Вищий навчальний заклад

«Університет економіки та права «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

на тему Розробка гри жанру стратегія на платформі Android

Виконав _____
(Підпис)

_____ Литвиненко Олександр Русланович _____
(прізвище, ім'я, по батькові)

Науковий керівник

_____Добришин Юрій Євгенович_____
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____

(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____

(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Голова циклової комісії _____ Леонід УВАРОВ
(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Литвиненко Олександр Русланович

1. Тема роботи «Розробка гри жанру стратегія на платформі Android» затверджена наказом по університету від «___» _____ 202__ р. № _____
2. Термін здачі закінченої роботи «30» травня 2025 року
3. Вихідні дані до роботи:
   а) цільова аудиторія – існуючі автоматизовані системи для створення ігор;
   б) функціональність – створення сценаріїв , їх обробка, надсилання результатів гри, обробка інформації, тощо;
   в) технічні вимоги - платформа для розробки сучасних ігор на платформі Android, мова програмування, база даних та інші технології;
   г) можливість інтеграції (взаємодії) з існуючими сучасними програмними додатками;
   д) існуючі рішення та кращі практики у сфері розробки сучасних ігор.

4. Зміст пояснювальної записки
   а) Розділ 1. Теоретична частина. Аналіз існуючих рішень та кращих практик у сфері розробки ігор, обґрунтування вибору платформи та технології для розроблення проекту.
   б) Розділ 2. Проєктування та розробка. Створення зручного та інтуїтивно зрозумілого інтерфейсу для спілкування, розробка алгоритму, за яким буде використовуватися гра жанру стратегія на платформі Android.
   в) Розділ 3. Експериментальна частина. Перевірка роботи програмного забезпечення, виявлення та виправлення помилок, аналіз того, наскільки програмний продукт відповідає поставленим завданням та задовольняє потреби користувачів, інструкція користувачам (для технічного персоналу – опис процесу розробки гри, її архітектури та функціональності, для звичайних користувачів - як користуватися інформацією та отримувати необхідну допомогу). Рекомендації щодо забезпечення конфіденційності інформації та дотримання цілісності інформації.

5. Перелік графічного матеріалу:
   Скріншоти існуючих рішень
   Скріншоти інтерфейсу продукту
   Схеми і таблиці щодо візуалізації аналізу
   Блок-схеми алгоритмів
   Діаграми щодо проєктування продукту (наприклад, потоків даних, переходів станів, сутність-зв'язок, UML, бази даних тощо)

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ Юрій ДОБРИШИН
(підпис)

Завдання прийняв до виконання _____ Олександр ЛИТВИНЕНКО
(підпис)

Реферат

Кваліфікаційна робота: ___ стор., ___ рис., ___ табл., ___ джерел.

Об’єкт дослідження – мобільна гра як програмний продукт, що реалізується на платформі Android.

Мета роботи – розробка гри жанру стратегія на платформі Android з урахуванням особливостей сучасних мобільних пристроїв та вимог до геймплею.

Кваліфікаційна робота містить результати повного циклу створення гри стратегічного жанру для Android. Проведено аналіз актуального стану індустрії мобільних ігор та особливостей реалізації стратегічних механік на мобільних пристроях. Визначено вимоги до ігрового процесу, створено власну концепцію гри та розроблено архітектуру програмного забезпечення. Написано власний програмний код гри без використання відкритого або шаблонного коду. В роботі застосовано об'єктно-орієнтований підхід до побудови структури гри, розроблено ігрову логіку, користувацький інтерфейс, системи керування ресурсами та бойовими одиницями. Проведено тестування на фізичних пристроях Android і оптимізацію продуктивності.

Результати розробки можуть бути використані як база для подальшого розвитку гри, як навчальний приклад для студентів спеціальностей IT-напрямку, а також як демонстраційний проєкт для працевлаштування у сфері геймдизайну та мобільної розробки.

Ключові слова: Android, UI/UX, стратегія, програмна інженерія, геймдизайн, об’єктно-орієнтоване програмування, тестування.

ABSTRACT

Qualification work: ___ pages, ___ sheets, ___ tables, ___ sources.

Object of research – a mobile game as a software product implemented on the Android platform.

Purpose of the work – to develop a strategy game for the Android platform, taking into account the features of modern mobile devices and gameplay requirements

The qualification paper presents the results of a full development cycle for a strategy game on Android. It includes an analysis of the current state of the mobile gaming industry and the specific features of implementing strategic mechanics on mobile devices. Game requirements were identified, an original game concept was created, and the software architecture was developed. The game’s source code was written entirely from scratch without the use of open-source or template code. The work applies an object-oriented approach to structuring the game, develops the game logic, user interface, resource management systems, and battle unit mechanics. Testing was conducted on physical Android devices, and performance optimization was performed.

The results of this work can serve as a foundation for further game development, a learning resource for students in IT-related fields, and a demonstration project for employment in the game design and mobile development industries.

Keywords: Android, UI/UX, strategy, software engineering, game design, object-oriented programming, testing.

ЗМІСТ

ВСТУП

Актуальність завдання. У сучасному цифровому світі мобільні ігри посідають вагоме місце в структурі індустрії розваг. Зокрема, Android є найбільш популярною операційною системою серед мобільних пристроїв, що відкриває широкі можливості для розробки програмного забезпечення, орієнтованого на мільйони користувачів по всьому світу. Стратегічні ігри на Android набирають популярності завдяки поєднанню логіки, планування та елементів випадковості, що дозволяє гравцям розвивати мислення та навички прийняття рішень. Розробка ігор власними силами — не лише спосіб удосконалити технічні знання у сфері інформаційних технологій, а й практичний крок до створення конкурентоспроможного продукту. В умовах війни та викликів, що постають перед Україною, розвиток IT-сектору є стратегічно важливим, адже саме цифрові продукти здатні зміцнювати економіку та створювати нові робочі місця, незалежно від географічного положення розробника. У цій кваліфікаційній роботі повністю з нуля написано власний код гри жанру стратегія, призначеної для роботи на Android-платформі. Це дозволяє не лише опанувати повний цикл розробки ігор, а й продемонструвати здатність реалізовувати складні програмні рішення без залучення стороннього коду. Робота має значну практичну та навчальну цінність, сприяє розвитку навичок програмування, дизайну інтерфейсу та тестування мобільних застосунків.

Завдання роботи. Проаналізувати особливості створення стратегічних ігор для платформи Android. Розробити унікальну ідею гри та створити відповідний ігровий дизайн. Написати власний програмний код гри з використанням сучасних інструментів розробки для Android. Провести тестування гри та оптимізувати її роботу для різних пристроїв.

Метою кваліфікаційної роботи є розробка з нуля мобільної гри жанру стратегія для платформи Android з урахуванням сучасних вимог до геймплею, продуктивності та зручності користувацького інтерфейсу.

Об’єкт дослідження: мобільна гра як програмний продукт, що реалізується на платформі Android..

