ігор з великою кількістю елементів та ілюстрацій.

1.3.3 Аналіз відомих програмних продуктів

Для проведення порівняльного аналізу існуючих аналогів розроблюваного ігрового додатку було обрано такі популярні карткові ігри, як «Hearthstone», «Gwent: The Witcher Card Game» та «Magic: The Gathering Arena». Вибір зазначених аналогів заснований на їх популярності, інноваційності та значущості у сфері ККІ.

Перед порівнянням варто зазначити, що усі наведені аналоги є комерційними проектами із закритим кодом, що робить їх недоступними для загального огляду та аналізу з боку звичайних користувачів або розробників. Деталі їх розробки, внутрішні алгоритми та код залишаються конфіденційними.

Для порівняння аналогів розроблюваного ПЗ можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння аналогів

Критерії порівняння	Hearthstone	Gwent: The Witcher Card Game	Magic: The Gathering Arena
Видавець	Blizzard Entertainment	CD Projekt	Wizards of the Coast

Розробник	Blizzard Entertainment	CD Projekt RED	Wizards Digital Games Studio

Продовження таблиці 1.1

Підтримувані платформи	Microsoft Windows, macOS, iOS, Android	Microsoft Windows, macOS, iOS, Android, PlayStation, Xbox	Microsoft Windows, macOS, iOS, Android
Наявність монетизації	Монетизація наявна	Монетизація наявна	Монетизація наявна
Візуальний стиль	Карикатурний стиль, яскраві кольори	Реалістичний та деталізований стиль, темні тони	Різноманітність стилів, висока деталізація
Наявність мультиплеєру	Мультиплеєр наявний	Мультиплеєр наявний	Мультиплеєр наявний
Наявність сингліплеєру	Сингліплеєр наявний	Сингліплеєр наявний	Сингліплеєр наявний
Підтримка	Регулярні оновлення	Регулярні оновлення	Регулярні оновлення

1.4 Аналіз вимог до програмного забезпечення

Діаграму варіантів використання розробки можна побачити на рисунку 1.2.

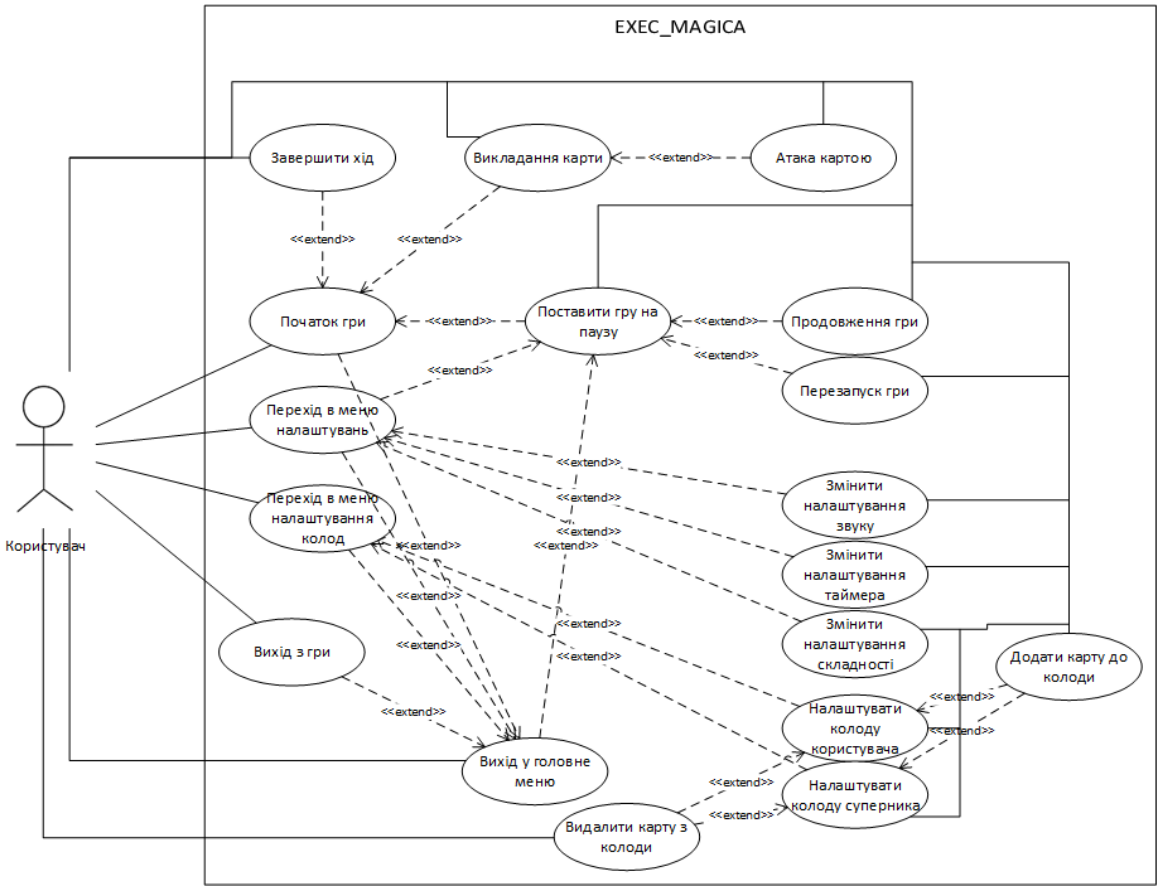

Рисунок 1.2 – Діаграма варіантів використання

В таблицях 1.2 - 1.19 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Початок гри
Use case ID	UC-01
Goals	Розпочати гру
Actors	Користувач
Trigger	Користувач бажає розпочати гру.
Pre-conditions	Відкрите головне меню.
Flow of Events	Користувач натискає на кнопку «Start game». Завантажується сцена з геймплеєм. Розпочинається ігровий процес.
Extension	-
Post-Condition	Завантажується сцена з геймплеєм. Розпочинається ігровий процес.

Таблиця 1.3 - Варіант використання UC-2

Use case name	Перехід в меню налаштування колод

Use case ID	UC-02

Продовження таблиці 1.3

Goals	Перейти в меню налаштування колод
Actors	Користувач
Trigger	Користувач бажає змінити свою колоду або колоду суперника.
Pre-conditions	Відкрите головне меню.
Flow of Events	Користувач натискає на кнопку «Change deck». Завантажується сцена з меню зміни колод. З’являється меню вибору колоди.
Extension	-
Post-Condition	Завантажується сцена з меню зміни колод. З’являється меню вибору колоди.

Таблиця 1.4 - Варіант використання UC-3

Use case name	Перехід в меню налаштувань
Use case ID	UC-03
Goals	Перейти в меню налаштувань
Actors	Користувач
Trigger	Користувач змінити налаштування гри.
Pre-conditions	Відкрите головне меню.
Flow of Events	Користувач натискає на кнопку «Settings». Відкривається меню налаштувань.
Extension	-
Post-Condition	Відкривається меню налаштувань.

Таблиця 1.5 - Варіант використання UC-4

Use case name	Вихід з гри
Use case ID	UC-04
Goals	Вийти з гри
Actors	Користувач
Trigger	Користувач хоче вийти з гри.
Pre-conditions	Відкрите головне меню.
Flow of Events	Користувач натискає на кнопку «Quit». Гра закривається.
Extension	-
Post-Condition	Гра закривається.

Таблиця 1.6 - Варіант використання UC-5

Use case name	Викладання карти
Use case ID	UC-05

Продовження таблиці 1.6

Goals	Викласти карту
Actors	Користувач
Trigger	Користувач хоче викласти карту.
Pre-conditions	Гру розпочато. Корисистувач має достатню кількість мани на викладання обраної карти.
Flow of Events	Користувач перетягує обрану карту на своє ігрове поле за допомогою курсора. Обрана карта переміщається на ігрове поле користувача.
Extension	Якщо користувача бракує мани на викладання обраної карти, перетягування карти буде недоступне.
Post-Condition	Обрана карта переміщається на ігрове поле користувача.

Таблиця 1.7 - Варіант використання UC-6

Use case name	Атака картою
Use case ID	UC-06
Goals	Атакувати обрану ціль викладеною на поле картою
Actors	Користувач
Trigger	Користувач хоче атакувати певну ціль викладеною на поле картою.
Pre-conditions	Гру розпочато. Карта, що має атакувати викладена на ігрове поле. Карта викладена на поле не під час поточного ходу. Обрана ціль доступна для атаки.
Flow of Events	Користувач обирає карту для атаки серед доступних карт, що викладено на поле. Користувач перетягує карту на обрану ціль. Між картою та ціллю відбувається «бій», показники здоров'я карти та цілі змінюються в залежності від результатів «бою».
Extension	Під час ходу доступні для атаки карти підсвічуються. При перетягуванні карти, доступні для атаки цілі підсвічуються.
Post-Condition	Між картою та ціллю відбувається «бій», показники здоров'я карти та цілі змінюються в залежності від результатів «бою».

Таблиця 1.8 - Варіант використання UC-7

Use case name	Завершення ходу
Use case ID	UC-07
Goals	Завершити хід
Actors	Користувач
Trigger	Користувач хоче завершити хід.

Продовження таблиці 1.8

Pre-conditions	Гру розпочато. Наразі хід користувача.
Flow of Events	Користувач натискає на кнопку “END TURN”. Користувачеві забороняється взаємодія зі своїми картами. Розпочинається хід суперника.
Extension	Якщо користувач не натискає кнопку “END TURN”, хід користувача автоматично завершиться по закінченню часу, якщо ввімкнено таймер у меню налаштувань.
Post-Condition	Розпочинається хід суперника.

Таблиця 1.9 - Варіант використання UC-8

Use case name	Поставити гру на паузу
Use case ID	UC-08
Goals	Поставити гру на паузу
Actors	Користувач
Trigger	Користувач хоче поставити гру на паузу.
Pre-conditions	Гру розпочато.
Flow of Events	Користувач натискає на кнопку “PAUSE”. Усі ігрові процеси призупиняються. Відкривається меню паузи.
Extension	-
Post-Condition	Відкрите меню паузи.

Таблиця 1.10 - Варіант використання UC-9

Use case name	Продовження гри
Use case ID	UC-09
Goals	Продовжити гру
Actors	Користувач
Trigger	Користувач хоче продовжити гру під час паузи.
Pre-conditions	Гру розпочато. Відкрите меню паузи.

Flow of Events	Користувач натискає на кнопку «Resume». Усі ігрові процеси продовжуються. Закривається меню паузи. Гру продовжено.
Extension	-
Post-Condition	Гру продовжено.

Таблиця 1.11 - Варіант використання UC-10

Use case name	Перезапуск гри

Продовження таблиці 1.11

Use case ID	UC-10
Goals	Почат игру спочатку
Actors	Користувач
Trigger	Користувач хоче почат игру спочатку.
Pre-conditions	Гру розпочато. Відкрите меню паузи.
Flow of Events	Користувач натискає на кнопку «Restart». Усі ігрові процеси перезапускаються. Закривається меню паузи. Гру розпочато.
Extension	-
Post-Condition	Гру розпочато.

Таблиця 1.12 - Варіант використання UC-11

Use case name	Змінити налаштування звуку
Use case ID	UC-11
Goals	Змінити налаштування звуку
Actors	Користувач
Trigger	Користувач хоче змінити налаштування звуку.
Pre-conditions	Відкрите меню налаштувань.
Flow of Events	Користувач керує повзунком, підписаним написом “Sound”. Рівень гучності змінено.
Extension	В залежності від позиції повзунка змінюється рівень гучності звуку гри.
Post-Condition	Рівень гучності змінено.

Таблиця 1.13 - Варіант використання UC-12

Use case name	Змінити налаштування таймера
Use case ID	UC-12

Goals	Змінити налаштування таймера
Actors	Користувач
Trigger	Користувач хоче змінити налаштування таймера.
Pre-conditions	Відкрите меню налаштувань.
Flow of Events	Користувач обирає серед 4-х режимів роботи таймера. Налаштування таймера змінено.
Extension	Достіпні 4 режими роботи таймера: «off» (таймер вимкнено), «60s», «120s», «180s».
Post-Condition	Налаштування таймера змінено.

Таблиця 1.14 - Варіант використання UC-13

Use case name	Змінити налаштування складності
Use case ID	UC-13
Goals	Змінити налаштування складності
Actors	Користувач
Trigger	Користувач хоче змінити налаштування складності.
Pre-conditions	Відкрите меню налаштувань.
Flow of Events	Користувач обирає серед 3-х режимів складності гри. Налаштування складності змінено.
Extension	Достіпні 3 режими складності гри: «Easy» (Низький), «Normal» (Середній) , «Hard» (Високий).
Post-Condition	Налаштування складності змінено.

