МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики,
інформаційних технологій та інженерії

Кафедра комп'ютерних наук та прикладної математики

"До захисту допущений"
Завідувач кафедри
_____
_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка 3D гри у жанрі Magic Survival з адаптивним сюжетом та кастомізацією ігрових механік
(тема роботи)

Виконав: Фоменко Максим Денисович _____
(прізвище, ім'я, по батькові) (підпис)

група ІПЗ- 41

Керівник:_____ _____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2025

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 44 с., 7 рисунки, 12 джерел.

Мета роботи: розробка 3D гри у жанрі Magic Survival, яка матиме адаптивний геймплей та можливості кастомізації ігрових механік, забезпечуючи високу реграбельність та занурення гравця у світ.

Об'єкт дослідження – процес розробки 3D гри у жанрі Magic Survival.

Предмет дослідження – методи та засоби створення ігрових механік, реалізація адаптивного геймплею та кастомізації в межах ігрового рушія Unity.

Методи вивчення – аналіз сучасної ігрової індустрії та жанру Survival, дослідження технологій розробки 3D ігор (Unity, C#, Blender, Git/GitHub), проектування архітектури та ігрових механік, практична реалізація прототипу, демонстрація функціоналу та аналіз результатів.

Результатом роботи є функціональний демонстраційний прототип 3D гри у жанрі Magic Survival. Гра пропонує адаптивний геймплей через систему вибору покращень у вигляді карт та процедурну генерацію спавну ворогів, що забезпечує унікальний досвід кожного проходження. Завдяки використанню ігрового рушія Unity та мови програмування C# було реалізовано ключові механіки: керування персонажем-чаклуном, магічні атаки, поведінка ворогів, а також систему прогресії та локального збереження даних. Розроблений продукт демонструє потенціал для подальшого розвитку та може служити основою для залучення інвестицій та інтересу до ідеї проекту.

Ключові слова: 3D ГРА, MAGIC SURVIVAL, UNITY, C#, ГЕЙМДЕВ, КАСТОМІЗАЦІЯ, ПРОТОТИП, ІГРОВІ МЕХАНІКИ, ПРОГРЕСІЯ.

ВСТУП

Стрімкий розвиток інформаційних технологій та комп'ютерної графіки відкриває нові горизонти для індустрії розваг, зокрема для розробки відеоігор. Сучасні ігрові проекти виходять за межі простої інтерактивної розваги, перетворюючись на складні системи, що поєднують художні, технічні та наративні аспекти. У цьому контексті жанр *Survival* (виживання) у поєднанні з елементами фентезі та магії набуває особливої популярності, пропонуючи гравцям унікальний досвід подолання викликів у динамічному середовищі. Важливість створення таких ігрових продуктів зумовлена постійно зростаючим попитом на ігри з високою реграбельністю, що забезпечується завдяки адаптивним механікам та можливостям кастомізації ігрового досвіду.

Актуальність теми даної дипломної роботи полягає у дослідженні та практичній реалізації сучасної 3D гри, яка відповідає вимогам ринку та інтегрує інноваційні підходи до геймплею. Проблема полягає у створенні ігрового проекту, що здатен занурити гравця у темний фентезійний світ, що поглинається нечистью, та утримувати його увагу через динамічне наростання складності, стратегічний вибір та глибокі можливості кастомізації. Розробка такої гри демонструє здатність до комплексного застосування сучасних технологій та методологій розробки програмного забезпечення у сфері комп'ютерних ігор. Це також дозволяє поглибити розуміння архітектурних рішень та оптимізації, необхідних для створення продуктивного та привабливого ігрового продукту.

Метою кваліфікаційної роботи є розробка 3D гри у жанрі Magic Survival, яка матиме адаптивний геймплей та можливості кастомізації ігрових механік, забезпечуючи високу реграбельність та занурення гравця у світ. Для досягнення цієї мети було поставлено наступні завдання:

1. Проаналізувати сучасний стан ігрової індустрії та особливості жанру *Magic Survival*.
2. Дослідити ключові технології та інструменти для розробки 3D ігор,

зокрема ігровий рушій Unity.

3. Розробити архітектуру ігрового проекту з урахуванням адаптивності геймплею та можливостей кастомізації ігрових механік.
4. Реалізувати основні компоненти ігрового процесу: систему бою, систему прогресії персонажа через вибір покращень (апгрейдів) у вигляді карт, систему спавну ворогів та їхньої поведінки.
5. Провести демонстрацію та оцінку працездатності розробленого прототипу гри.
6. Визначити потенційні напрямки подальшого розвитку проекту, враховуючи його інноваційність та можливості для вдосконалення.