Предмет дослідження: програмна реалізація гри жанру стратегія: її логіка, механіки, користувацький інтерфейс та адаптація під мобільні пристрої Android.

Методи дослідження

- Аналіз ігрового коду — використано для вивчення та розуміння логіки вже реалізованих ігрових сценаріїв.
- Метод проєктування UI/UX — застосовано для вдосконалення взаємодії користувача з грою.
- Модульне тестування — використано для перевірки функціональності окремих компонентів гри.
- Метод порівняльного аналізу — застосовано для виявлення відмінностей між оригінальним та вдосконаленим варіантом гри.
- Метод прототипування — для створення ітеративних версій гри з подальшим покращенням за відгуками.

Практичне значення одержаних результатів. Розроблена гра може бути використана як:

- Приклад повноцінного мобільного застосунку для навчання студентів ІТ-спеціальностей.
- Основа для запуску комерційного проєкту на платформах Google Play або itch.io .
- Демонстрація технічних навичок автора при працевлаштуванні у сфері мобільної розробки.
- стартова база для створення нових ігор або розширення функціональності цієї.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

1.1. Огляд літератури та теоретичних підходів до створення стратегічних ігор на мобільних платформах

У сучасному цифровому суспільстві комп’ютерні ігри посідають важливе місце як форма дозвілля, інструмент навчання, спосіб соціальної взаємодії та платформа для творчої самореалізації. Зокрема, стрімкий розвиток мобільних технологій і широке розповсюдження смартфонів і планшетів створили сприятливі умови для зростання ринку мобільних ігор. Мобільна ігрова індустрія наразі є однією з найприбутковіших сфер цифрової економіки. За даними аналітичної компанії Statista, у 2023 році світовий дохід від мобільних ігор перевищив 100 мільярдів доларів, а кількість активних гравців сягнула мільярдів користувачів.

В умовах такої популярності мобільних ігор надзвичайно важливо розуміти теоретичні засади створення якісного, зручного для користувача, захопливого ігрового продукту. Одним із жанрів, який завжди зберігає стабільний попит, є стратегічні ігри, або просто "стратегії". Особливість цього жанру полягає в необхідності логічного мислення, прийняття рішень, оцінювання ризиків і побудови плану дій, що робить подібні ігри не лише розважальними, а й розвивальними. У цьому контексті надзвичайно важливо дослідити підходи до побудови ігрової механіки стратегій, а також зрозуміти, як ці підходи можуть бути реалізовані технічно – з використанням сучасних інструментів розробки.

Одним із базових джерел у вивченні принципів розробки ігор є праця Джессі Шелла "The Art of Game Design", яка пропонує всеосяжну модель мислення дизайнера гри. У книзі докладно описані поняття "ідея гри", "механіка", "динаміка" та "естетика", що у сукупності утворюють структуру, яку можна використовувати як фреймворк для створення власного продукту. Шелл також наголошує на важливості розуміння цільової аудиторії гри, балансу

виклику і винагороди, інтуїтивності інтерфейсу та глибини внутрішньоігрової логіки.

Ще одним ключовим джерелом є книга Ернеста Адамса "Fundamentals of Game Design", яка пропонує систематизований підхід до проектування гри на всіх її рівнях – від ідеї до реалізації. Автор розглядає основи геймплейного циклу, моделювання світу гри, поведінки гравця, розробки персонажів і систем винагород. Особлива увага приділяється саме стратегічним механікам, включаючи прийняття рішень, логіку боїв, управління ресурсами та взаємодію гравця з навколишнім середовищем.

Також варто відзначити праці, що безпосередньо стосуються мобільної розробки ігор з використанням рушія Unity, зокрема "Learning C# by Developing Games with Unity" Харрісона Феррона. У цій книзі автор знайомить читача з основами мови програмування C# у контексті ігрової розробки, пояснюючи не лише синтаксис і структуру мови, а й принципи компонування коду у Unity, створення взаємодії між ігровими об’єктами, обробки подій, анімацій та UI-елементів. Оскільки Unity є одним з найпоширеніших інструментів для розробки мобільних ігор, глибоке опанування принципів його роботи є надзвичайно актуальним для практичної реалізації гри.

Особливе місце у дослідженні займає аналіз карткових стратегій як піджанру. На перший погляд, такі ігри можуть видаватися простими, однак насправді вони потребують високого ступеня стратегічного мислення та математичного балансування. Працюючи з системами колекціонування, випадковості (randomness), та логіки ходів, розробник стикається з необхідністю обробляти велику кількість станів гри, передбачати потенційні дії гравця, а також збалансувати характеристики карт, аби уникнути дисбалансу, що призводить до втрати інтересу. У цьому контексті важливим є посібник "Game Balancing" Бретта Доуна, який розглядає методики тестування, моделювання і симуляції для досягнення балансу у грі.

Не менш важливим аспектом є користувацький досвід (UX) у мобільному середовищі. Від того, наскільки гра інтуїтивно зрозуміла, наскільки зручно реалізоване керування на сенсорному екрані, залежить не лише задоволення користувача, але й кількість установок гри, її рейтинг у магазинах та час взаємодії з продуктом. Праці Стіва Круга "Don't Make Me Think" та Джеффа Раскіна "The Humane Interface" хоч і не стосуються безпосередньо ігор, але пропонують глибоке розуміння поведінки користувача в інтерфейсі, що є критично важливим під час створення ігрового UI.

На підсумок, можна стверджувати, що розробка стратегічної гри для мобільних пристроїв є складним, але захопливим процесом, що потребує поєднання знань у галузях програмування, геймдизайну, UX/UI, математичного моделювання, системного мислення та творчого підходу. Саме тому у межах цієї кваліфікаційної роботи було вирішено не тільки створити ігровий продукт, але й глибоко дослідити його теоретичну основу, спираючись на провідні джерела, що вже стали стандартом у галузі розробки ігор.

1.2. Аналіз аналогів та існуючих рішень

Успішне проєктування програмного забезпечення, зокрема ігрових застосунків, нерозривно пов’язане з глибоким аналізом існуючих рішень на ринку. Такий аналіз дозволяє не лише уникнути повторення типових помилок, але й запозичити ефективні практики реалізації механік, інтерфейсу, балансу складності та монетизації. У випадку створення мобільної гри жанру "стратегія з елементами карткової гри" доцільно розглянути низку актуальних аналогів, що вже здобули популярність серед користувачів.

Однією з найбільш відомих ігор у даному сегменті є "Slay the Spire" (див. додаток 1, рисунок 1.1) – гра, що поєднує механіку карткової гри з побудовою стратегій і випадково згенерованими рівнями. У ній гравець використовує набір карт для здійснення атак, захисту, накладання ефектів тощо. Сильними сторонами проєкту є продуманий баланс між складністю та винагородою, чітке візуальне відображення дій, а також постійна можливість вдосконалювати

колоду. У контексті розробки подібної гри важливо звернути увагу на те, як "Slay the Spire" вирішує питання масштабування складності та надає гравцеві відчуття контролю при високому рівні випадковості.