Таблиця 1.15 - Варіант використання UC-14

Use case name	Налаштувати колоду користувача
Use case ID	UC-14
Goals	Відкрити меню налаштування колоди користувача.
Actors	Користувач
Trigger	Користувач хоче змінити свою колоду.
Pre-conditions	Відкрите меню налаштування колод. З'явилося меню вибору колоди.
Flow of Events	В меню вибору колоди користувач натискає кнопку «My deck». Відображається колода користувача.
Extension	Карти, що виділено зеленим кольором, входять до обраної колоди, а ті що не виділені – не входять.

Post-Condition	Відображається колода користувача.

Таблиця 1.16 - Варіант використання UC-15

Use case name	Налаштувати колоду суперника
Use case ID	UC-15
Goals	Відкрити меню налаштування колоди суперника.
Actors	Користувач
Trigger	Користувач хоче змінити колоду суперника.
Pre-conditions	Відкрите меню налаштування колод. З'явилося меню вибору колоди.
Flow of Events	В меню вибору колоди користувач натискає кнопку «Enemy deck». Відображається колода суперника.

Продовження таблиці 1.16

Extension	Карти, що виділено зеленим кольором, входять до обраної колоди, а ті що не виділені – не входять.
Post-Condition	Відображається колода суперника.

Таблиця 1.17 - Варіант використання UC-16

Use case name	Додати карту до колоди
Use case ID	UC-16
Goals	Додати обрану карту до колоди
Actors	Користувач
Trigger	Користувач хоче додати обрану карту до колоди.
Pre-conditions	Відкрите меню налаштування колод. Відображається обрана колода.
Flow of Events	Користувач натискає на обрану карту, що не виділено зеленим кольором. Обрана карта додана до обраної колоди.
Extension	Якщо обрана колода вже містить 30 карт, функція додавання нової карти буде недоступна.
Post-Condition	Обрана карта додана до обраної колоди.

Таблиця 1.18 - Варіант використання UC-17

Use case name	Видалити карту з колоди
Use case ID	UC-17

Goals	Видалити обрану карту з колоди
Actors	Користувач
Trigger	Користувач хоче видалити обрану карту з колоди.
Pre-conditions	Відкрите меню налаштування колод. Відображається обрана колода.
Flow of Events	Користувач натискає на обрану карту, що виділено зеленим кольором. Обрана карта видалена з обраної колоди.
Extension	-
Post-Condition	Обрана карта видалена з обраної колоди.

Таблиця 1.19 - Варіант використання UC-18

Use case name	Вихід у головне меню
Use case ID	UC-18

Продовження таблиці 1.19

Goals	Відкрити головне меню.
Actors	Користувач
Trigger	Користувач вийти у головне меню.
Pre-conditions	Гру розпочато. Відкрите меню паузи.
Flow of Events	Користувач натискає на кнопку «Quit». Відкрите головне меню.
Extension	-
Post-Condition	Відкрите головне меню.

1.4.1 Розроблення функціональних вимог

На рисунку 1.3 наведено загальну модель вимог, а в таблиці 1.20 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити таблиці 1.21.

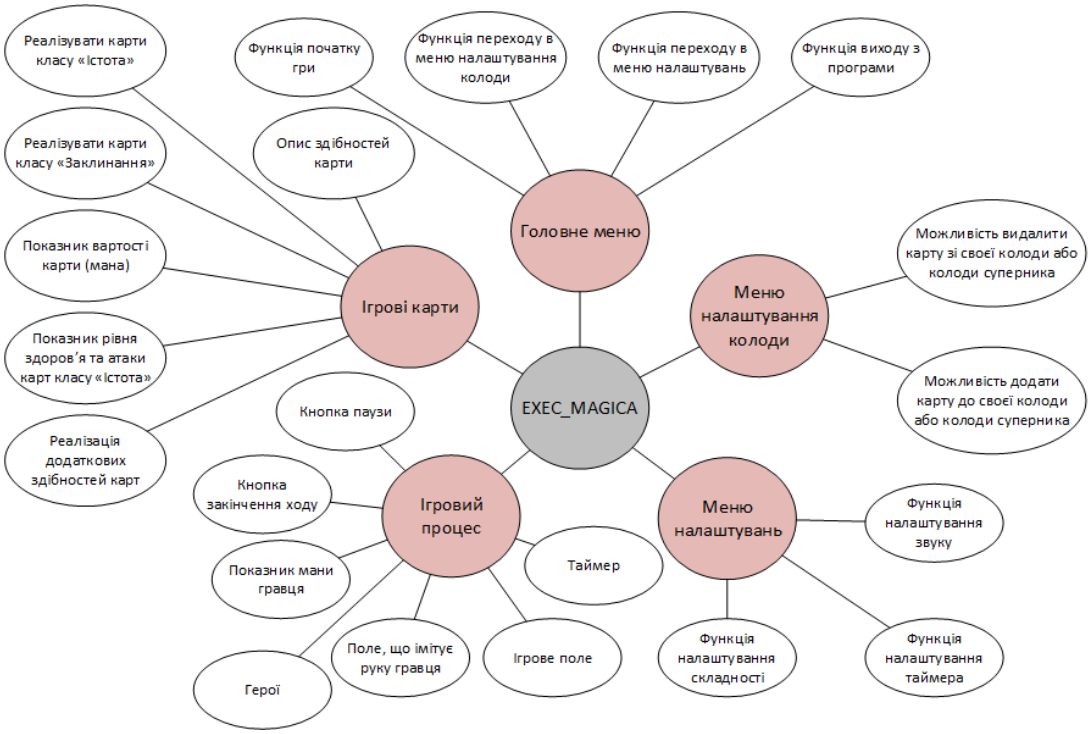

Рисунок 1.3 – Модель вимог у загальному вигляді

Таблиця 1.20 – Функціональні вимоги

Номер	Назва	Опис
FR-1	Функція початку гри	При натисканні на кнопку має розпочинатися ігровий процес

Продовження таблиці 1.20

FR-2	Функція переходу в меню налаштування колоди	При натисканні на кнопку має завантажуватися сцена з меню налаштування колоди, де користувач матиме змогу змінити або свою колоду, або колоду суперника.
FR-3	Функція переходу в меню налаштувань	При натисканні на кнопку має відкриватися меню налаштувань, де користувач матиме змогу змінити налаштування звуку, таймеру та складності гри.

FR-4	Функція виходу з програми	При натисканні на кнопку виконання програми має завершитися.
FR-5	Можливість видалити карту зі своєї колоди або колоди суперника.	При натисканні на карту в меню налаштування колоди, якщо карта присутня у обраній колоді, вона має видалитися із колоди.
FR-6	Можливість додати карту до своєї колоди або колоди суперника.	При натисканні на карту в меню налаштування колоди, якщо карта відсутня у обраній колоді, вона має додатися до колоди.
FR-7	Функція налаштування звуку	Користувач повинен мати можливість налаштовувати рівень гучності звукових ефектів за допомогою спеціального повзунка.
FR-8	Функція налаштування таймера	Користувач повинен мати можливість налаштовувати ігровий таймер: змінити час, що дається на хід, або вимкнути таймер.

Продовження таблиці 1.20

FR-9	Функція налаштування складності	Користувач повинен мати можливість налаштовувати рівень складності гри: легкий, середній або високий.
FR-10	Таймер	Під час гри, на ігровому полі має бути присутній таймер, що демонструватиме час, що залишився до кінця поточного ходу.
FR-11	Ігрове поле	Під час гри, у кожного з гравців має бути своє ігрове поле, на яке вони

		викладатимуть свої ігрові карти з рук. Картою можна атакувати тількі якщо вона викладена на ігрове поле.
FR-12	Поле, що імітує руку гравця	Під час гри, у кожного з гравців має бути своє поле, що імітує руку гравця. В цьому полі мають з'являтися карти при їх видачі. Карту можна викласти на ігрове поле тільки якщо вона знаходиться у руці гравця.
FR-13	Герої	Під час гри, у кожного з гравців має бути свій герой. Герой є умовним «аватаром» гравця під час ігрового процесу зі своїм показником здоров'я. Гра вважається завершеною тільки у тому випадку, якщо рівень здоров’я одного з героїв впаде до рівня нуля. Той гравець, чий герой «помер» першим, програє

Продовження таблиці 1.20

FR-14	Показник мани гравця	На початку ходу, гравцю надається певна кількість мани. Мана витрачається на розігрування певної карти. На ігровому полі мають бути присутні показники поточної кількості мани кожного з гравців.
FR-15	Кнопка закінчення ходу	Під час свого ходу у користувача має бути можливість закінчення ходу при

		натисканні на кнопку. Після натискання має розпочинатися хід супротивника.
FR-16	Кнопка паузи	Під час гри, у користувача має бути можливість зупинити ігровий процес натиснувши кнопку паузи. При натисканні має з'явитися меню паузи.
FR-17	Реалізація додаткових здібностей карт	У деяких карт мають бути додаткові здібності, що певним чином впливатимуть на інші карти, героїв або ігровий процес вцілому.
FR-18	Реалізувати карти класу «Істота»	Карти класу «Істота» - карти що мають свої показники здоров'я, атаки та іноді мають свої додаткові здібності. Вони можуть бути викладені на ігрове поле, можуть атакувати героїв та ворожі карти, можуть бути атакованими ворожими картами.

Продовження таблиці 1.20

FR-19	Реалізувати карти класу «Заклинання»	Карти класу «Заклинання» - карти що після розігрування певним чином впливають на інші карти, героїв або ігровий процес вцілому. Вони не можуть бути викладені на ігрове поле, не мають своїх показників здоров'я та

		атаки. Розігруються напряму із руки гравця.
FR-20	Показник вартості карти (мана)	У кожної карти має бути свій показник вартості, що відображає кількість мани, що потребується витратити гравцю на розігрування цієї карти.
FR-21	Показники рівня здоров'я та атаки карти класу «Істота»	У кожної карти класу «Істота» мають бути свої показники рівня здоров'я та атаки. Показник атаки карти відобража шкоду, яку карта завдасть при атакуванні ворожої карти або героя. Якщо показник здоров'я карти знижується до нуля, карта «помирає».
FR-22	Опис здібностей карти	На кожній карті має бути присутнє текстове поле, на якому описуються здібності карти, якщо вони присутні. Якщо ж вони відсутні, текстове поле залишається порожнім.

Таблиця 1.21 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15	UC-16	UC-17	UC-18
FR-1	+				+	+	+	+	+	+								+
FR-2		+												+	+	+	+	
FR-3			+								+	+	+					

FR-4				+														
FR-5		+												+	+		+	
FR-6		+												+	+	+		
FR-7			+								+							
FR-8			+									+						
FR-9			+										+					
FR-10	+		+															
FR-11	+				+	+												
FR-12	+				+	+												
FR-13	+					+												
FR-14	+				+													
FR-15	+						+											
FR-16	+							+	+	+								+
FR-17	+																	
FR-18	+	+			+	+												
FR-19	+	+																
FR-20	+	+																
FR-21	+	+																
FR-22	+	+																

1.4.2 Розроблення нефункціональних вимог

1.4.2.1 Вимоги до надійності

Забезпечення стабільної роботи програми при заданих системних вимогах.

1.4.2.2 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

1.4.2.2.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються

1.4.2.2.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

1.4.2.3 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Pentium® D или AMD® Athlon™ 64 X2;
- об‘єм ОЗП: 3 Гб;
- відеокарта: NVIDIA® GeForce® 8600 GT або ATI™ Radeon™ HD 2600XT, або краще

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel® Core™ 2 Duo (2,2 ГГц) или AMD® Athlon™ 64 X2 (2,6 ГГц);
- об‘єм ОЗП: 4 Гб;
- відеокарта: NVIDIA® GeForce® 240 GT або ATI™ Radeon™ HD 4850, або краще

1.4.2.4 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows 10 та 11.

1.4.2.4.1 Вимоги до вхідних даних

Вимоги до вхідних даних не висуваються.

1.4.2.4.2 Вимоги до вихідних даних

Вимоги до вихідних даних не висуваються.

1.4.2.4.3 Вимоги до мови розробки

Розробку виконати на мові програмування C#.

1.4.2.4.4 Вимоги до середовища розробки

Розробку виконати на платформі Unity.