Об'єктом дослідження є процес розробки 3D гри у жанрі Magic Survival. Предметом дослідження є методи та засоби створення ігрових механік, реалізація адаптивного геймплею та кастомізації в межах ігрового рушія Unity.

У процесі розробки використано такі методи, інструменти та засоби: ігровий рушій Unity 2022.3.20f1, мова програмування C#, інтегроване середовище розробки JetBrains Rider 2025.1.2. Для створення та обробки графічних активів застосовувались Blender, Aseprite, Adobe Photoshop 2022 та Paint. Анімації імпортувалися з Mixamo та створювалися за допомогою вбудованого в Unity Animator. Контроль версій проекту здійснювався за допомогою GitHub та GitKraken. Локальне збереження даних гравця реалізовано за допомогою PlayerPrefs Unity, що не передбачає використання повноцінних баз даних.

Особливістю роботи є інтеграція елементів адаптивного геймплею та кастомізації ігрових механік у жанрі Magic Survival через динамічну систему вибору покращень (апгрейдів) через карти та процедурну генерацію спавну ворогів, що забезпечує унікальний досвід кожного проходження. Практична цінність роботи полягає у створенні функціонального прототипу 3D гри, який демонструє потенціал для подальшої розробки та може служити основою для

створення повноцінного ігрового продукту.

Структура роботи відповідає вимогам і складається зі вступу, чотирьох розділів основної частини, висновків, списку використаних джерел та додатків. У першому розділі буде розглянуто теоретичні основи та сучасний стан розробки ігор, у другому – проектування архітектури та ігрових механік, у третьому – практична реалізація та демонстрація продукту, а у четвертому – демонстрація та аналіз результатів.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ІГРОВИХ ПРОЕКТІВ

1.1. Еволюція комп'ютерних ігор та сучасний стан ігрової індустрії

Розвиток комп'ютерних ігор пройшов шлях від простих текстових симуляторів та примітивних аркад до складних інтерактивних світів, які є однією з найдинамічніших та найприбутковіших галузей світової економіки. Перші ігри, такі як «Tennis for Two» (1958) або «Spacewar!» (1962), заклали основи цифрових розваг, демонструючи потенціал електронних пристроїв для створення віртуальних світів. Поява аркадних автоматів у 1970-х роках, зокрема «Pong» (1972) та «Space Invaders» (1978), відкрила шлях до масового ринку. Згодом, розвиток персональних комп'ютерів та ігрових консолей у 1980-х та 1990-х роках (Atari, Nintendo, PlayStation) сприяв виникненню різноманіття жанрів та значному покращенню графіки та ігрового процесу.

Сучасна ігрова індустрія є глобальним феноменом, що об'єднує мільйони розробників, видавців та гравців. Її щорічний дохід перевищує показники кіно- та музичної індустрії разом узятих. Це обумовлено не лише технологічним прогресом (реалістична графіка, віртуальна та доповнена реальність, штучний інтелект), але й розширенням аудиторії, що охоплює гравців різного віку та соціальних груп. Ключовими тенденціями сучасної ігрової індустрії є:

1. Мобільний геймінг: Швидке зростання популярності мобільних ігор, які стають доступними для широкого кола користувачів.
2. Кросплатформність: Можливість гри на різних пристроях, що забезпечує більшу гнучкість для гравців.
3. Ігри як сервіс (Games as a Service - GaaS): Модель, за якої ігри отримують постійні оновлення контенту та підтримуються протягом тривалого часу.
4. Кіберспорт: Професійні змагання з відеоігор, що приваблюють мільйони глядачів.

5. Інді-розробка: Зростання впливу незалежних розробників, які створюють унікальні та інноваційні проекти, що часто відзначаються творчою свободою та експериментами з жанрами.

В контексті цих тенденцій, розробка 3D гри, що поєднує елементи популярних жанрів та пропонує адаптивний геймплей, є актуальним напрямком, що має значний потенціал на сучасному ринку.