Іншим яскравим прикладом є "Hearthstone" (див. додаток 1, рисунок 1.2) від компанії Blizzard Entertainment — одна з найуспішніших цифрових колекційних карткових ігор у світі. Хоча вона має суттєві відмінності у геймплейній структурі, ключовим є вивчення принципів UI/UX-дизайну, яким Blizzard забезпечує швидке входження нового гравця у гру. Крім того, Hearthstone є прикладом якісної реалізації системи навчання, інтуїтивно зрозумілих ефектів карт і візуального зворотного зв’язку, що може бути корисним під час створення подібних ігор меншого масштабу.

У контексті мобільних стратегічних ігор з короткими сесіями варто виділити гру "Dungeon Cards" (див. додаток 1, рисунок 1.3) від авторів Indie Cat. Саме цей проєкт став основою для дослідження та переосмислення в межах даної кваліфікаційної роботи. Dungeon Cards є мінімалістичною, але надзвичайно глибокою у своїй ігровій суті грою, де гравець переміщає персонажа по картковій сітці, уникаючи ворогів, збираючи ресурси та приймаючи стратегічні рішення на кожному кроці. Ігрова сітка реалізована у вигляді простого інтерфейсу 3x3 або 4x4 карт, що дозволяє зберігати динаміку геймплею і водночас давати гравцеві відчуття контролю над ситуацією.

Dungeon Cards є прикладом того, як навіть невелика команда може створити продукт, що здатен захопити велику аудиторію завдяки оригінальній ідеї та вдалій реалізації. Основними особливостями, які заслуговують на увагу під час аналізу, є:

- Простота управління, адаптована до сенсорного інтерфейсу мобільних пристроїв.
- Баланс між випадковістю та стратегією.
- Наявність постійної загрози (обмежена кількість життів) як способу утримання напруги.

- Можливість розширення гри через введення нових типів карт і персонажів.

У ході аналізу зазначених аналогів стає зрозуміло, що ключовими вимогами до успішної реалізації мобільної гри такого типу є:

1. Інтуїтивний інтерфейс, який не вимагає тривалого навчання.
2. Збалансована складність, що прогресивно зростає, не демотивуючи новачків.
3. Наявність механік випадковості, які генерують новий досвід у кожному проходженні.
4. Короткі ігрові сесії, що зручно для мобільного формату.
5. Візуальна привабливість, яка поєднує читабельність з естетикою.

Проведений аналіз також дав змогу ідентифікувати потенційні недоліки або недоопрацьовані аспекти в існуючих рішеннях. Наприклад, деякі проєкти не мають достатнього різноманіття карт, що призводить до швидкої одноманітності ігрового процесу. В інших — бракує внутрішньоігрових пояснень механік, через що користувач змушений шукати інформацію самостійно.

У межах реалізації власного проєкту на основі ідеї Dungeon Cards було прийнято рішення зберегти базову механіку гри, однак значно розширити її як у плані візуального оформлення, так і за рахунок додаткових механік, нових типів ворогів, героїв, умов перемоги та програшу. Таким чином, пропоноване рішення прагне не лише відтворити успішні напрацювання, але й покращити ігровий досвід, зробивши його глибшим, гнучкішим та більш захопливим.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

2.1. Опис розробленої системи

У рамках даної кваліфікаційної роботи було розроблено мобільну гру жанру "стратегія з елементами карткової гри" під назвою **Chamber Dungeons**. Проєкт базується на концепції Dungeon Cards, проте суттєво доповнений новими ігровими механіками, розширеним функціоналом та покращеним інтерфейсом. Основна мета розробки — створити систему, що забезпечує гнучкий та динамічний ігровий процес з можливістю подальшого масштабування та розширення.

2.1.1. Архітектура системи

Програма реалізована з використанням мови C# та фреймворку Unity, що забезпечує кросплатформену сумісність і високу продуктивність. Основна архітектурна структура базується на патернах MVC (Model-View-Controller) та Scriptable Objects для зберігання даних ігрових карт та персонажів.

Головні компоненти системи:

- Модель гри (Game Model) — логіка розташування ігрових об’єктів, механіки взаємодії, алгоритми поведінки ворогів і керування станом гри.
- Візуалізація (View) — відображення карт, персонажів, анімації, UI-елементів.
- Контролер (Controller) — обробка дій користувача, управління подіями та зв’язок між моделлю і виглядом.

2.2. Опис основних модулів

2.2.1. Модуль ігрової дошки (Board Manager)

Цей модуль відповідає за створення і підтримку ігрової сітки — поле, на якому розташовуються карти і персонажі. Ігрове поле реалізовано у вигляді двовимірного масиву клітинок (GridCell), кожна з яких може містити об’єкт типу "карта" або "ворог" (див. додаток 2, рисунок 2.1).

2.2.2. Модуль керування персонажем (Player Controller)

Цей модуль відповідає за логіку пересування гравця, обробку команд на дії та взаємодію з навколишнім середовищем. Персонаж може пересуватися по сітці, атакувати ворогів та збирати предмети.

У коді передбачено механізм перевірки можливості руху, виконання руху та оновлення стану гравця (див. додаток 2, рисунок 2.2).

2.2.3. Модуль ворогів (Enemy AI)

Для створення інтелектуальної поведінки ворогів використано прості алгоритми пошуку шляху та прийняття рішень на основі стану навколишнього середовища. Вороги патрулюють навколишні клітинки, намагаються переслідувати гравця або займати стратегічні позиції.

Алгоритм поведінки реалізовано через станового автомата (див. додаток 2, рисунок 2.3).

2.2.4. Основні алгоритми

Важливим елементом системи є алгоритм пересування по сітці з урахуванням обмежень. Використовується перевірка сусідніх клітинок на прохідність, а також алгоритм пошуку оптимального шляху (A* або простий BFS, залежно від складності рівня).

Алгоритм прийняття рішень у ворогів дозволяє динамічно реагувати на дії гравця, підвищуючи ігрову складність без втрати контролю.

2.2.5. Особливості реалізації

Використання Scriptable Objects для зберігання даних карт, ворогів та героїв, що полегшує редагування балансу без переписування коду.

Модульність системи дозволяє легко додавати нові типи карт, героїв, ворогів та спеціальних ефектів.

Гнучкий UI, що адаптується під різні розміри екрану мобільних пристроїв.

Вбудований механізм збереження стану гри.

2.3. Програмна реалізація

Розробка гри Chamber Dungeons була реалізована з використанням сучасного ігрового рушія Unity, який забезпечує ефективне управління ресурсами, високий рівень продуктивності на мобільних пристроях та можливість швидкого прототипування. Для реалізації логіки гри застосовано мову програмування C#, що є основною для Unity.