1.4.2.4.5 Вимоги представлення вихідних кодів

Вихідний код програми має бути представлений у вигляді усіх класів програми.

1.4.2.5 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

1.4.2.6 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

1.4.2.7 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

1.5 Постановка задачі

Задачею курсової роботи є розробка Колекційної карткової гри у цифровому представленні. Мають бути реалізовані основні геймплейні механіки гри, що наведені вище, включаючи управління картковими колодами користувача та його суперника, використання створених колод у грі. У гру має бути імплементований штучний інтелект, що імітуватиме дії супротивника під час ігрового процесу. Користувач повинен мати можливість в будь-який момент призупинити гру, відкрити меню налаштування та покинути гру. Також необхідно забезпечити зберігання інформації про поточні налаштування гри та ігрових колод після завершення програми, щоб уникнути втрати даних та забезпечити зручність користування при наступному запуску програми. Уся інформація, що потребує збереження, має зберігатися у файлах формату .json.

Ігровий процес має відповідати наступним правилам: на початку гри, після видачі кожному гравцю чотирьох карт з їх колод, між двома гравцями випадковим чином обирається той, що ходитиме першим. На початку ходу, гравцю видається одна додаткова карта, а також гравцю надається обмежена кількість мана-кристалів, які він може витратити на розігрування карт. Гравець

може грати карти зі своєї руки, розташовуючи їх на ігровому полі. Кожна карта має вартість мана-кристалів, і гравець може грати лише ті карти, на які він має достатньо мана-кристалів. Гравці можуть використовувати свої карти для атаки інших карт, героя супротивника або для активації спеціальних ефектів. Переможець визначається, коли показник здоров'я героя одного з гравців падає до нуля і виходить з гри. Той гравець, чий герой «помер» першим, програє.

Цільовою аудиторією розроблюваної колекційної карткової гри є любителі карткових ігор, колекціонування та змагань в ігровому просторі.

Гра має бути створена за допомогою платформи Unity з використанням мови програмування C#, налагоджена та якісно протестована з ціллю уникнення багів, що псуватимуть ігровий досвід користувача.

Загалом програмне забезпечення має відповідати усім функціональним та нефункціональним вимогам, що наведені у цьому розділі

Висновки до розділу

Отже, у даному розділі було наведено загальні положення по обраній предметній області, було змістовно описано та проаналізовано предметну область. Було проаналізоване відоме алгоритмічне рішення, що допоможе у реалізації штучного інтелекту в розроблюваній колекційній картковій грі. Проаналізовано засіб розробки під назвою «Unity» та наведено аргументи доцільності його використання для розроблюваного програмного забезпечення. Було проаналізовано та порівняно відомі аналоги розроблюваного програмного забезпечення.

Крім того було проведено аналіз функціональних вимог: побудовано діаграму варіантів використання та модель вимог у загальному вигляді, побудовано матрицю трасування вимог, що ілюструє залежність відповідного сценарію використання від функціональних вимог. Також наведені нефункціональні вимоги ПЗ.

Колекційна карткова гра буде розроблюватися з використанням ігрового рушія Unity та використанням мови програмування C#. В якості алгоритму для

роботи штучного інтелекту було вирішено використати спрощену версію метода Монте-Карло. Уся інформація, що потребує збереження, буде зберігатися у файлах формату .json.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису основних бізнес-процесів програмного забезпечення використаємо BPMN моделі. Бізнес-процеси, описані за допомогою BPMN наведені на рисунках 2.1 – 2.2.

2.1.1 Опис процесу відображення карт колоди користувача в меню налаштування колод:

– користувач відкриває меню налаштування колод;

– менеджер управління колодами бере інформацію про усі доступні карти в грі, карти колоди користувача та карти колоди супротивника зі сховища даних;

– менеджер управління колодами створює колекції об’єктів (колоди карт) на основі інформації про колоди;

– менеджер відображання колод перевіряє кожну карту в колекції, щоб побачити, чи присутня вона в колоді гравця;

– якщо карта присутня в колоді гравця, карта зафарбовується зеленим кольором;

– карта додається на об’єкт відображення колоди гравця;

– користувач обирає колоду користувача для відображення

– відображається колоди користувача.

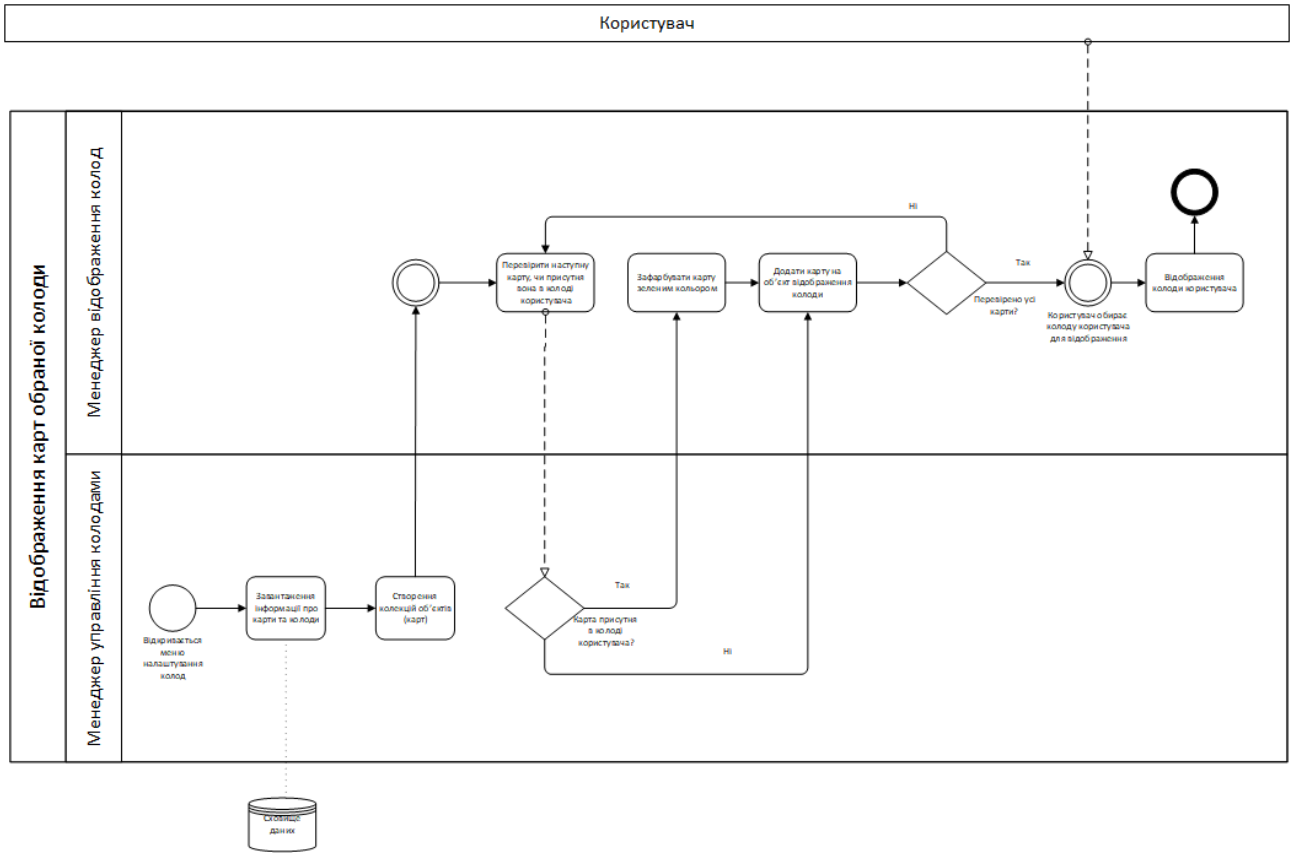

Рисунок 2.1 – Схема бізнес-процесу відображення карт колоди користувача в меню налаштування колод

2.1.2 Опис процесу гри:

– початок гри;
– ініціалізація гри;
– роздача початкових карт;
– випадково визначається гравець, що ходить першим;
– початок ходу;
– якщо хід гравця, то гравець виконує свій хід;
– якщо хід ШІ, то ШІ виконує свій хід
– завершення ходу;
– перевірка умов перемоги;
– якщо умови виконані, виведення результату гри;
– кінець гри;

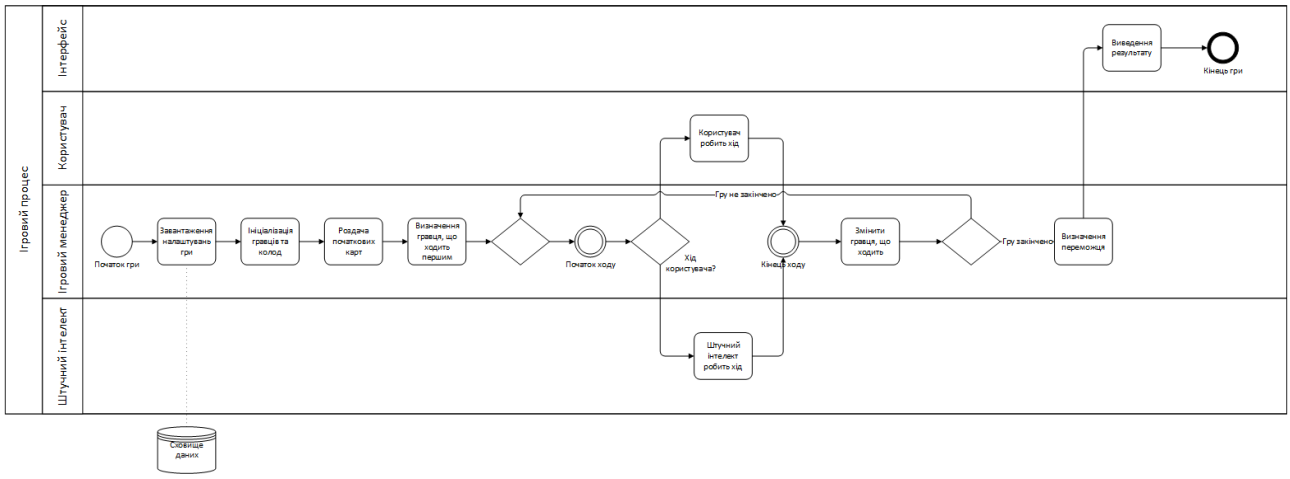

Рисунок 2.2 – Схема бізнес-процесу відображення карт колоди користувача в меню налаштування колод

2.2 Архітектура програмного забезпечення

У структурі розроблюваної колекційної карткової гри ключову роль відіграє концепція сцен, кожна з яких представляє окремий фрагмент ігрового процесу та користувацького інтерфейсу. Кожна сцена містить свій власний набір класів та компонентів, які відповідають за відображення інтерфейсу, обробку вводу користувача та управління ігровим станом. На наступній діаграмі (Рисунок 2.3) представлено високорівневий огляд основних сцен гри та можливі шляхи переходу між ними, що формують кістяк навігації користувача в ігровому середовищі.

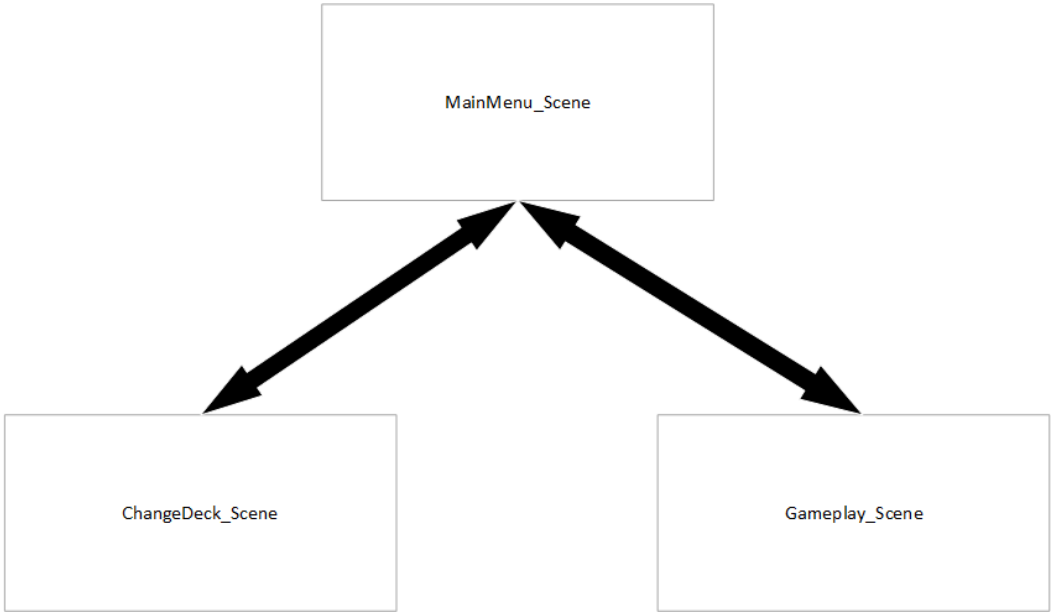

Рисунок 2.3 – Діаграма сцен ККГ

Для деталізації компонентів програмного забезпечення створимо діаграму класів, що наведено на рисунку 2.4, та наведемо детальний опис функціоналу кожного класу (таблиця 2.1).