1.2. Огляд жанрів ігор та особливості *Magic Survival*

Ігрова індустрія відзначається надзвичайним жанровим розмаїттям, яке постійно еволюціонує та породжує нові гібридні форми. Серед найбільш популярних та впливових жанрів можна виділити рольові ігри (RPG), шутери (FPS/TPS), стратегії, пригоди, симулятори та багато інших. Кожен жанр має свої ключові характеристики, що визначають ігровий процес та очікування гравців.

Особливе місце в сучасному геймдеві посідають ігри в жанрі Survival (виживання). Ці ігри зосереджені на здатності гравця виживати в агресивному або ворожому середовищі, часто з обмеженими ресурсами. Ключові елементи жанру Survival включають: збір ресурсів, крафтинг, менеджмент показників персонажа (голод, спрага, здоров'я), будівництво, а також постійну загрозу з боку ворогів або навколишнього середовища. Мета гравця, як правило, полягає у максимально довгому виживанні або досягненні певної кінцевої цілі, що вимагає стратегічного планування та адаптації до мінливих умов.

Жанр Magic Survival є гібридом, що поєднує вищезгадані елементи виживання з фентезійним сеттінгом та акцентом на магічних здібностях. У таких іграх гравець часто виступає в ролі чаклуна або мага, який використовує різноманітні заклинання та магічні артефакти для боротьби з надприродними ворогами та подолання викликів. Основні елементи Magic Survival включають:

1. Використання магії: Бойова система побудована на застосуванні заклинань з різними візуальними формами та механіками (наприклад,

прямий урон, площинний вплив, точковий вплив, перешкоди для ворогів).

2. Прогресія та кастомізація магічних здібностей: Розвиток персонажа відбувається через вивчення нових заклинань, покращення існуючих або вибір унікальних магічних шляхів.
3. Фентезійний світ: Сеттінг, наповнений міфічними істотами, таємничими локаціями та елементами магії.
4. Виживання під натиском: Постійне протистояння хвилям ворогів, що вимагає продуманого вибору карт та майстерного переміщення гравця для уникнення атак та ефективної боротьби.

Для даної дипломної роботи розробляється 3D гра саме в жанрі Magic Survival. У ній гравець, керуючи чаклуном, має знищувати хвилі ворогів, що спавняться, використовуючи магічні атаки. Після кожної хвилі гравцю надається вибір однієї з трьох випадкових карт покращень, що дозволяє кастомізувати ігрові механіки, збільшуючи урон магії, максимальне здоров'я або інші характеристики. Основна мета гравця – вижити та дійти до 20-ї хвилі, де на нього чекає фінальний бос. З кожною наступною хвилею вороги стають сильнішими, що збільшує ризик поразки та підкреслює елемент виживання у постійно наростаючому темному фентезійному світі, що поглинається нечистю.

1.3. Огляд інструментів та технологій для розробки ігор

Розробка сучасної 3D гри – це багатогранний процес, який вимагає застосування широкого спектру спеціалізованих інструментів та технологій. Правильний вибір цих засобів є критично важливим для ефективності розробки, якості кінцевого продукту та можливості його подальшої підтримки. У даному проекті було обрано низку провідних технологій, що забезпечують повний цикл створення гри.

1.3.1. Ігровий рушій Unity

Unity є одним із найпопулярніших та найпотужніших кросплатформних ігрових рушіїв у світі, що дозволяє розробляти 2D, 3D, VR та AR додатки. Його широке використання в ігровій індустрії (від інді-проектів до AAA-ігор) зумовлене багатофункціональністю, гнучкістю та великою спільнотою розробників. Нами використовується версія Unity 2022.3.20f1, яка надає актуальний набір інструментів та оптимізацій [1].

Ключові переваги Unity для розробки 3D гри:

1. Інтуїтивно зрозуміле середовище розробки – Unity Editor надає потужний візуальний інтерфейс, що дозволяє створювати сцени, розміщувати об'єкти, налаштовувати освітлення та анімацію без глибоких знань програмування на початкових етапах.
2. Кросплатформність – Рушій підтримує розробку для великої кількості платформ (ПК, мобільні пристрої, консолі, веб), що дозволяє охопити широку аудиторію.
3. Компонентно-орієнтована архітектура – Модель GameObjects та Components спрощує організацію проекту, дозволяючи додавати функціонал до об'єктів шляхом приєднання готових або власних скриптів та компонентів.
4. Широка екосистема – Unity Asset Store пропонує величезну бібліотеку готових асетів (моделі, текстури, анімації, скрипти, UI-елементи), що значно прискорює розробку та дозволяє зосередитись на ключових ігрових механіках та їх вдосконаленні.
5. Потужні інструменти – Вбудовані системи для роботи з фізикою (PhysX), анімацією (Mecanim), аудіо, UI, шейдерами та іншими аспектами розробки.