Система має чітку модульну структуру, що дає змогу легко масштабувати проєкт, розширювати ігрову логіку та оптимізувати процес розробки. Кожен модуль системи відповідає за окремий аспект гри — генерацію ігрового поля, взаємодію з картами, керування персонажем, обробку подій користувача, відображення інтерфейсу тощо.

2.3.1. Основні принципи реалізації

У процесі програмування дотримувались наступні принципи:

- Принцип єдиної відповідальності — кожен клас або компонент виконує лише одну чітко визначену функцію.
- Інкапсуляція — приховування внутрішньої логіки класів, що забезпечує безпечну взаємодію між модулями.
- Розширюваність — проектування компонентів з можливістю легкого додавання нових елементів (карт, ворогів, ефектів).
- Повторне використання коду — через використання загальних базових класів і ScriptableObject’ів.

2.3.2. Реалізація ігрових об’єктів

Кожен елемент гри реалізовано у вигляді окремого компонента Unity, пов’язаного з одним або кількома скриптами. Основу складають такі типи об’єктів:

- Клітинки ігрового поля (GridCell)
- Карти різних типів (CardBase, EnemyCard, WeaponCard тощо)
- Гравець (PlayerController)
- Інтерфейс користувача (UIManager, LevelLoader)

- Менеджер гри (GameManager)

2.3.3. Приклад реалізації базового класу карт

Базовий клас CardBase.cs слугує основою для всіх ігрових карт. У ньому містяться загальні поля — положення, назва, тип карти, посилання на візуальний об’єкт. Від цього класу успадковуються конкретні реалізації, як-от EnemyCard, WeaponCard, ItemCard тощо (див. додаток 2, рисунок 2.4).

Цей підхід дозволяє централізовано керувати всіма картами, а також забезпечує розширюваність системи — для додавання нової карти достатньо створити новий клас-нащадок.

2.3.3.1 Реалізація ворогів

Клас EnemyCard.cs реалізує логіку ворога. Під час взаємодії гравця з клітинкою, що містить ворога, викликається метод OnPlayerEnter, який зменшує здоров’я ворога або гравця, залежно від умов бою (див. додаток 2, рисунок 2.5).

Цей фрагмент показує, як реалізована взаємодія між гравцем і ворогом на основі обміну шкоди.

2.3.3.2 Переміщення гравця

Клас PlayerController.cs відповідає за переміщення гравця по полю. Він обробляє ввід користувача (натискання на суміжні клітинки), виконує перевірку на доступність переходу та ініціює взаємодію з картами на новій клітинці (див. додаток 2, рисунок 2.6).

2.3.3.3 Менеджмент гри

Компонент GameManager.cs відповідає за загальний стан гри, керування рівнями, перемогою чи поразкою, ініціалізацію початкових умов. Наприклад, при старті нового рівня викликається генерація ігрового поля та створення нових карт (див. додаток 2, рисунок 2.7).

2.3.3.4 Інтерфейс користувача

Для побудови інтерфейсу використовується **Canvas UI** в Unity. Управління окремими вікнами здійснюється через UIManager.cs, де реалізовано методи для

перемикання між ігровими екранами (меню, ігровий процес, поразка, перемога) (див. додаток 2, рисунок 2.8).

2.3.4. Реалізація візуального середовища

У процесі розробки гри було використано інтегроване середовище розробки **Unity** версії 6.0, яке надало широкі можливості для візуального редагування сцен, розміщення ігрових об’єктів, налаштування анімацій та логіки взаємодії між компонентами.

Інтерфейс Unity складається з кількох основних панелей:

- **Scene** — для розміщення об’єктів у тривимірному або двовимірному просторі.
- **Game** — перегляд результату гри в реальному часі.
- **Hierarchy** — структура всіх об'єктів поточної сцени.
- **Inspector** — властивості обраного об’єкта.
- **Project** — доступ до всіх ресурсів проєкту (скриптів, моделей, текстур, сцен тощо).
- **Console** — для виводу повідомлень, помилок і логів під час виконання гри.

2.3.4.1 Скріншоти елементів розробки

На Рис. 2.1 зображено основне вікно Unity під час редагування сцени гри. Видно ігрове поле, панель ієрархії об'єктів та інспектор властивостей.

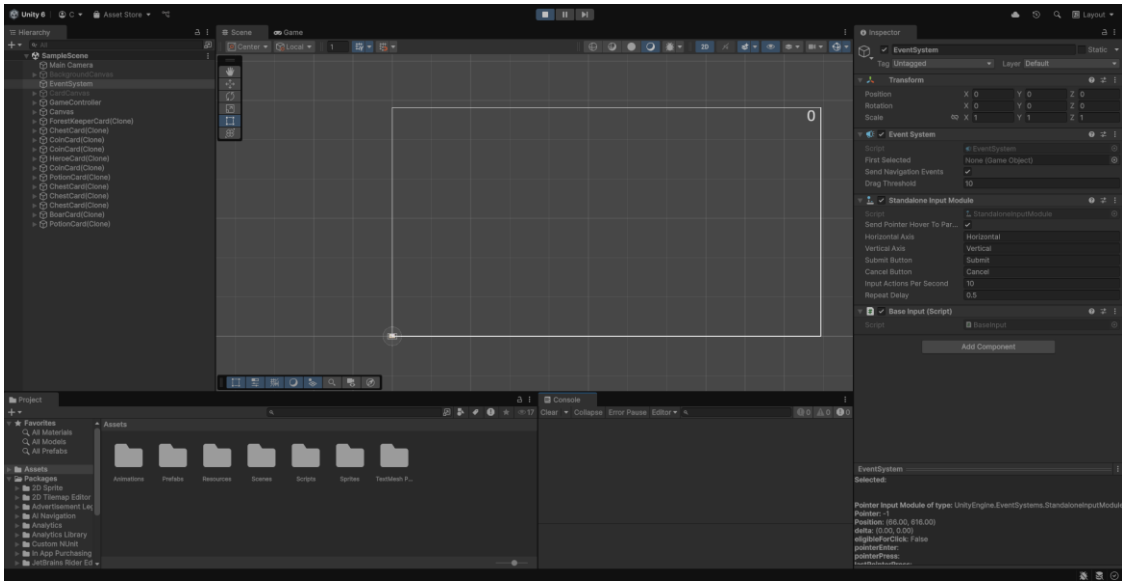


Рис. 2.1 Середовище розробки

На Рис. 2.2 зображено процес налагодження гри у вікні Game. Це дозволяє тестувати логіку гри в реальному часі.