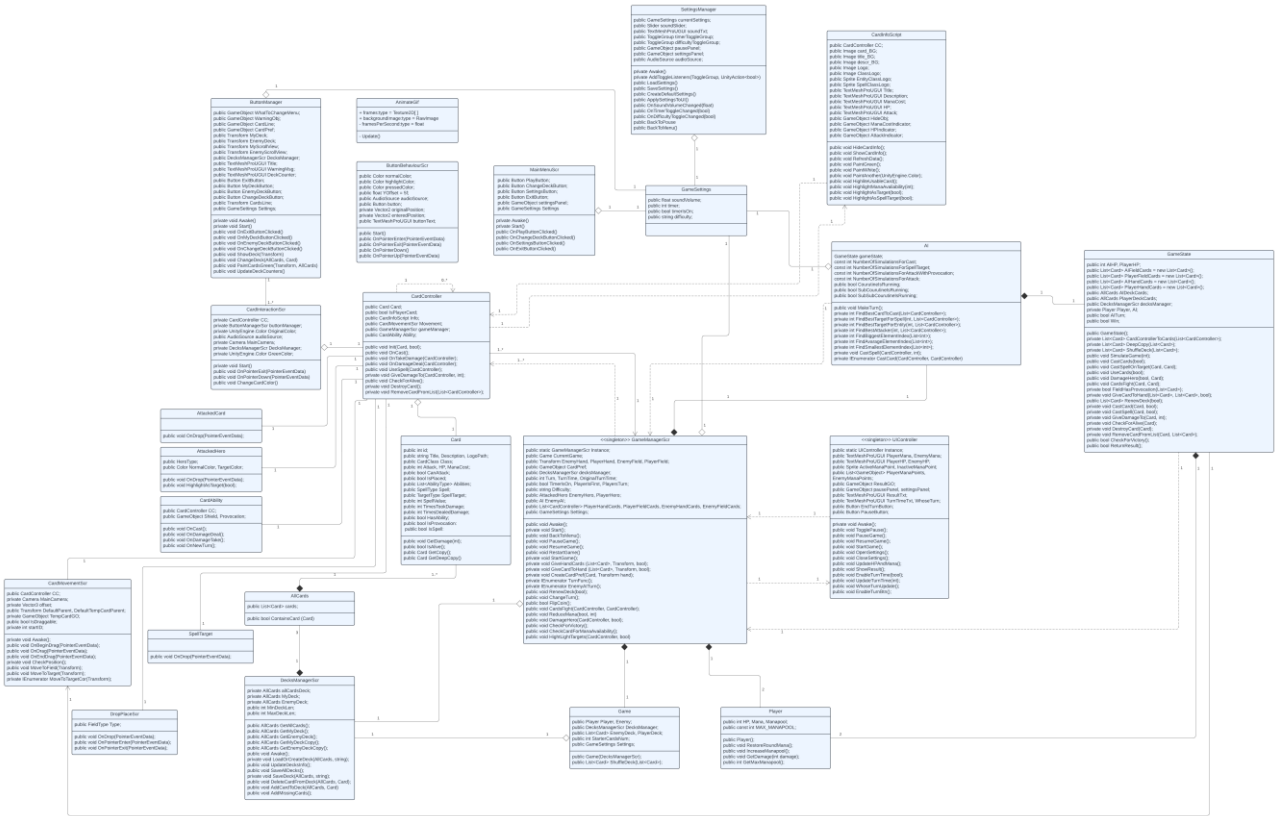

Рисунок 2.4 – Діаграма класів

Таблиця 2.1 – Опис класів

Номер	Назва класу	Опис
1	MainMenuScr	Керує головним меню гри. Відповідає за обробку натискань на кнопки, таких як "Грати", "Змінити колоду", "Налаштування" та "Вихід", а також за перехід між сценами.
2	ButtonBehaviour	Відповідає за зміну візуальної поведінки кнопок під час взаємодії з ними, включаючи наведення курсору, натискання та відпускання. Змінює колір тексту кнопки залежно від дії користувача і регулює положення тексту для створення ефекту натискання. Клас також відтворює звук при натисканні на кнопку, якщо кнопка активна.
3	AnimateGif	Використовується для анімації серії зображень (кадрів), симулюючи відтворення GIF-анімації.
4	GameSettings	Використовується для зберігання налаштувань гри. Клас є серіалізованим, що дозволяє зберігати і відновлювати його стан.
5	SettingsManager	Відповідає за управління налаштуваннями гри, збереженням та завантаженням їх із файлу, а також за відображення налаштувань у візуальному інтерфейсі гри.
6	Card	Зберігає інформацію про ігрову карту з її атрибутами, такими як назва, опис, вартість мани, атака, здоров'я та здібності. Містить методи для обробки пошкоджень та перевірки життєздатності карт. Клас є серіалізованим, що дозволяє зберігати і відновлювати його стан.
7	AllCards	Представляє колоду карт, зберігаючи список карт.

8	DecksManagerScr	Керує колодами карт у грі. Він містить три основних колоди: «allCardsDeck» (всі можливі карти), «MyDeck» (колода гравця) та «EnemyDeck» (колода супротивника). Клас має методи для завантаження, збереження, оновлення інформації в колодах, а також для додавання та видалення карт з колод. Він також забезпечує функціонал для створення копій колод, що можуть використовуватися в ігровій логіці без зміни оригінальних колод.
9	Game	Відповідає за налаштування ключових аспектів однієї ігрової сесії. Він ініціалізує гравців, керує їхніми колодами та встановлює початкові налаштування гри.
10	GameManagerScr	Відповідає за координацію основних ігрових процесів, таких як управління ходами, контроль за грою та її станом, а також управління картами та гравцями. Цей клас забезпечує взаємодію між різними компонентами гри, такими як ШІ, карти, гравці та ігрове поле, координуючи логіку гри та ігровий інтерфейс. Також використовується для організації ігрового процесу (наприклад, роздачі карт, управління здоров'ям героїв, перемішування колод) та відстеження поточного стану гри.
11	CardController	Відповідає за управління поведінкою окремих карт. Цей клас містить методи та властивості, необхідні для ініціалізації карт, управління їхнім відображенням, реагування на

		геймплейні події (наприклад, нанесення чи отримання пошкоджень), та виконання специфічних дій залежно від типу карт (наприклад, застосування заклинань).
12	CardInfoScript	Керує візуальним відображенням інформації про карту в грі. Він пов'язаний з класом CardController, з якого отримує дані про карту, і використовує ці дані для оновлення тексту та зображень на ігровому інтерфейсі.
13	AttackedCard	Відповідає за обробку подій атаки нп карту в ігровому процесі. Цей клас реалізує інтерфейс IDropHandler, що дозволяє йому обробляти події "перетягування та відпускання" (drag and drop).
14	AttackedHero	Реалізує функціонал, пов'язаний з обробкою подій атаки на героїв у грі. Цей клас реалізує інтерфейс IDropHandler, що дозволяє йому обробляти події "перетягування та відпускання" (drag and drop).
15	CardAbility	Забезпечує управління спеціальними здібностями карток у грі. Він взаємодіє з картками, представленими об'єктами CardController, та використовує різні методи для активації та обробки здібностей карток.
16	CardMovementScr	Реалізує логіку переміщення карток у грі. Він відповідає за взаємодію карток з користувачем, їхнє перетягування та позиціонування на ігровому полі.
17	DropPlaceScr	Визначає функціональність розміщення карт у певних місцях в ігровому середовищі, таких як

		рука гравця чи ігрове поле. Він використовує події перетягування для керування тим, де і коли картки можуть бути розміщені, залежно від правил гри, таких як доступна мана гравця і чий зараз хід.
18	SpellTarget	Визначає поведінку при націлюванні заклинань на ігрові карти. . Цей клас реалізує інтерфейс IDropHandler, що дозволяє йому обробляти події "перетягування та відпускання" (drag and drop), дозволяючи гравцям застосовувати заклинання з однієї карти на іншу.
19	Player	Клас представляє ігрового персонажа гравця (героя), керуючи його здоров'ям та маною.
20	AI	Контролює дії ШІ під час його ходу, роблячи вибір карт для розігрування, визначення цілей для атаки та використання заклинань. Ключова функція цього класу полягає у симуляції гри, починаючи з різних ігрових станів, щоб обрати найбільш ефективні ходи.
21	GameState	Клас представляє поточний стан гри. Він відстежує здоров'я гравця та ШІ, а також зберігає картки, які знаходяться на полі та у руці гравців. Клас відповідає за симуляцію ходів гри, включаючи кидання заклинань, атаки картами та перевірку умов перемоги. Цей клас використовується для аналізу потенційних ходів та їх впливу на результат гри, допомагаючи ШІ ухвалювати рішення про наступні дії.

22	UIController	Відповідає за керування інтерфейсом користувача під час гри. Він містить методи для оновлення інформації про здоров'я та ману гравців, відображення результатів гри та керування елементами UI, такими як панель паузи та налаштувань.
23	ButtonManager	Відповідає за управління інтерфейсом користувача у меню налаштування ігрових колод. Він дозволяє гравцям переглядати, модифікувати та змінювати свої колоди карт, керуючи візуальними елементами. Клас також включає функції для оновлення візуального представлення колоди і реагування на дії користувача через інтерфейс.
24	CardInteractionScr	Відповідає за обробку взаємодії користувача з картами в меню налаштування ігрових колод. Він дозволяє змінювати колір карт, вказуючи на їх додавання або видалення з колоди, та реагує на натискання користувача.

2.3 Конструювання програмного забезпечення

2.3.1 Опис алгоритму пошуку оптимального ходу ШІ

Обраний підхід пошуку оптимального ходу ШІ полягає у виконанні великої кількості симуляцій ігрових сценаріїв для визначення оптимальних ходів. Процес складається з наступних кроків:

1. Штучний інтелект проводить тисячі симуляцій для кожного можливого ходу, включаючи викладення карти на ігрове поле, використання заклинань, вибір цілі для заклинання та атаку картами.

Кожна симуляція відтворює усі процеси, властиві справжньому ігровому процесу для кожного ігрового стану у випадковому порядку.

2. Для кожного потенційного ходу ШІ розраховує відсоток виграшів, який він отримав у симуляціях. Це дає ШІ уявлення про те, які ходи є найбільш ефективними у даній ситуації на ігровому полі.

3. На основі результатів симуляцій ШІ обирає хід, який має найвищу ймовірність привести до перемоги. Цей хід може включати використання певної карти, атаку картою або застосування заклинань.

4. Після вибору найкращого ходу ШІ виконує обраний хід у грі.

Більш детальний опис алгоритму пошуку оптимального ходу ШІ наведено на блок-схемі нижче (Рисунок 2.5).