1.3.2. Мова програмування C# та середовище розробки JetBrains Rider

Для написання скриптів у Unity використовується мова програмування C# (Сі-Шарп). C# [2]– це сучасна, об'єктно-орієнтована мова, розроблена Microsoft, яка має високу продуктивність, строгу типізацію та велику кількість бібліотек. Її тісна інтеграція з Unity дозволяє ефективно реалізовувати ігрову логіку, взаємодію об'єктів, управління інтерфейсом та інші функціональні елементи.

Як середовище розробки (IDE) було обрано JetBrains Rider 2025.1.2. Rider є потужним кросплатформним IDE, спеціально розробленим для .NET розробки, що включає C# та Unity. Його ключові переваги:

1. Глибока інтеграція з Unity – Rider розуміє специфіку Unity, надаючи розширену автодоповнення, інспекції коду та можливості рефакторингу, що значно прискорює написання та налагодження ігрових скриптів.
2. Потужні інструменти рефакторингу – Дозволяє швидко та безпечно змінювати структуру коду, підвищуючи його читабельність та підтримуваність.
3. Інтелектуальне автодоповнення та аналіз коду – Допомагає уникати помилок ще на етапі написання, пропонуючи варіанти завершення коду та вказуючи на потенційні проблеми.
4. Зручні засоби налагодження (debugging) – Дозволяє ефективно знаходити та виправляти помилки в ігровій логіці.

Використання Rider у поєднанні з C# забезпечує високу продуктивність розробки, дозволяючи зосередитись на реалізації складних ігрових механік.

1.3.3. Інструменти для створення та управління контентом

Якість візуального та анімаційного контенту є вирішальною для занурення гравця у 3D гру. Для створення та управління ігровими асетами були використані наступні інструменти:

1. Blender[3] – Це потужний безкоштовний 3D-редактор з відкритим вихідним кодом, що дозволяє створювати складні 3D-моделі (персонажі,

оточення, об'єкти), текстури та анімації. Його можливості включають моделювання, скульптінг, UV-розгортку, текстурування та рендеринг, що робить його універсальним інструментом для 3D-художника.

2. Mixamo [4] – Онлайн-сервіс від Adobe, що надає велику бібліотеку готових 3D-анімацій для персонажів. Використання Mixamo значно прискорює процес створення анімацій, дозволяючи швидко застосовувати рухи до власних 3D-моделей або стандартних персонажів. Це особливо корисно на етапі прототипування та для проектів з обмеженими ресурсами.
3. Unity Animator [5] – Вбудована в Unity система для створення та керування анімаціями. Вона дозволяє імпортувати анімації з зовнішніх джерел (як Mixamo), комбінувати їх, створювати переходи між анімаціями (наприклад, з бігу на стрибок) та інтегрувати анімаційну логіку зі скриптами.
4. Aseprite, Adobe Photoshop 2022 [6] та Paint – Ці графічні редактори використовуються для створення та редагування 2D-графіки, текстур, іконок та інших візуальних елементів, які не потребують 3D-моделювання. Aseprite ідеально підходить для піксельної графіки, Photoshop – для професійної обробки зображень, а Paint може бути використаний для простих швидких правок. Деякі готові зображення та 3D-моделі також можуть бути отримані з Unity Asset Store [7].

1.3.4. Системи контролю версій: GitHub, GitKraken та

У командній та навіть індивідуальній розробці програмного забезпечення, зокрема ігор, критично важливим є використання систем контролю версій. Для цього проекту були обрані GitHub та GitKraken.

1. GitHub [8] – Це веб-сервіс для хостингу репозиторіїв Git. Він дозволяє зберігати історію змін коду, працювати в команді над одним проектом

(хоча в цьому випадку це індивідуальна робота, GitHub забезпечує резервне копіювання та доступність проекту з будь-якого місця), відстежувати зміни та керувати версіями проекту.