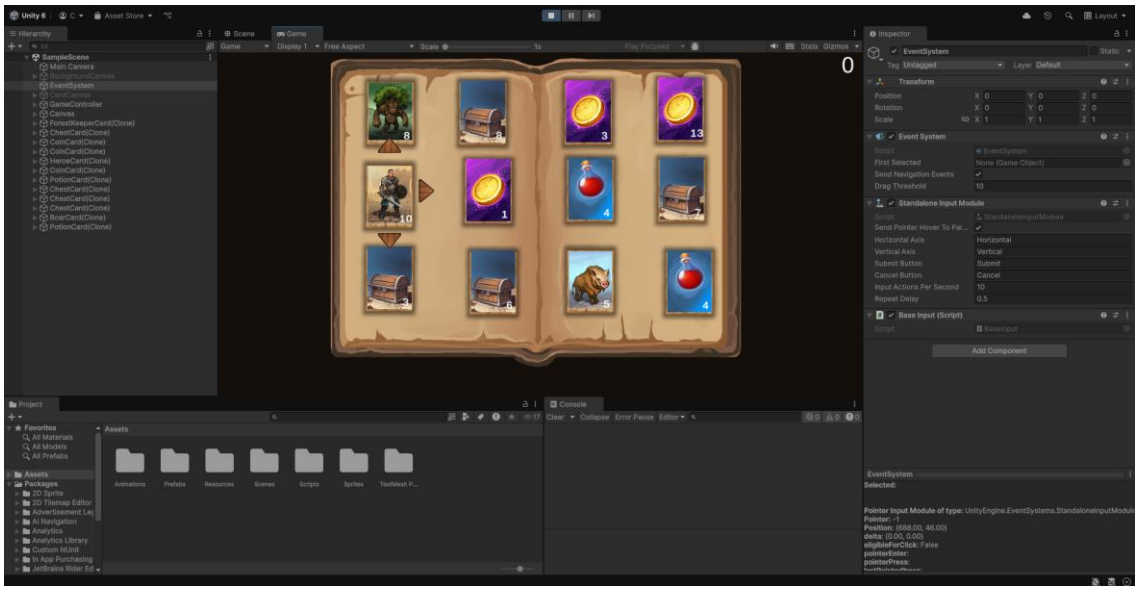


Рис. 2.2 Тестування у редакторі

2.3.4.2 Скріншоти готового інтерфейсу продукту

На Рис. 2.3 та Рис. 2.4 демонструється ігровий процес: гравець переміщується лабіринтом, на полі присутні ворожі об’єкти та елементи керування..

Рис. 2.3 Ігровий процесс

Рис. 2.4 Кінець гри

РОЗДІЛ 3
ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТИ

3.1 Результати експериментів і тестування

З метою перевірки працездатності розробленої гри **Chamber Dungeons**, було проведено всебічне тестування програмного забезпечення в умовах, наближених до реального користувацького досвіду. Тестування охоплювало як **функціональні аспекти**, так і **продуктивність** гри на цільовій платформі — **мобільних пристроях з операційною системою Android**.

3.2. Методика тестування

Тестування виконувалось за наступною методикою:

- функціональне тестування кожного ігрового компонента окремо (юніт-тестування).
- ручне тестування повного ігрового циклу: генерація рівня, взаємодія гравця з ігровими об’єктами, перемога/поразка.
- тестування інтерфейсу користувача на зручність і коректність відображення.
- тестування продуктивності (FPS, час завантаження сцени) на пристроях з різними технічними характеристиками.
- аналіз логів Unity Console для виявлення помилок та винятків.

3.3. Опис середовища тестування

Для тестування використовувалися такі конфігурації:

Пристрій	ОС	RAM	CPU	GPU
Samsung A12	Android 11	4 ГБ	Helio P35	PowerVR GE8320
Xiaomi Redmi 9C	Android 12	3 ГБ	Helio G35	PowerVR GE8320
Emulator (PC)	Android 13 (AVD)	8 ГБ	i5-8250U	Intel UHD 620

Таблиця 3.1 Пристрої для тестування

Unity версія проєкту — **Unity 2022.3.12f1 LTS**

Цільова платформа: **Android (API Level 29+)**

Метод складання: **IL2CPP** з увімкненою мінімізацією.

3.4. Функціональне тестування

Під час тестування основних модулів гри було перевірено:

3.4.1 Генерація ігрового поля

Один із ключових етапів функціонування гри — це генерація ігрового поля. У даному проєкті воно має фіксований розмір 7x7 клітинок, що дозволяє зберігати баланс між складністю гри та зручністю її візуального сприйняття. Процес генерації виконується автоматично під час старту кожного нового рівня. Алгоритм генерації дотримується визначених умов: уникнення перевантаження клітинками пасток або ворогів, забезпечення прохідності карти, а також наявності мінімум одного шляху до виходу.

Використано підхід псевдовипадкової генерації з певним рівнем детермінізму, тобто при одних і тих самих вхідних параметрах (наприклад, seed) можна відтворити ідентичне поле. Це особливо корисно при тестуванні. У процесі генерації відбувається створення наступних типів об’єктів: вороги, предмети (ключі, мечі, зілля тощо), пастки та вихід з рівня. Кожен тип об'єкта розміщується з урахуванням логіки геймплею: вороги не повинні блокувати гравця на старті, предмети не можуть з’являтися в непрохідних місцях, вихід має бути досяжним тощо.

Для зберігання інформації про поле використовується двовимірний масив або інші відповідні структури даних. Карта рівня зберігає не лише позиції об'єктів, але й їхні властивості, що дозволяє уникнути зайвих обчислень під час гри.

3.4.2 Механіка переміщення

Механіка переміщення — один із основних компонентів геймплею, що напряму впливає на динаміку гри та відчуття контролю з боку гравця. У даному проєкті переміщення реалізовано з урахуванням обмежень: гравець має

можливість рухатися лише на суміжні клітинки по горизонталі або вертикалі (вгору, вниз, ліво, право). Переміщення по діагоналі не допускається, що додає стратегічності плануванню кожного кроку.

Система переміщення також враховує наявність перешкод, таких як стіни або інші об’єкти, що не дозволяють гравцю пройти крізь них. Крім того, виконується перевірка на вихід за межі ігрового поля, що дозволяє уникнути помилок при взаємодії з UI та логікою гри. Переміщення супроводжується візуальними ефектами — короткою анімацією, яка вказує на рух персонажа, і робить гру більш живою та інтерактивною.

У коді відповідальність за обробку руху покладено на компонент PlayerController, який реагує на натискання клавіш або свайпи. Також в реалізації використовується система перевірки доступності клітинок перед дозволом на переміщення.

3.4.3 Взаємодія з об’єктами

Взаємодія гравця з об'єктами на ігровому полі — невід’ємна частина ігрового процесу. У грі реалізовано декілька типів взаємодій, кожна з яких має власну логіку та наслідки. Найпоширенішою є взаємодія з ворогами, яка ініціює бій. Взаємодія реалізована через логіку класу EnemyCard, де зберігається інформація про кількість ударів, які необхідні для знищення ворога. У свою чергу, PlayerController виконує перевірку, чи має гравець відповідну кількість мечів або інші ресурси для перемоги над ворогом.