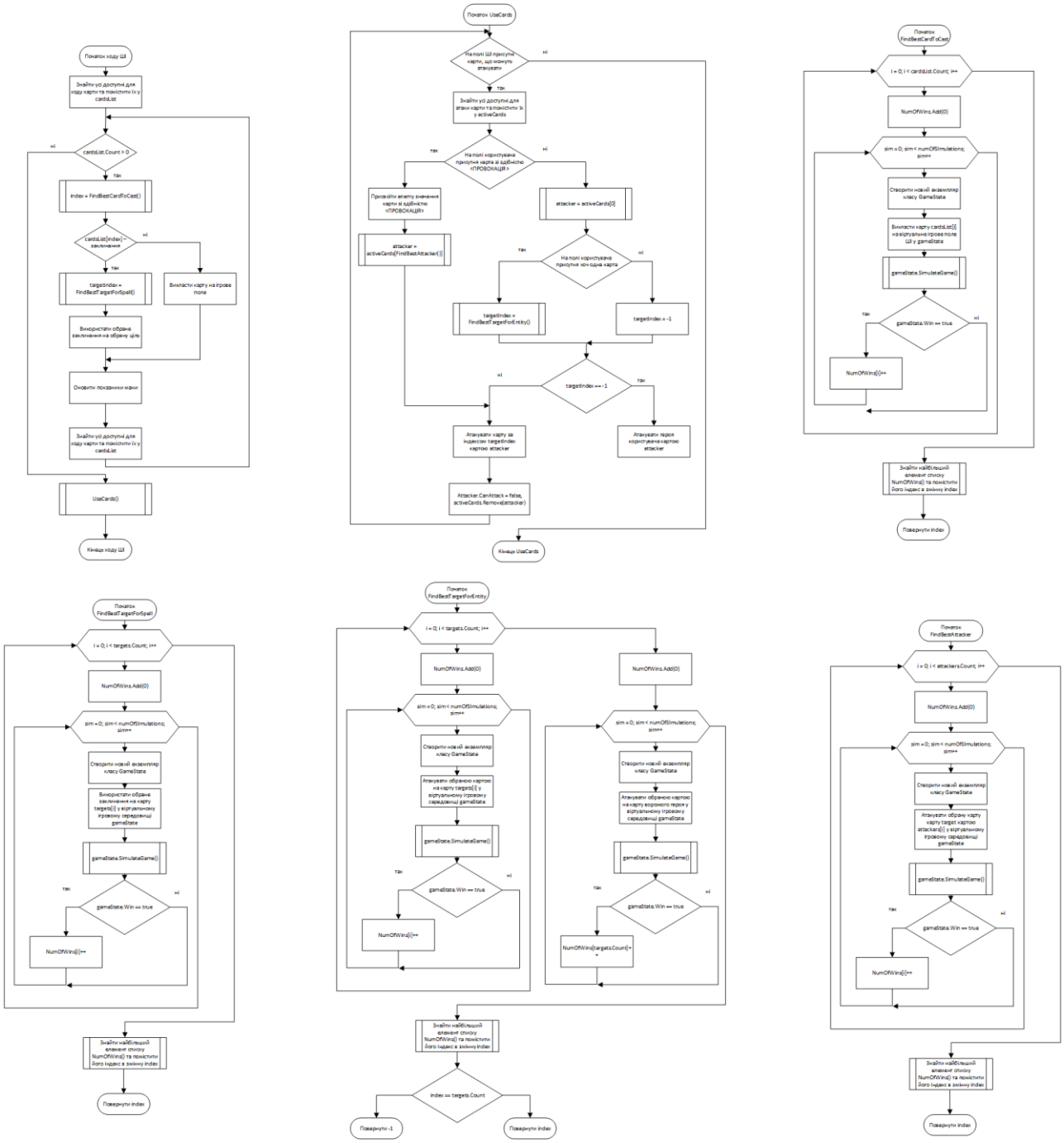

Рисунок 2.5 – Блок-схема алгоритму пошуку оптимального ходу ШІ

2.3.2 Зберігання даних

У рамках проекту для зберігання інформації про картки, ігрові колоди та налаштування гри використовуються JSON-файли. JSON (JavaScript Object Notation) [13] було обрано через його простоту, гнучкість та широку підтримку різними мовами програмування. JSON файли дозволяють легко серіалізувати та

десеріалізувати дані, що значно спрощує обмін даними між різними компонентами системи.

Файли для зберігання інформації про ігрові карти мають наступну структуру (Таблиця 2.2):

Таблиця 2.2 – Структура JSON-файлів для зберігання інформації про ігрові карти

Ключ	Тип даних	Опис
id	int	Унікальний ідентифікатор картки
Title	string	Назва картки
Description	string	Опис здібностей картки
Class	int	Клас картки (enum)
LogoPath	string	Шлях до логотипу картки
Attack	int	Показник атаки картки
HP	int	Показник здоров'я картки
ManaCost	int	Вартість картки

Загалом для зберігання інформації про карти використовуються три json-файли: «AllCards.json», «MyDeck.json» та «EnemyDeck.json». Файл «AllCards.json» використовується лише для зчитування інформації, в ньому зберігається інформація про усі ігрові карти, що доступні в грі. «MyDeck.json» та «EnemyDeck.json» - файли, що слугують для зберігання інформації про те, які карти входять до колод гравців. Вміст цих двох файлів змінюється при редагуванні ігрових колод в меню налаштування колод.

Для зберігання даних про налаштування гри використовується файл «Settings.json», що має наступну структуру (Таблиця 2.3):

Таблиця 2.3 – Структура json-файлу для зберігання інформації про ігрові налаштування

Ключ	Тип даних	Опис
soundVolume	float	Гучність звуку, від 0.0 до 1.0
timer	int	Час таймера в секундах

timerIsOn	boolean	Чи активний таймер (true/false)
difficulty	string	Рівень складності гри («Easy», «Normal», «Hard»)

Вміст файлу «Settings.json» зчитується при запуску гри для виставлення ігрових налаштувань, а його зміст змінюється при редагуванні ігрових налаштувань.

Усі дані для зчитування (JSON-файли, зображення) зберігаються у папці Resources [14], що використовується для зберігання і динамічного завантаження ресурсів у грі на платформі Unity.

2.3.3 Взаємодія користувача з програмою

Для реалізації взаємодії користувача з ігровими об'єктами було використано декілька ключових інтерфейсів Unity, які дозволяють забезпечити плавну та інтуїтивно зрозумілу взаємодію. У таблиці 2.4 наведено назви використаних інтерфейсів, їх методи, описи та реалізації.

Таблиця 2.4 – Використані інтерфейси Unity

Інтерфейс	Метод	Опис	Реалізація
IPointerEnterHandler	OnPointerEnter	Відповідає за обробку подій, коли курсор наводиться на об'єкт.	Використовується в класі ButtonBehaviourScr для зміни кольору та позиції тексту кнопки, коли користувач наводить курсор на кнопку.
IPointerDownHandler	OnPointerDown	Відповідає за обробку подій, коли на об'єкт натискають.	Використовується в класі ButtonBehaviourScr, змінює колір тексту кнопки та його

			позицію при натисканні на кнопку, а також відтворює звук. Це створює ефект натискання. Також використовується в класі CardInteractionScr для зміни кольору картки, додавання або видалення її з колоди при натисканні.
IPointerUpHandler	OnPointerUp	Відповідає за обробку подій, коли натискання на об'єкт закінчується.	Використовується в класі ButtonBehaviourScr, змінює колір тексту кнопки залежно від того, чи курсор залишається над кнопкою після відпускання кнопки миші. Це дозволяє відобразити візуальний зворотний зв'язок про стан кнопки.
IPointerExitHandler	OnPointerExit	Відповідає за обробку подій,	Використовується в класі

		коли курсор покидає об'єкт.	ButtonBehaviourScr, повертає колір та позицію тексту кнопки до початкового стану, коли користувач веде курсор за межі кнопки.
IBeginDragHandler	OnBeginDrag	Активується на початку перетягування об'єкта.	Використовується в класі CardMovementScr для реалізації механіки перетягування карти, перевірки можливості перетягування та маркування можливих цілей обраної карти.
IDragHandler	OnDrag	Активується під час перетягування об'єкта.	Використовується в класі CardMovementScr для реалізації механіки перетягування карти, оновлення позиції карти під час перетягування.

IEndDragHandler	OnEndDrag	Активується коли перетягування об’єкта закінчується	Використовується в класі CardMovementScr для реалізації механіки перетягування карти, встановлення нового батьківського елемента карти.
IDropHandler	OnDrop	Активується коли об’єкт перетягується та відпускається над компонентом	Використовується в багатьох класах, пов'язаних з ігровим процесом. Слугує для реалізації мехінік атакування карт, атакування героїв, застосування заклинань, встановлення карти на ігрове поле.

2.3.4 Використання класу MonoBehaviour

MonoBehaviour [15] є фундаментальним класом в Unity, від якого автоматично успадковуються всі скрипти, що взаємодіють з ігровими об'єктами та сценою. Клас MonoBehaviour надає структуру, яка дозволяє прикріпляти створені скрипти до певних ігрових об’єктів на сцені (GameObject) у редакторі. Також клас дозволяє скриптам реагувати на ключові моменти ігрового циклу через методи, такі як Start() (викликається перед першим кадром) та Update() (викликається кожен кадр).

2.3.5 Опис додаткових утиліт, що використовуються в розробці

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.5.

Таблиця 2.5 – Опис використаних утиліт

Назва утиліти	Опис застосування
Unity	Головне середовище розробки для створення інтерактивних ігрових проектів. Використовувався для розробки, тестування та відлагодження ігрових сцен, об'єктів та скриптів.
Visual Studio 2022	Основне інтегроване середовище розробки (IDE) для написання та відлагодження коду C# у проектах Unity. Використовувався для створення скриптів, реалізації логіки гри та інтерфейсів користувача.
ChatGPT 4.0	Інструмент штучного інтелекту від OpenAI. Використовувався для генерації логотипів ігрових карт.

Висновки до розділу

Отже, в розділі «Моделювання та конструювання програмного забезпечення» було описано бізнес-процеси розроблюваної колекційної карткової гри: процес відображення карт колоди користувача в меню налаштування колод та процес гри. Були наведені BPMN-моделі для зазначених процесів.

Також було розглянуто архітектуру програмного забезпечення. Наведено високорівневий огляд ігрових сцен та можливі шляхи переходу між ними. Була розроблена діаграма класів, що відображає класи, використовувані в програмі та відношення між ними. До кожного класу наведено детальний опис функціоналу у таблиці 2.1.

В підрозділі «Конструювання програмного забезпечення» описано алгоритм роботи ігрового штучного інтелекту, а також наведено блок-схему алгоритму. Були наведені способи зберігання даних, що використовуються у роботі ПЗ, шляхи реалізації інструментів взаємодії користувача з системою та додаткові утиліти з описаннями, що використовувались під час розробки колекційної карткової гри.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

У контексті розробки програмного забезпечення статичний аналіз якості коду є важливим, оскільки він допомагає виявляти потенційні проблеми на ранніх етапах циклу розробки, що може зекономити час і ресурси в довгостроковій перспективі.

Аналіз якості програмного забезпечення було проведено за допомогою застосунку NDepend [16], що автоматично генерує звіт якості коду за різними метриками. Звіт аналізу якості коду наведено на рисунку 3.1.