2. GitKraken [9] – Це графічний клієнт для Git, який спрощує взаємодію з системою контролю версій. Він надає візуальний інтерфейс для виконання таких операцій, як коміти, гілки, злиття (merge), вирішення конфліктів, що робить роботу з Git більш інтуїтивною та менш схильною до помилок порівняно з командним рядком.
3. Trello [10] - це вебплатформа, широко використовувана для ефективного планування та керування проектами. Цей сервіс дозволяє організовувати етапи виконання завдань за допомогою тасків, що можуть включати стандартні статуси, як-от "Потрібно зробити", "В роботі", "Зроблено", або індивідуально визначені етапи.

Використання цих інструментів забезпечує надійне зберігання проекту, можливість відкату до попередніх версій у разі потреби та систематизацію процесу розробки.

1.4. Методології розробки ігор

Розробка комп'ютерних ігор, як і будь-якого складного програмного забезпечення, вимагає застосування структурованих методологій, що дозволяють ефективно керувати процесом, оптимізувати ресурси та мінімізувати ризики. Хоча ігрова індустрія має свої особливості, багато загальноприйнятих методологій розробки програмного забезпечення адаптуються для потреб геймдеву.

Серед найбільш поширених підходів можна виділити:

1. Водоспадна модель (Waterfall Model): Традиційний лінійний підхід, де кожен етап розробки (вимоги, дизайн, реалізація, тестування,

впровадження) завершується повністю перед переходом до наступного. У геймдеві цей підхід рідко використовується для всього проекту через його негнучкість та труднощі з адаптацією до змін у процесі розробки[11].

2. Ітеративні та інкрементальні моделі: Ці підходи передбачають розбивку проекту на невеликі цикли (ітерації), де на кожній ітерації створюється робочий, але не повний, фрагмент продукту. Це дозволяє отримувати зворотний зв'язок на ранніх етапах та поступово нарощувати функціонал[12].
3. Гнучкі методології (Agile Methodologies): Наприклад, Scrum або Kanban. Це найпопулярніші підходи в сучасній ігровій індустрії. Agile зосереджується на гнучкості, співпраці, швидкій адаптації до змін та постійному наданні цінності. Ключові принципи включають короткі ітерації (спринти), щоденні зустрічі (стендапи), постійне тестування та інтеграцію.
   3.1. Scrum: передбачає роботу у фіксованих за часом ітераціях (спринтах), які зазвичай тривають від 1 до 4 тижнів. Кожен спринт завершується робочим інкрементом продукту.
   3.2. Kanban: фокусується на візуалізації робочого процесу, обмеженні кількості одночасно виконуваних завдань та безперервному потоці розробки.

В умовах розробки інді-ігор або прототипів, як у випадку даної дипломної роботи, часто використовується спрощений, але гнучкий підхід, що поєднує елементи ітеративної розробки та принципи Agile. Це дозволяє швидко втілювати ідеї, тестувати механіки та адаптуватися до викликів, які неминуче виникають у процесі створення ігрового продукту. Постійні перевірки працездатності окремих компонентів та швидке внесення змін є ключовими для ефективної розробки.

Отже в першому розділі було здійснено комплексний аналіз теоретичних та методологічних аспектів розробки ігрових проектів. Розглянуто еволюцію комп'ютерних ігор від їх зародження до сучасного стану індустрії, виділено ключові тенденції та вплив на ринок. Детально проаналізовано жанр *Magic Survival*, його основні характеристики та особливості, які знайшли відображення у розроблюваній грі. Проведено огляд сучасних інструментів та технологій, таких як Unity, C#, JetBrains Rider, Blender, Mixamo, Unity Animator, графічні редактори та системи контролю версій (GitHub, GitKraken), обґрунтовано їхній вибір для проекту. Крім того, були розглянуті основні методології розробки ігор, з акцентом на гнучкі підходи, що забезпечують ефективність та адаптивність процесу створення ігрового продукту. Отримана теоретична база є фундаментом для подальшого проектування та практичної реалізації дипломної роботи.

РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО КОМПЛЕКСУ

2.1. Аналіз вимог до ігрового продукту

Етап аналізу вимог є фундаментальним для успішної розробки будь-якого програмного продукту, зокрема ігор. Він дозволяє чітко визначити, що саме має робити система, які її можливості та обмеження. Це допомагає уникнути непорозумінь на подальших етапах та забезпечити відповідність кінцевого продукту очікуванням. У контексті розробки 3D гри в жанрі Magic Survival, вимоги були розділені на функціональні та нефункціональні.