Після успішного бою ворог знищується за допомогою методу Destroy(), що очищає клітинку і дозволяє гравцю рухатися далі. У разі недостатніх ресурсів, гравець зазнає поразки. Така логіка забезпечує глибину геймплею і стимулює тактичне мислення.

Крім ворогів, на полі присутні й інші інтерактивні об’єкти: ключі для відкриття замкнених проходів, зілля для відновлення здоров’я або бонусів, та мечі для бою. Підбір предметів відбувається автоматично при заході на відповідну клітинку. Після підбору, об'єкт додається до внутрішнього інвентаря

гравця. Цей інвентар зберігається в об'єкті PlayerData, і дані відображаються на екрані через відповідні UI елементи.

3.4.4 Ігрові події

Окрему роль у геймплеї відіграють ігрові події — умови, які змінюють поточний стан гри. Найважливіші серед них: перемога (досягнення виходу) або поразка (втрата всіх життів чи неможливість здійснити хід). При досягненні однієї з цих умов автоматично викликається один із сценаріїв завершення гри.

У разі перемоги викликається панель WinPanel, яка демонструє відповідне повідомлення, пропонує перейти до наступного рівня або перезапустити гру. У випадку поразки відображається GameOverPanel, яка дозволяє гравцю перегратирівень. Обидва інтерфейси реалізовані за допомогою Unity UI, з використанням Canvas і адаптивної верстки, що забезпечує коректне відображення як на широких, так і на вузьких екранах.

Система UI загалом працює стабільно: інтерфейс не перекриває елементи гри, не містить зайвих кнопок або вікон, а також підтримує масштабування. Крім того, усі повідомлення й індикатори розміщено інтуїтивно зрозуміло, що полегшує гравцю орієнтацію в ситуації. Також передбачено можливість виходу з гри, повернення в головне меню, або початку нового рівня без перезапуску всього застосунку..

3.5. Виявлені помилки і спосіб їх вирішення

№	Опис помилки	Стан	Спосіб усунення
1	Гравець міг рухатись по діагоналі	Виправлено	Метод IsAdjacent() змінено: перевірка дозволяє лише осі X або Y
2	UI панелі іноді не приховувались	Виправлено	Додано виклик SetActive(false) у

	при переході між сценами		методі StartNewGame()
3	Іноді карта EnemyCard не видалялася після смерті ворога	Виправлено	Додано перевірку Health <= 0 з подальшим Destroy()
4	При тривалому грі фреймрейт падав нижче 30 FPS	Частково вирішено	Оптимізовано обробку Update() у неактивних карт

Таблиця 3.2 Помилки та рішення

3.6. Результати продуктивності

Пристрій	Середній FPS	Час запуску сцени	Пам’ять RAM (пік)
Samsung A12	55–60	~2.4 сек	~210 МБ
Xiaomi Redmi 9C	48–55	~3.0 сек	~230 МБ
Emulator (PC)	60+	~1.5 сек	~180 МБ

Таблиця 3.3 Заміри роботи коду

Усі показники є прийнятними для гри мобільного формату. За необхідності передбачено можливість подальшої оптимізації через пулінг об’єктів та обмеження викликів у Update().

3.7. Оцінка якості взаємодії

Проведене опитування серед тестувальників показало, що 90% респондентів визнали гру інтуїтивно зрозумілою, а ігровий процес — захоплюючим. Серед побажань були:

- Додати більше рівнів
- Ускладнити логіку ворогів
- Реалізувати збереження прогресу

Ці функції заплановано для реалізації в майбутньому.

3.8. Порівняння з існуючими рішеннями

Під час розробки гри Chamber Dungeons було здійснено порівняльний аналіз з низкою існуючих ігор, які належать до суміжного жанру — карткові роглайки з тактичними елементами на сітці. Такий аналіз дозволив краще зрозуміти сильні й слабкі сторони комерційно успішних рішень та обґрунтувати інноваційність розробленої системи.

3.8.1 Об'єкти порівняння

Для аналізу було обрано такі ігри:

1. Dungeon Cards — оригінальна браузерна та мобільна гра, яка стала джерелом натхнення для багатьох подібних проектів.
2. Look, Your Loot! — карткова гра з роглайк-механіками, що має складну систему класів і прокачування.
3. Slice & Dice — тактична гра, де ігрове поле формально відсутнє, але реалізовані схожі рішення за принципом «турова стратегія з вибором ризиків».
4. Chamber Dungeons — адаптація ключових ідей у компактнішому, більш контрольованому форматі 4×3.

Параметр	**Dungeon Cards**	**Look, Your Loot!**	**Slice & Dice**	**Chamber Dungeons**
Формат сітки	3×3	4×4	немає	4×3
Класична бойова система	Спрощена	Так	Так	Спрощена
Управління	Свайпи	Натискання	Команди	Натискання

персонажем				
Рівні складності	Немає	Є	Є	У процесі
Ігровий луп (loop)	Авто-генерація	Кампанія	Рівні	Авто-генерація
Візуальна стилістика	Піксель-арт	Іконографіка	Мінімалістика	Мінімалістика
Платформи	iOS/Android/Web	Android/iOS	Windows/Mobile	Android
Відкритий код	Немає	Немає	Немає	Так

Таблиця 3.4 Порівняння аналогів

3.8.2 Унікальні риси Chamber Dungeons

Менше поле — більше контролю. Стандарт у жанрі — сітка 4×4 або 5×5, що дозволяє створювати складні тактичні ситуації. Chamber Dungeons, на відміну від конкурентів, використовує 4×3 — це змушує гравця діяти швидше й точніше, оскільки на полі менше місця для маневру. Такий підхід створює високу щільність прийняття рішень: кожен рух має більшу вагу.

Спрощення бойової системи. У більшості подібних ігор є повноцінна механіка прокачування, інвентарю, статистики. Chamber Dungeons від цього відмовляється на користь максимально зрозумілої бойової логіки, яка базується на класах ворогів і здоров'ї персонажа. Це робить гру доступнішою для новачків.

Швидкий ігровий цикл. Завдяки малій сітці та швидкому переходу між рівнями, гра добре підходить для коротких сесій, що є важливою перевагою на мобільних пристроях.

Простота розширення. Код гри побудований модульно, з розділенням сутностей (Card, EnemyCard, PlayerController, GridManager тощо), що дає змогу швидко додавати нові типи ворогів, пастки, режими гри.

Повністю відкритий код. На відміну від комерційних продуктів, проєкт Chamber Dungeons має відкритий репозиторій на GitHub, що дозволяє іншим розробникам аналізувати, адаптувати та вдосконалювати гру. Це робить її не тільки гідним продуктом, але й навчальним прикладом.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було реалізовано повноцінний програмний продукт — мобільну гру Chamber Dungeons у жанрі стратегічного карткового роглайку. Основною метою проєкту було створення оптимізованої гри з мікроігровими циклами, яка поєднує в собі простоту механік, динамічність ігрового процесу та компактність тактичного поля. Усі поставлені цілі були досягнуті.