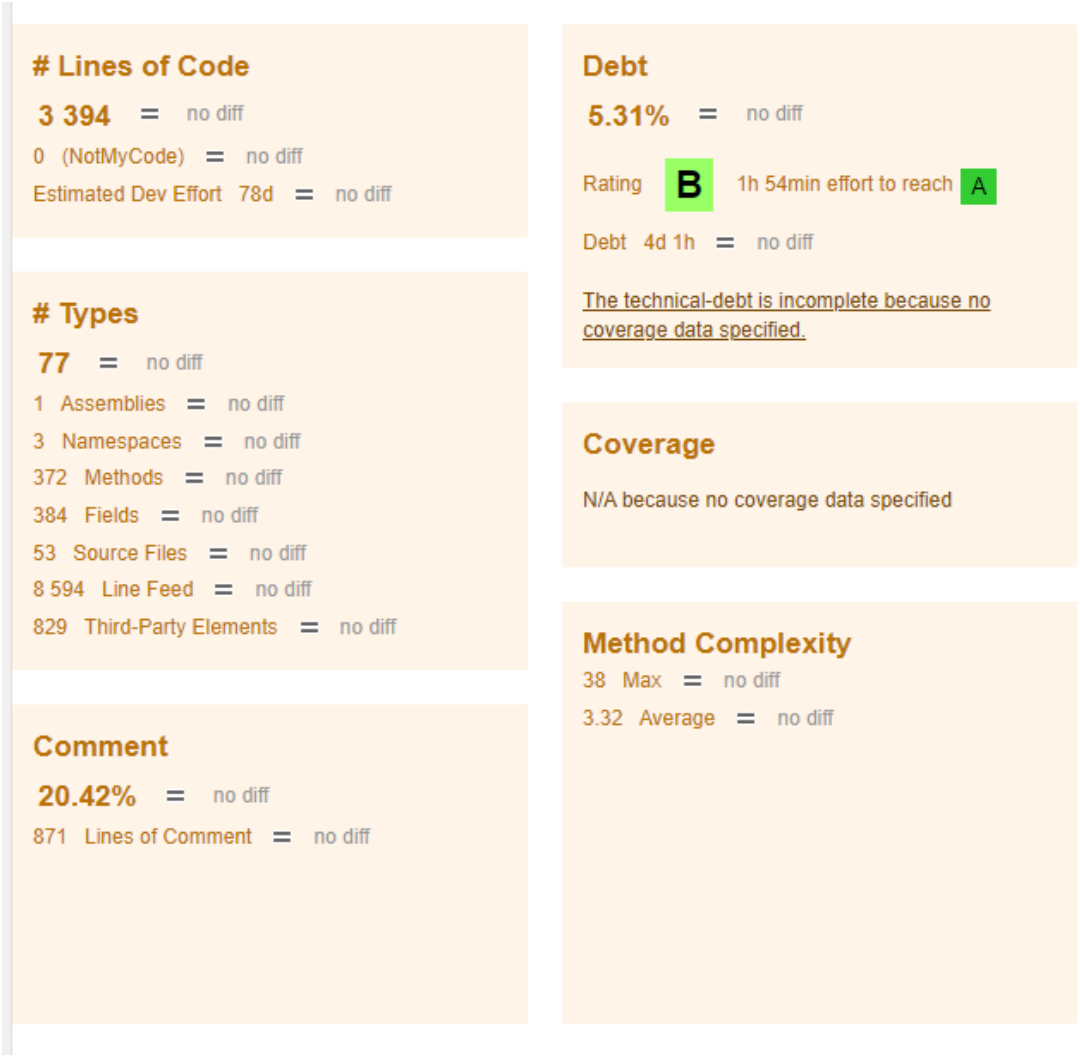

Рисунок 3.1 – Звіт якості коду ПЗ

З отриманого звіту аналізу якості програмного забезпечення видно, що у застосунку 3 394 рядків коду. В проекті наявні 77 різних типів та 871 рядок коментарів, що становить 20,42% усього коду. Міра «технічного боргу» [17] становить 5,31% та має оцінку «B». Максимальна оцінка складності одного метода становить 38, а середня складність – 3,32. Метрики складності методів можуть вказувати на те, наскільки важко може бути підтримувати та тестувати код.

3.2 Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.7.

Таблиця 3.1 – Тест 1

Тест	Перевірка коректності роботи ігрових налаштувань
Номер тесту	1
Початковий стан системи	Користувач знаходиться у меню налаштування гри
Вхідні данні	-
Опис проведення тесту	Змінюються налаштування гучності звуку, таймеру, ігрової складності. Перевіряється, чи були застосовані виставлені параметри в грі.
Очікуваний результат	Усі виставлені параметри були успішно застосовані. Рівень гучності звуку змінюється в реальному часі при взаємодії з повзунком, під час ігрової сесії діють виставлені налаштування таймера та складності.
Фактичний результат	Усі виставлені параметри були успішно застосовані. Рівень гучності звуку змінюється в реальному часі при взаємодії з повзунком, під час ігрової сесії діють виставлені налаштування таймера та складності.

Таблиця 3.2 – Тест 2

Тест	Перевірка коректного збереження даних ігрових налаштувань
Номер тесту	2

Продовження таблиці 3.2

Початковий стан системи	Користувач знаходиться у меню налаштування гри
Вхідні данні	-
Опис проведення тесту	Змінюються налаштування гучності звуку, таймеру, ігрової складності. Виходимо з додатку. Відкриваємо додаток знову. Заходимо у меню налаштувань. Перевіряємо, які налаштування виставлені.
Очікуваний результат	В меню налаштування гри виставлені ті ж самі параметри, що були виставлені до виходу з додатку.
Фактичний результат	В меню налаштування гри виставлені ті ж самі параметри, що були виставлені до виходу з додатку.

Таблиця 3.3 – Тест 3

Тест	Перевірка коректності роботи функції додавання та видалення карти з колоди
Номер тесту	3
Початковий стан системи	Користувач знаходиться у меню налаштування ігрових колод
Вхідні данні	-
Опис проведення тесту	Натискаємо на 5 карток зеленого кольору задля видалення їх із колоди. Намагаємося додати 6 карток до колоди, шляхом натискання на незафарбовані картки. Видаляємо 2 картки з колоди. Виходимо з меню налаштування ігрових колод.

Продовження таблиці 3.3

Очікуваний результат	При натисканні на зелену картку, вона успішно змінює свій колір та видаляється з колоди. При натисканні на незафарбовану катку, вона успішно зафарбовується зеленим кольором та додається до колоди. Додавання картки до колоди перестає бути можливим, якщо колода вже містить 30 карток. Якщо колода містить кількість карток, меншу за 30, при виході з меню налаштування колод з'являється відповідне повідомлення та колода автоматично заповнюється бракуючими картами.
Фактичний результат	При натисканні на зелену картку, вона успішно змінює свій колір та видаляється з колоди. При натисканні на незафарбовану катку, вона успішно зафарбовується зеленим кольором та додається до колоди. Додавання картки до колоди перестає бути можливим, якщо колода вже містить 30 карток. Якщо колода містить кількість карток, меншу за 30, при виході з меню налаштування колод з'являється відповідне повідомлення та колода автоматично заповнюється бракуючими картами.

Таблиця 3.4 – Тест 4

Тест	Перевірка коректного збереження даних про ігрові колоди
Номер тесту	4
Початковий стан системи	Користувач знаходиться у меню налаштування ігрових колод

Вхідні данні	-
Опис проведення тесту	Заповнюємо ігрову колоду будь-якими картами. Виходимо з додатку. Відкриваємо додаток. Заходимо у меню налаштування ігрових колод. Відкриваємо редактовану колоду. Перевіряємо, чи складається колода з тих самих карт, що були обрані до виходу з додатку.

Продовження таблиці 3.4

Очікуваний результат	Колода складається з тих самих карт, що були обрані до виходу з додатку.
Фактичний результат	Колода складається з тих самих карт, що були обрані до виходу з додатку.

Таблиця 3.4 – Тест 5

Тест	Перевірка коректності роботи здібності «SHIELD»
Номер тесту	5
Початковий стан системи	Ігрова сесія розпочата. На ігрове поле суперника викладено карту зі здібністю «SHIELD».
Вхідні данні	-
Опис проведення тесту	Атакуємо ворожу карту зі здібністю будь-якою своєю картою шляхом перетягування союзної картки та відпускання її над ворожою карткою зі здібністю.
Очікуваний результат	Перед атакуванням ворожої картки зі здібністю «SHIELD», карта зі здібність містить на собі значок у формі щита. Після атакування карти зі дібністю, значок у формі щита зникає, але здоров'я не зменшується.
Фактичний результат	Перед атакуванням ворожої картки зі здібністю «SHIELD», карта зі здібність містить на собі значок у формі щита. Після атакування карти зі дібністю, значок у формі щита зникає, але здоров'я не зменшується.

На рисунках 3.2 та 3.3 продемонстровано, який вигляд та які параметри має картка зі здібністю «ЩИТ» до та після її атакування.

Рисунок 3.2 – Карта зі здібністю «SHIELD» до того, як бути атакованою

Рисунок 3.3– Карта зі здібністю «SHIELD» після її атакування

Таблиця 3.6 – Тест 6

Тест	Перевірка коректності роботи здібності «PROVOCATION»
Номер тесту	6

Продовження таблиці 3.6

Початковий стан системи	Ігрова сесія розпочата. На ігрове поле суперника викладено карту зі здібністю «PROVOCATION».
Вхідні данні	-
Опис проведення тесту	Починаємо перетягувати союзну карту з ігрового поля на будь-яку ціль, окрім карти зі здібністю, для атакування.
Очікуваний результат	При перетягуванні союзної карти з ігрового поля, жодна ціль, окрім карти зі здібністю «PROVOCATION» не підсвічується. Атакувати інші цілі неможливо.
Фактичний результат	При перетягуванні союзної карти з ігрового поля, жодна ціль, окрім карти зі здібністю «PROVOCATION» не підсвічується. Атакувати інші цілі неможливо.

На рисунках 3.4 та 3.5 продемонстровано, які цілі підсвічуються, як можливі для атаки, при наявній карті зі здібністю «PROVOCATION» на ворожому полі та без неї.

Рисунок 3.4– Можливі цілі для атаки у присутності карти зі здібністю «PROVOCATION»

Рисунок 3.4– Можливі цілі для атаки, коли карта зі здібністю «PROVOCATION» відсутня

Таблиця 3.7 – Тест 7

Тест	Перевірка коректності роботи системи мана-кристалів
Номер тесту	7
Початковий стан системи	Ігрова сесія розпочата. Користувач має 1 мана-кристал.
Вхідні данні	-
Опис проведення тесту	Намагаємося викласти на поле карту із вартістю більшою за 1. Викладаємо на поле карту вартістю 1.
Очікуваний результат	На початку ходу усі карти, що мають вартість більшу за поточну кількість мана-кристалів гравця, помічаються як недоступні. При спробі викласти їх на ігрове пол нічого не вдається. При викладанні карти на ігрове поле,

	забирається кількість мана-кристалів, що дорівнює вартості викладеної карти.

Продовження таблиці 3.7

Фактичний результат	На початку ходу усі карти, що мають вартість більшу за поточну кількість мана-кристалів гравця, помічаються як недоступні. При спробі викласти їх на ігрове пол нічого не вдається. При викладанні карти на ігрове поле, забирається кількість мана-кристалів, що дорівнює вартості викладеної карти.

Висновки до розділу

Отже, в розділі «Аналіз якості та тестування програмного забезпечення» було проведено аналіз якості програмного забезпечення за допомогою застосунку Ndepend та наведено звіт аналізу. В звіті описані наступні метрики: кількість рядків коду, кількість типів, кількість рідків з коментарями, «технічний борг», метрика складності методів.

Були мануально протестовані основні функції програмного забезпечення. Для кожного тест-кейсу були наведені таблиці, що містять описи проведеного тесту, очікувані та фактичні результати. Також надані рисунки, що ілюструють роботу функцій там, де ілюстрація роботи необхідна. Було протестовано коректність роботи ігрових налаштувань та їх збереження, коректність роботи функції видалення та додавання карт до ігрових колод та збереження інформації про ігрові колоди. Також були протестовані основні здібності карт та коректність роботи системи мана-кристалів.

Загалом програмне забезпечення задоволняє ямови якісної роботи. Критичних помилок у роботі ПЗ під час тестування виявлено не було.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Створену колекційну карткову гру було вирішено розгорнути на платформі Itch.io. Itch.io [18] – це вебсайт, призначений для розміщення, продажу та завантаження інді-відеоігор. Через велику свободу, яку ігрові розробники мають на сайті, платформа вважається гарним способом розгортання відеоігор для новачків у цій сфері.

Загалом присутні два способи розгортання гри: завантаження з сайту та завантаження через клієнт Itch.io.

Для розгортання ККГ через сайт платформи Itch.io необхідно виконати наступні кроки:

– перейти за посиланням: https://mizantrop4real.itch.io/exec-magica;
– на відкритій сторінці натиснути кнопку «Download» (Рисунок 4.1), після натискання почнеться завантаження архіву з усіма необхідними файлами для роботи додатку;
– розархівувати файли гри;
– для запуску гри необхідно запустити файл «EXEC_MAGICA.exe» (Рисунок 4.2).