2.1.1. Функціональні вимоги

Функціональні вимоги описують, що система повинна робити, тобто її основний ігровий процес та можливості взаємодії гравця. Для даної гри було визначено наступні ключові функціональні вимоги:

1. Система гравця:
   - 1.1. Керування персонажем-чаклуном у 3D просторі.
   - 1.2. Відображення рівня здоров'я (HP) гравця.
   - 1.3. Реалізація механіки отримання урону від ворогів.
   - 1.4. Можливість атаки ворогів магічними заклинаннями.
2. Система ворогів:
   - 2.1. Спавн хвиль ворогів, що наростають по складності.
   - 2.2. Різні типи ворогів (за виглядом, силою, швидкістю).
   - 2.3. Механіка руху ворогів (переслідування гравця).
   - 2.4. Атака ворогів (наприклад, контактний урон).
   - 2.5. Відображення рівня здоров'я (HP) ворогів.
   - 2.6. Зникнення ворогів після їх знищення.
3. Система прогресії та кастомізації:

3.1. Надання гравцю вибору однієї з трьох випадкових карт покращень після кожної пройденої хвилі.

3.2. Застосування ефектів від обраних карт (збільшення урону, максимального HP, швидкості руху тощо).

3.3. Збереження прогресії (наприклад, досягнутої хвилі, вбитих ворогів, кращого часу проходження) між ігровими сесіями.

4. Ігровий процес та логіка:

4.1. Запуск гри та перехід до ігрової сцени.

4.2. Управління ігровими хвилями (початок, завершення, наростання складності).

4.3. Умови перемоги (наприклад, перемога над фінальним босом на 20-й хвилі).

4.4. Умови поразки (зменшення здоров'я гравця до нуля).

4.5. Відображення ігрового інтерфейсу (UI: здоров'я гравця, номер хвилі, кількість вбитих ворогів).

5. Візуальна та звукова складові:

5.1. Відображення 3D оточення та персонажів.

5.2. Візуалізація магічних атак та ефектів.

5.3. Відтворення звукових ефектів (атаки, отримання урону, спавн ворогів, вибір карт).

2.1.2. Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи та обмеження, в яких вона повинна функціонувати. Для даної гри були визначені такі нефункціональні вимоги:

1. Продуктивність: Забезпечення стабільного робочого білду для цільових ПК, що дозволяє демонструвати основний функціонал гри.
2. Зручність використання (Usability): Інтуїтивно зрозуміле керування

персонажем та навігація по ігровому інтерфейсу. Чітке відображення ігрової інформації (здоров'я, хвиля, вибір карт).

3. Надійність: Стабільна робота гри без критичних збоїв та помилок. Коректне збереження та завантаження прогресу гравця.
4. Безпека: Забезпечення базового захисту даних гравця (локально) від несанкціонованого доступу або пошкодження.
5. Сумісність: Можливість запуску гри на стандартних конфігураціях ПК з операційною системою Windows.
6. Розширюваність: Архітектура проекту повинна забезпечувати можливість легкої зміни параметрів карт покращень та налаштування спавну ворогів, що є важливим для подальшого розвитку прототипу

Аналіз цих вимог став основою для подальшого проектування архітектури ігрового продукту, його компонентів та взаємодії між ними, з метою створення демонстраційного прототипу для привернення уваги потенційних спонсорів та зацікавлених осіб до ідеї продукту.

2.2. Архітектура програмного продукту

Проектування архітектури програмного продукту є критично важливим етапом, що визначає його стабільність, розширюваність та підтримуваність. Для 3D гри в жанрі Magic Survival, розробленої на рушії Unity, була застосована модульна архітектура, яка забезпечує чітке розділення відповідальностей між різними компонентами системи. Такий підхід сприяє організації розробки, полегшує налагодження та дозволяє інтегрувати нові функціональні елементи без витрат зайвих ресурсів.

2.2.1. Загальна архітектура гри

Загальна архітектура гри базується на парадигмі, що використовується в Unity – компонентно-орієнтований підхід. Це означає, що кожен ігровий об'єкт (GameObject) складається з набору компонентів (Components), які визначають

його поведінку та властивості. Такий підхід дозволяє створювати гнучкі та багаторазово використовувані елементи.

У контексті даної роботи, під модулем розуміється логічно відокремлена частина програмного комплексу, яка відповідає за певну групу функціональності та об'єднує суміжні класи, скрипти, асети та налаштування, необхідні для її реалізації. Це дозволяє організувати код та ресурси у керовані та взаємозалежні блоки.