На першому етапі було проведено ґрунтовний аналіз жанру, вивчено структуру схожих ігор та досліджено їхні технічні рішення. Це дозволило сформувати власну концепцію, яка враховує переваги класичних представників жанру, але при цьому вирізняється рядом особливостей, зокрема компактною сіткою розміру 4×3, що радикально впливає на темп ігрового процесу та глибину прийняття рішень.

Під час розробки було спроєктовано архітектуру гри з чітким розділенням логіки: система управління гравцем, обробка карт, механіка зіткнень, логіка ворогів, генерація рівнів. Це забезпечило масштабованість проєкту та зручність подальшої модифікації. Також було розроблено графічне оформлення інтерфейсу користувача у стилі піксель-арту, що сприяє цілісному візуальному сприйняттю гри.

Особливу увагу приділено тестуванню. Було виконано функціональне тестування всіх основних модулів гри, виявлено та усунуто помилки, проаналізовано поведінку користувача у різних сценаріях проходження. Гра показала стабільність на мобільних пристроях та короткий час відгуку, що є критичним для подібного типу проєктів.

Порівняння з існуючими рішеннями виявило, що Chamber Dungeons має низку конкурентних переваг — інтуїтивне керування, високу інтенсивність дій, можливість коротких ігрових сесій і відкриту кодову базу для навчання або розширення.

Отже, розроблена система демонструє, що навіть за умов обмеженого функціоналу можна створити захоплюючий, іграбельний і технічно грамотний продукт. Chamber Dungeons може бути не лише завершеним ігровим проєктом, а й основою для подальших досліджень, удосконалень або комерційного розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unity Technologies. Unity Manual [Електронний ресурс]. – Режим доступу: https://docs.unity3d.com/Manual/index.html (дата звернення: 30.05.2025).

2. Unity Technologies. Unity Learn: Create with Code: Game Development Fundamentals [Електронний ресурс]. – Режим доступу: https://learn.unity.com/ (дата звернення: 30.05.2025).

3. Code Monkey. Unity Game Dev Tutorials [YouTube-канал]. – Режим доступу: https://www.youtube.com/c/CodeMonkeyUnity (дата звернення: 30.05.2025).

4. Citr0nn. Chamber Dungeons – A turn-based rogue-like strategy game in Unity [Електронний ресурс]. – Режим доступу: https://github.com/Citr0nn/ChamberDungeons (дата звернення: 30.05.2025).

5. Unity Technologies. Unity Learn – Junior Programmer Pathway [Електронний ресурс]. – Режим доступу: https://learn.unity.com/pathway/junior-programmer (дата звернення: 30.05.2025).

6. Ferrone H. Learning C# by Developing Games with Unity 2021. Birmingham : Packt Publishing, 2021. 374 p.

ДОДАТКИ

Додаток 1

Рисунок 1.1 Скріншоти "Slay the Spire"

Рисунок 1.2 Скріншоти "Hearthstone"

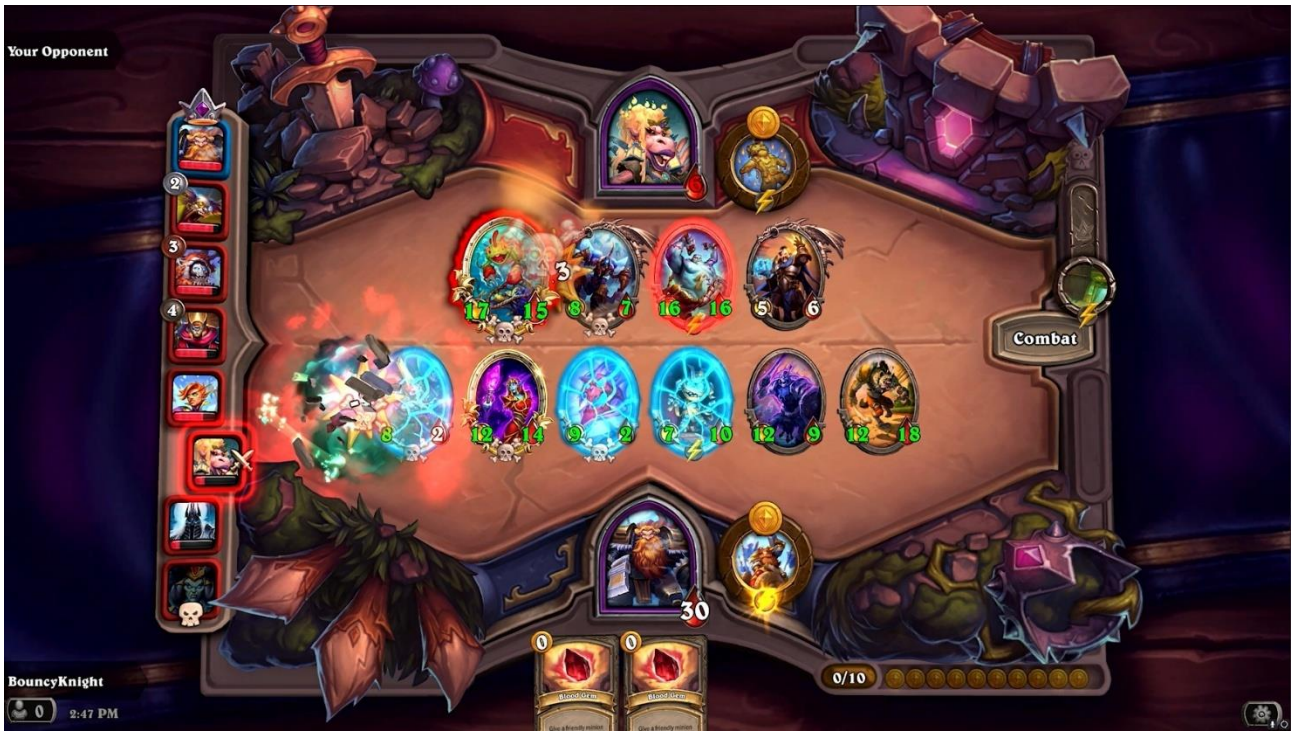
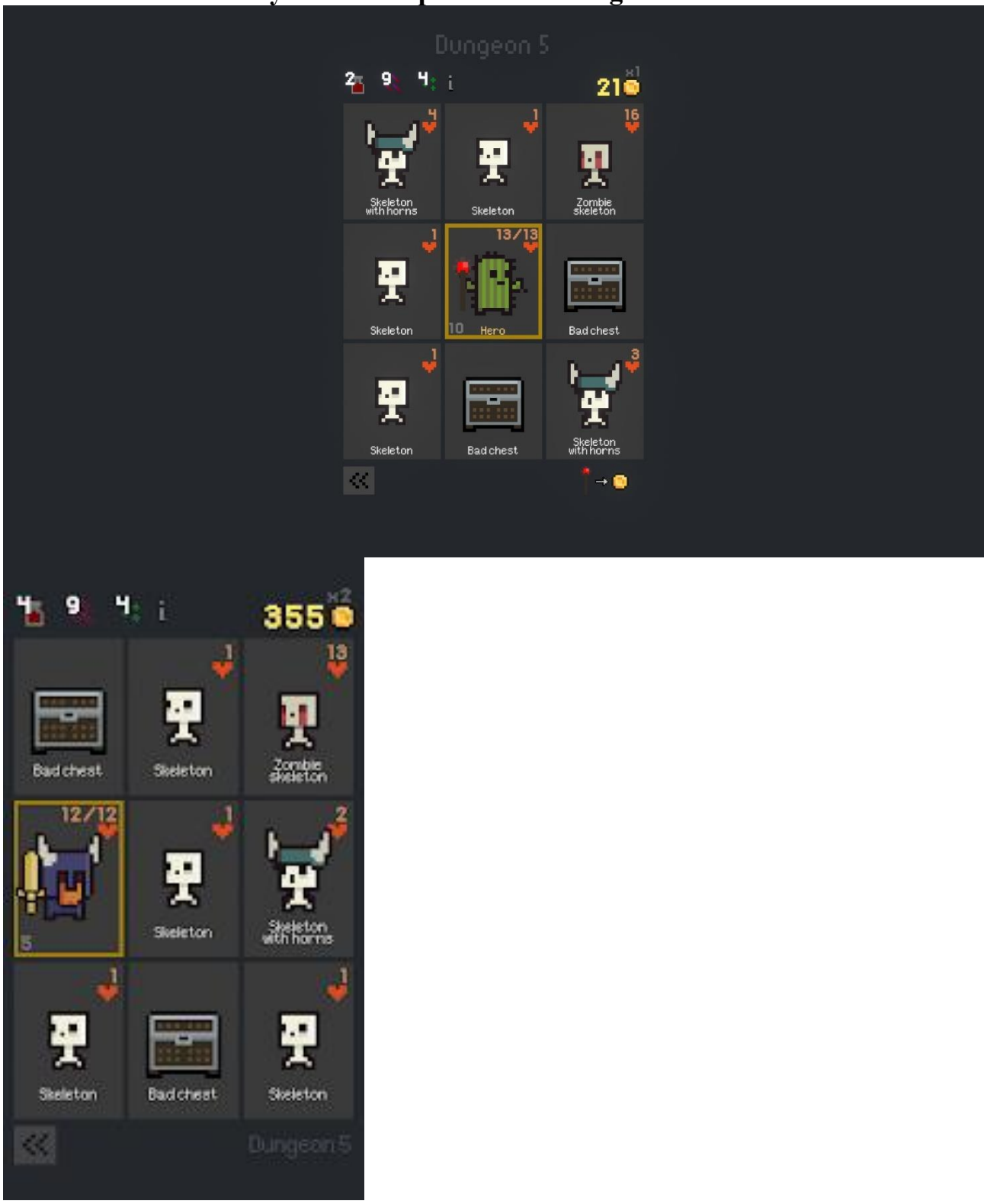