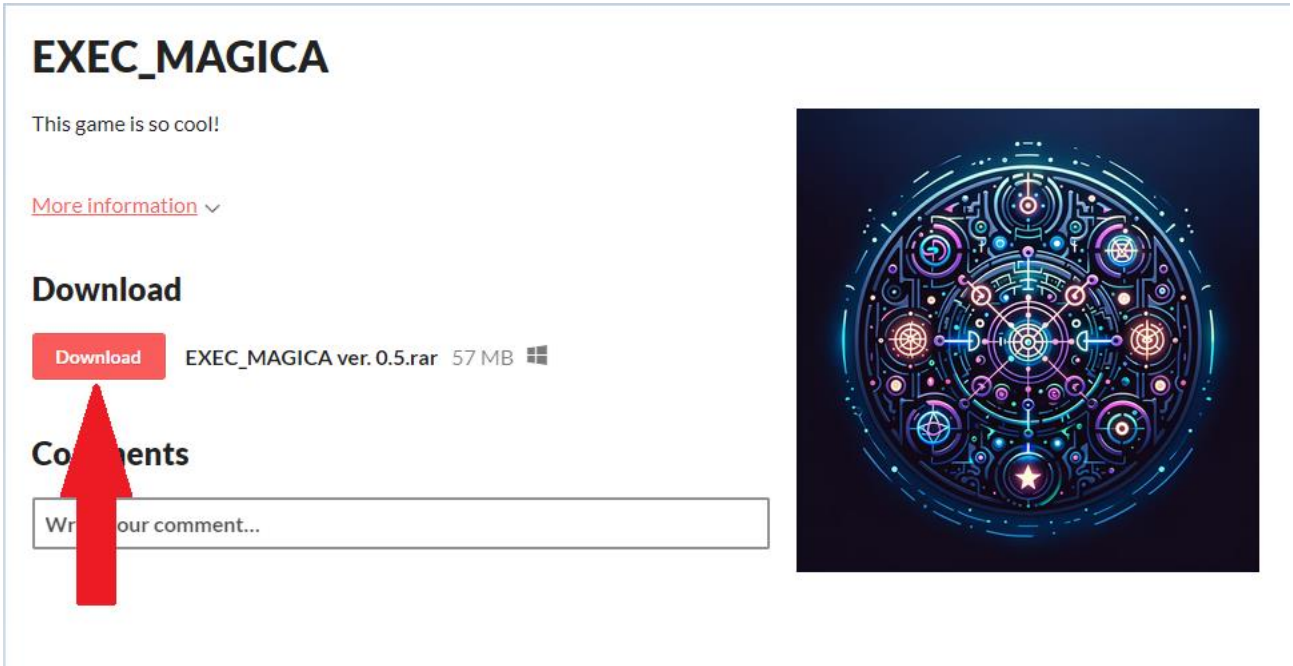

Рисунок 4.1– Кнопка завантаження файлів гри

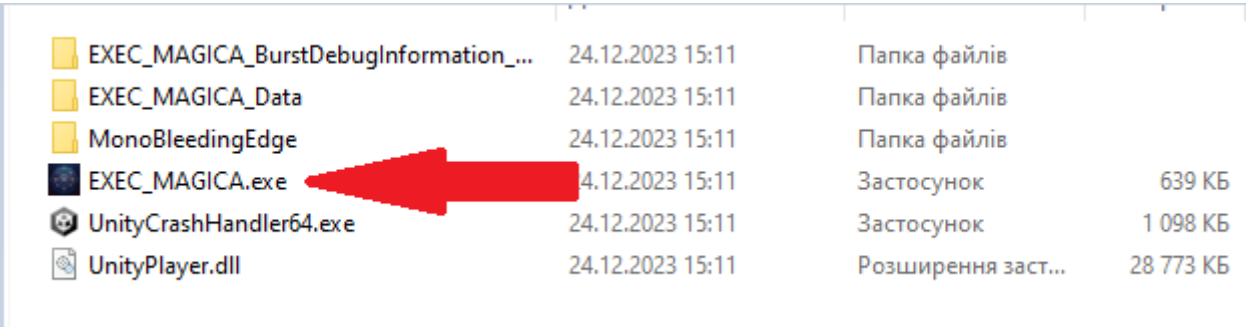

Рисунок 4.2– EXE-файл для запуску додатку

Для розгортання ККГ через клієнт платформи Itch.io необхідно виконати наступні кроки:

– перейти за посиланням: https://itch.io/app та завантажити клієнт Itch.io натиснувши на кнопку «Start Download» (Рисунок 4.3);
– встановити клієнт Itch.io за допомогою завантаженого EXE-файлу;
– зареєструватися на сайті https://itch.io/;
– після реєстрації увійти в свій обліковий запис в клієнті Itch.io (Рисунок 4.4);
– знайти гру за допомогою функції пошуку та перейти за посиланням (Рисунок 4.5);
– натиснути кнопку «Встановити» (Рисунок 4.6), обрати директорію для розміщення файлів гри та почати встановлення (Рисунок 4.7);
– по завершенню встановлення додатку необхідно натиснути кнопку «Запустити» у вікні завантажень для запуску гри (Рисунок 4.8).