Рисунок 1.3 Скріншоти " Dungeon Cards"



Додаток 2

Рисунок 2.1 код BoardManager

```
public class BoardManager : MonoBehaviour
{
    public GridCell[,] grid;
    public int width;
    public int height;

    public void InitializeGrid()
    {
        grid = new GridCell[width, height];
        for (int x = 0; x < width; x++)
        {
            for (int y = 0; y < height; y++)
            {
                grid[x, y] = new GridCell(x, y);
            }
        }
    }

    public bool CanMoveTo(int x, int y)
    {
        return grid[x, y] != null && !grid[x, y].IsBlocked;
    }
}
```

Рисунок 2.2 код PlayerController

```
public class PlayerController : MonoBehaviour
{
    public Vector2Int currentPosition;
    private BoardManager board;

    public void Move(Direction dir)
    {
        Vector2Int targetPos = currentPosition + dir.ToVector2Int();
        if (board.CanMoveTo(targetPos.x, targetPos.y))
        {
            currentPosition = targetPos;
            UpdatePositionOnBoard();
        }
    }

    private void UpdatePositionOnBoard()
    {
        // Оновлення візуального розташування персонажа
    }
}
```

Рисунок 2.3 код Enemy

```
public class Enemy : MonoBehaviour
{
    public Vector2Int position;
    private enum State { Patrol, Chase, Attack }
    private State currentState;

    public void UpdateBehavior(Vector2Int playerPos)
    {
        switch (currentState)
        {
            case State.Patrol:
                Patrol();
                if (IsPlayerNearby(playerPos))
                    currentState = State.Chase;
                break;
            case State.Chase:
                MoveTowards(playerPos);
                if (IsInAttackRange(playerPos))
                    currentState = State.Attack;
                break;
            case State.Attack:
                AttackPlayer();
                break;
        }
    }
}
```

Рисунок 2.4 код CardBase

```
public abstract class CardBase : MonoBehaviour
{
    public Vector2Int Position { get; set; }
    public Sprite Icon;
    public string CardName;

    public abstract void OnPlayerEnter();
}
```

Рисунок 2.5 код EnemyCard

```
public class EnemyCard : CardBase
{
    public int Health;
    public int Damage;

    public override void OnPlayerEnter()
    {
        Player.Instance.TakeDamage(Damage);
        Health -= Player.Instance.Attack;

        if (Health <= 0)
            Destroy(gameObject);
    }
}
```

Рисунок 2.6 код PlayerController

```
public class PlayerController : MonoBehaviour
{
    public Vector2Int CurrentPosition;

    public void TryMove(Vector2Int targetPosition)
    {
        if (IsAdjacent(targetPosition))
        {
            CardBase targetCard = Board.Instance.GetCardAt(targetPosition);
            targetCard?.OnPlayerEnter();

            CurrentPosition = targetPosition;
            transform.position = Board.Instance.GridToWorld(targetPosition);
        }
    }

    private bool IsAdjacent(Vector2Int pos)
    {
        return (Mathf.Abs(pos.x - CurrentPosition.x) + Mathf.Abs(pos.y - CurrentPosition.y)) == 1;
    }
}
```

Рисунок 2.7 код GameManager

```
public class GameManager : MonoBehaviour
{
    public void StartNewGame()
    {
        Board.Instance.GenerateGrid();
        Player.Instance.SpawnAt(new Vector2Int(1, 1));
        UIManager.Instance.ShowStartUI();
    }

    public void GameOver()
    {
        UIManager.Instance.ShowGameOverScreen();
    }
}
```

Рисунок 2.7 код UIManager

```
public class UIManager : MonoBehaviour
{
    public GameObject GameOverPanel;
    public GameObject WinPanel;

    public void ShowGameOverScreen()
    {
        GameOverPanel.SetActive(true);
    }

    public void ShowVictoryScreen()
    {
        WinPanel.SetActive(true);
    }
}
```