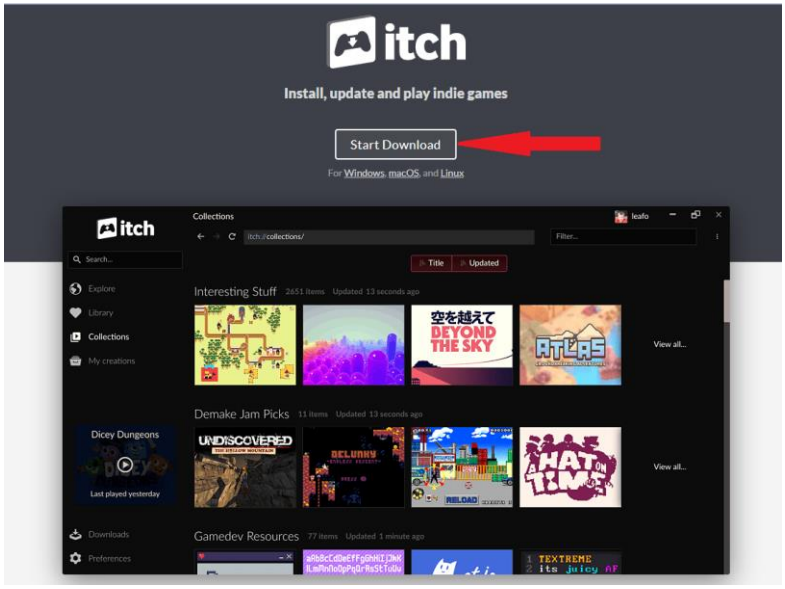

Рисунок 4.3– Завантаження клієнта Itch.io

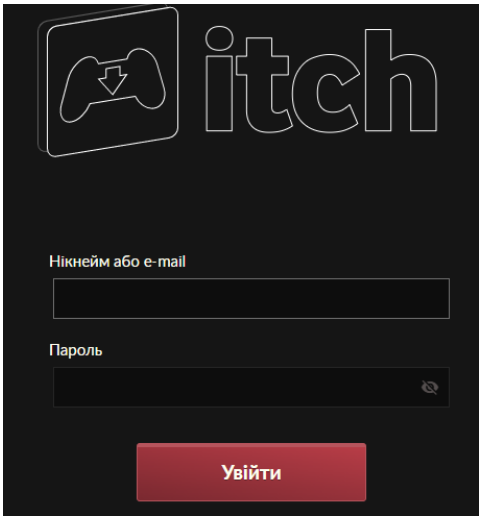

Рисунок 4.4– Авторизація користувача у клієнті Itch.io

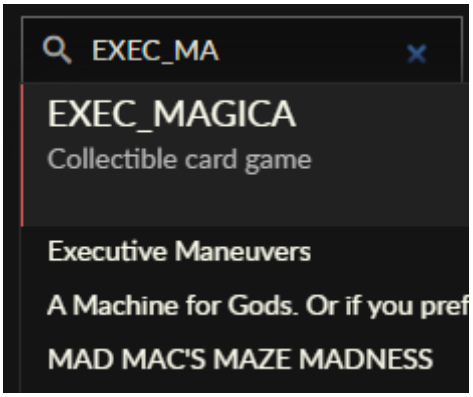

Рисунок 4.5 – Пошук ігрового додатку за ім'ям у клієнті

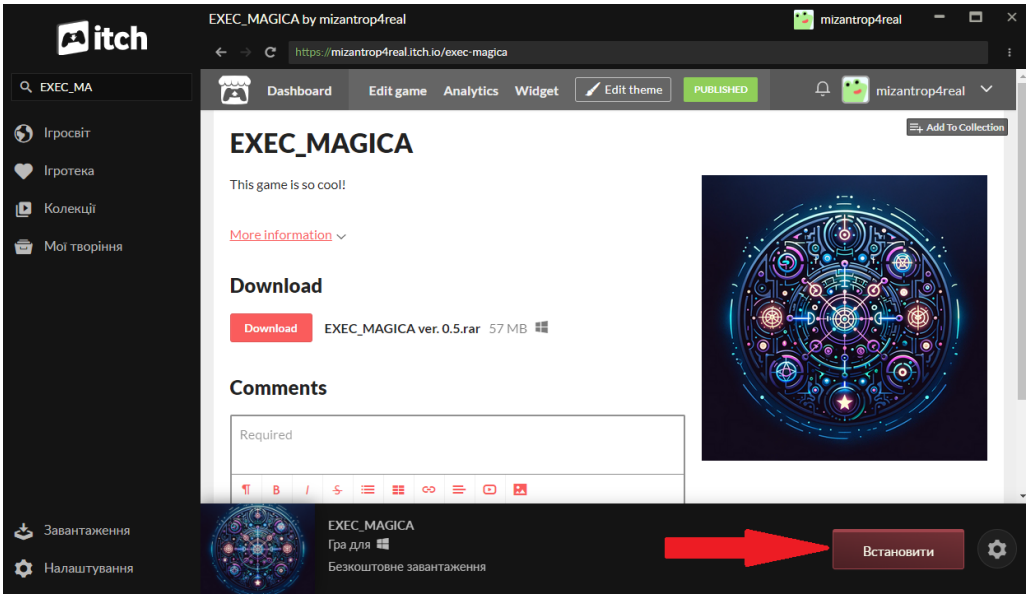

Рисунок 4.6 – Кнопка встановлення

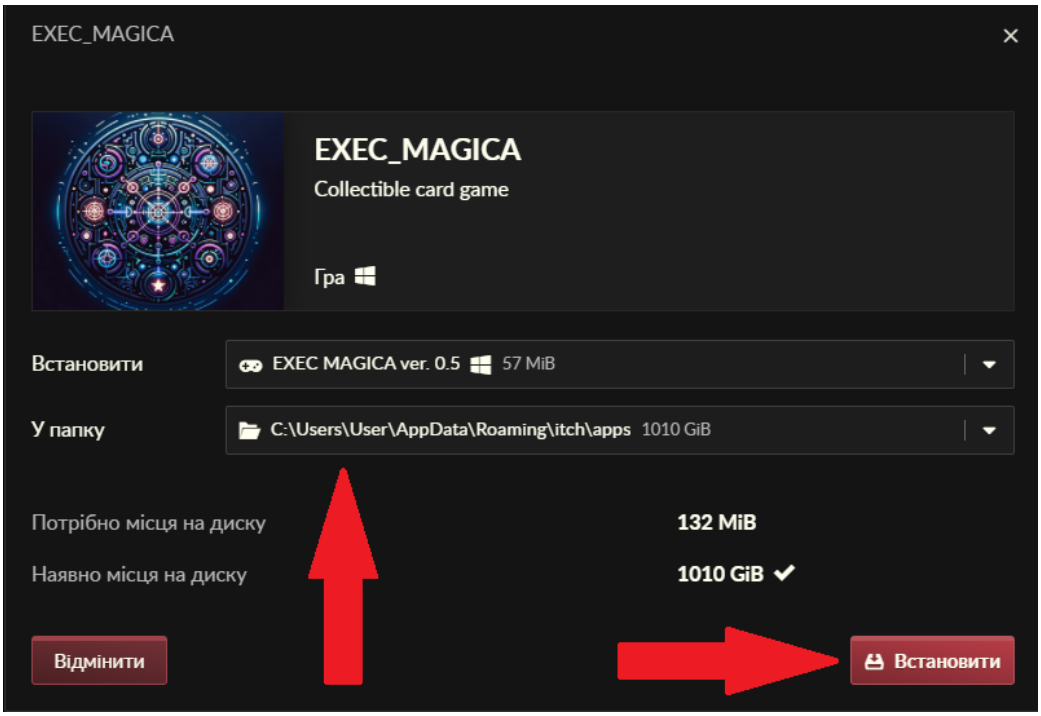

Рисунок 4.7 – Вибір директорії та встановлення додатку

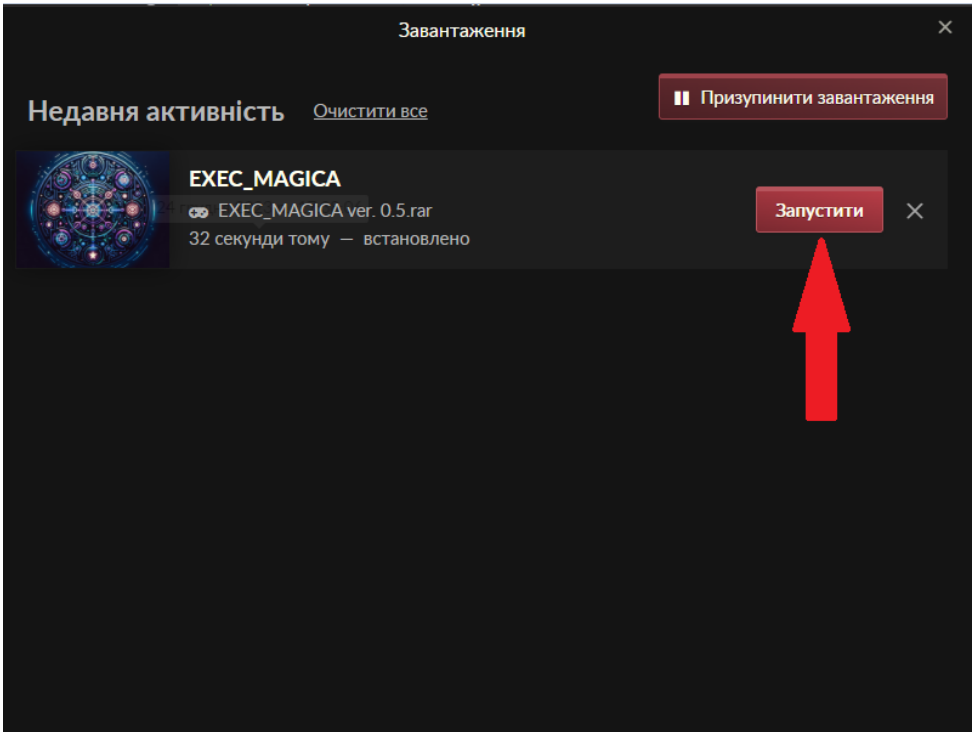

Рисунок 4.8 – Запуск додатку з вікна завантажень

4.2 Підтримка програмного забезпечення

Способи оновлення ігрового додатку відрізняються в залежності від шляху, яким було інстальовано ігровий додаток. Якщо програмне забезпечення було встановлене напряму з сайту платформи Itch.io, то для інсталяції останньої версії гри користувачу необхідно видалити попередню версію гри з комп'ютера (це може бути необхідно, щоб уникнути конфліктів між файлами) та знову встановити ПЗ слідуючи крокам для встановлення програми з сайту, описаним у попередньому підрозділі.

Якщо ж ігровий додаток було встановлено за допомогою клієнту Itch.io, то для його оновлення до останньої версії необхідно:

– перейти на сторінку ігрового додатку та відкрити меню налаштування (Рисунок 4.9);
– у відкритому вікні, якщо нове оновлення доступне, з'явиться відповідна опція, для оновлення додатку. Щоб розпочати оновлення необхідно натиснути кнопку «Встановити», після чого додаток оновиться до останньої версії (Рисунок 4.10).

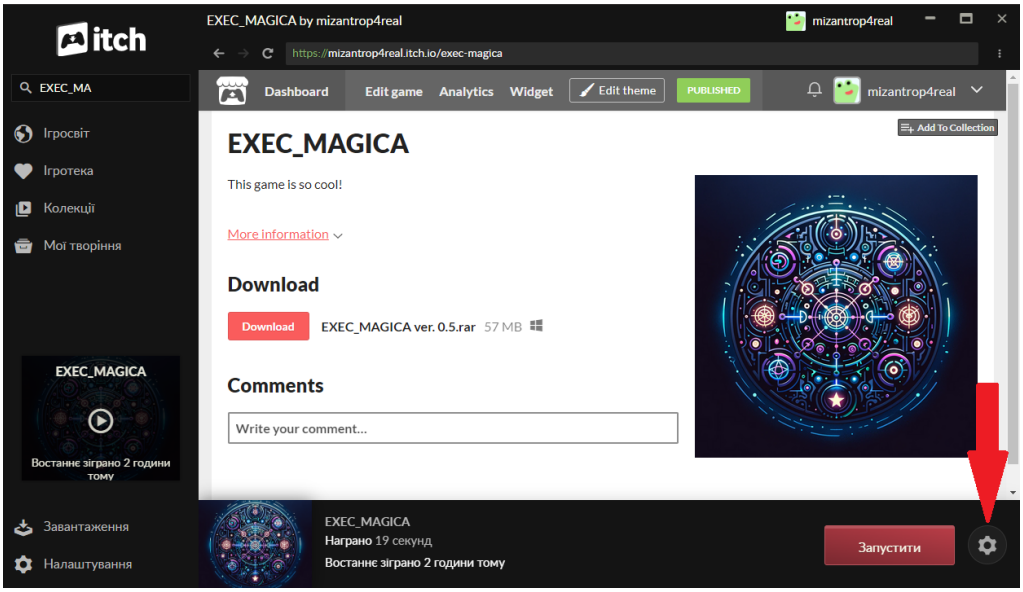

Рисунок 4.9 – Відкриття меню налаштування ігрового додатку

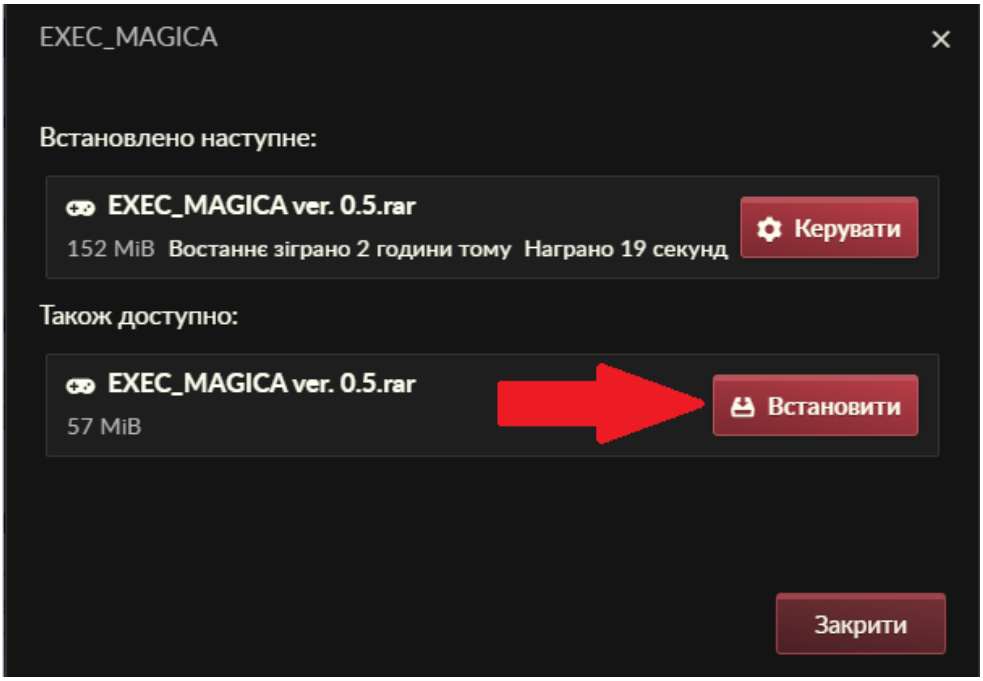

Рисунок 4.10 – Функція оновлення додатку

Висновки до розділу

Отже, у даному розділі було розглянуто процеси розгортання та підтримки програмного забезпечення для колекційної карткової гри.

Для розміщення гри було обрано платформу Itch.io через її доступність, простоту та гнучкість, що робить цю платформу ідеальним вибором для інді-розробників. Розглянуто та детально описано два методи розгортання гри:

безпосередньо через веб-сайт Itch.io та через використання клієнта Itch.io. Також описано процеси оновлення гри в залежності від способу інсталяції.

Загалом, рекомендується використання клієнта Itch.io як первинного способу інсталяції та оновлення ігрового додатку. Використання клієнта значно спрощує процеси встановлення та оновлення, оскільки він автоматизує більшість кроків і забезпечує легке управління ігровими додатками. Це не тільки підвищує зручність для кінцевих користувачів, але й зменшує потенційні технічні труднощі, пов'язані з ручним встановленням та оновленням ігор.

ВИСНОВКИ

В результаті виконання курсової роботи було спроєктоване та реалізоване ігрове програмне забезпечення, що представляє собою Колекційну карткову гру із штучним інтелектом.

В процесі аналізування вимог було наведено загальні положення по предметній області, проведено змістовний опис та аналіз. Були розглянуті відомі алгоритмичні рішення для вирішеної задачі створення ШІ. Проаналізовано засіб розробки під назвою «Unity» та наведено аргументи доцільності його використання для розроблюваного програмного забезпечення. Також були проаналізовані та порівняні відомі успішні ІТ-проекти у наведеній сфері. Було проведено аналіз функціональних та нефункціональних вимог.

У розділі, присвяченому меделюванню та конструюванню програмного забезпечення ми детально описали основні бізнес-процес. Була розглянута архітектура ПЗ, наведена діаграма класів, функціонал та призначення кожного класу були детально описані. Наведена структура сховища даних. Також був детально описаний алгоритм роботи штучного інтелекту, шляхи реалізації інструментів взаємодії користувача з системою та додаткові утиліти з описаннями, що використовувались під час розробки колекційної карткової гри.

Під час аналізу якості був використаний застосунок NDepend, що автоматично генерує звіт якості коду за різними метриками. Отриманий звіт було наведено та проаналізовано. Також були описані процеси тестування очновних ігрових механік ККГ.

У розділі «Впровадження та супровід програмного забезпечення» було розглянуто процеси розгортання та підтримки програмного забезпечення для колекційної карткової гри. Для розміщення гри було обрано платформу Itch.io.

Загалом в результаті роботи над курсовою роботою мною було вирішено усі поставлені задачі в даній курсовій роботі. Розроблено Колекційну карткову гру із штучним інтелектом, що включає функцію редагування ігрових колод користувача та суперника. Були набуті навички роботи з ігровим рушієм Unity для розробки 2D-проєктів. Також були розширені знання у сфері розробки

ігрового штучного інтелекту для покрокових карткових ігор з використанням відомих алгоритмічних рішень.

На мою думку цифрові колекційні карткові ігри будуть надалі розвиватися у майбутньому через їх величезний розважальний та стратегічний потенціал, який виявляється у їх здатності пропонувати глибокі та різноманітні ігрові механіки. Ці ігри вимагають від гравців ретельного планування, тактичного мислення та стратегічного розуміння для досягнення успіху. Вони пропонують не лише розвагу, але й інтелектуальний виклик, поєднуючи елементи випадковості, вміння передбачати ходи противника та управління ресурсами. Цифрові реалізації ККІ додатково розширюють цей потенціал, вдосконалюючи ігровий досвід та роблячи ігри більш доступними та комплексними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Collectible card game [Електронний ресурс] — https://en.wikipedia.org/wiki/Collectible_card_game
2. Magic: The Gathering [Електронний ресурс] — https://en.wikipedia.org/wiki/Magic:_The_Gathering
3. Tabletop role-playing game [Електронний ресурс] — https://en.wikipedia.org/wiki/Tabletop_role-playing_game
4. Hearthstone [Електронний ресурс] — https://en.wikipedia.org/wiki/Hearthstone
5. Monte Carlo tree search [Електронний ресурс] — https://en.wikipedia.org/wiki/Monte_Carlo_tree_search#Pure_Monte_Carlo_game_search
6. Unity (ігровий рушій) [Електронний ресурс] — https://uk.wikipedia.org/wiki/Unity_(%D1%96%D0%B3%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D1%80%D1%83%D1%88%D1%96%D0%B9)
7. DirectX [Електронний ресурс] — https://en.wikipedia.org/wiki/DirectX
8. OpenGL [Електронний ресурс] — https://en.wikipedia.org/wiki/OpenGL
9. Drag and drop [Електронний ресурс] — https://en.wikipedia.org/wiki/Drag_and_drop
10. C Sharp (programming language) [Електронний ресурс] — https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
11. Inscryption [Електронний ресурс] — https://en.wikipedia.org/wiki/Inscryption
12. Gwent: The Witcher Card Game [Електронний ресурс] — https://en.wikipedia.org/wiki/Gwent:_The_Witcher_Card_Game
13. JSON [Електронний ресурс] — https://en.wikipedia.org/wiki/JSON
14. Resources [Електронний ресурс] — https://docs.unity3d.com/ScriptReference/Resources.html
15. MonoBehaviour [Електронний ресурс] — https://docs.unity3d.com/ScriptReference/MonoBehaviour.html
16. NDepend [Електронний ресурс] — https://en.wikipedia.org/wiki/NDepend

17. Technical debt [Електронний ресурс] — https://en.wikipedia.org/wiki/Technical_debt
18. Itch.io [Електронний ресурс] — https://en.wikipedia.org/wiki/Itch.io