Основними логічними блоками та компонентами архітектури є:

1. Ігровий рушій Unity – Як базова платформа, що надає функціонал рендерингу, фізики, анімації, управління ресурсами та подіями.
2. Ігрові об'єкти (GameObjects): Базові елементи, що представляють собою все, що існує в ігровому світі (гравець, вороги, карти покращень, оточення, UI-елементи).
3. Скрипти C# (Components): Основний механізм для реалізації ігрової логіки. Кожен скрипт є компонентом, що прикріплюється до GameObject і керує його поведінкою (наприклад, рух гравця, штучний інтелект ворогів, логіка вибору карт).
4. Система подій (Event System): Ця система забезпечує взаємодію між різними компонентами гри без прямої залежності між ними. Один компонент (наприклад, система урону) може "сповістити" всі інші компоненти, що щось сталося (наприклад, "гравець отримав урон"). Компоненти, які "підписані" на цю подію (наприклад, менеджер UI для оновлення смуги здоров'я, система звуку для відтворення ефекту болю), обробляють отриману інформацію, не знаючи, хто саме ініціював подію. Це забезпечує високу модульність та дозволяє кожному компоненту займатися своєю областю відповідальності незалежно від інших.
5. Система збереження даних (PlayerPrefs) – Використовується для локального збереження ігрової прогресії (наприклад, кращий час, вбиті

ворогів). Ця операція не є ресурсно затратною, що дозволяє ефективно зберігати та завантажувати невеликі обсяги ігрових даних.

2.2.2. Модулі та їх взаємодія

Архітектура гри представлена як сукупність взаємодіючих модулів, кожен з яких відповідає за певну частину функціоналу. Це забезпечує легкість розширення та тестування.

1. Модуль гравця:
   - 1.1. Відповідає за керування рухом, атаками гравця, а також за його характеристики (здоров'я, урон, швидкість).
   - 1.2. Взаємодіє з модулем ворогів (нанесення урону), модулем UI (відображення HP), та модулем прогресії (отримання покращень).
2. Модуль ворогів та спавну:
   - 2.1. Керує генерацією хвиль ворогів, їхнім спавном на сцені з урахуванням процедурних параметрів.
   - 2.2. Відповідає за поведінку ворогів (переслідування, атаки), їхнє здоров'я та знищення.
   - 2.3. Взаємодіє з модулем гравця (атаки на гравця), модулем UI (кількість вбитих ворогів).
3. Модуль прогресії та кастомізації (система карт):
   - 3.1. Відповідає за логіку вибору карт після хвилі.
   - 3.2. Застосовує ефекти обраних карт до характеристик гравця або ігрового процесу.
   - 3.3. Взаємодіє з модулем гравця (зміна характеристик), модулем UI (відображення карт).
4. Модуль ігрового менеджменту:
   - 4.1. Керує загальним станом гри (меню, початок/кінець гри, переходи між хвилями, умови перемоги/поразки).

4.2. Відповідає за логіку підрахунку очок, часу, та збереження даних.

4.3. Взаємодіє з усіма іншими модулями, координуючи їх роботу.

5. Модуль користувацького інтерфейсу (UI):

5.1. Відповідає за візуальне відображення інформації для гравця (HUD, меню, карти покращень, вікно статистики).

5.2. Обробляє введення гравця через елементи UI (кнопки, текстові поля).

5.3. Взаємодіє з усіма модулями, отримуючи від них дані для відображення.

6. Модуль аудіо:

6.1. Відповідає за відтворення фонової музики та звукових ефектів (атаки, отримання урону, спавн ворогів, вибір карт).

6.2. Взаємодіє з іншими модулями, реагуючи на ігрові події.

Така модульна архітектура забезпечує чітке розділення відповідальностей, що робить проект більш керованим, дозволяє легко вносити зміни та розширювати функціонал, що є особливо важливим для демонстраційного прототипу.

2.3. Проектування ігрових механік та алгоритмів

Детальне проектування ігрових механік та алгоритмів є основою для реалізації функціонального та захоплюючого ігрового процесу. На цьому етапі визначається логіка поведінки ключових елементів гри, їхня взаємодія та математичні моделі, що лежать в їх основі.

2.3.1. Механіка гравця

Центральним елементом гри є керований гравцем персонаж-чаклун. Проектування механіки гравця включає:

1. Рух та керування: Реалізація системи переміщення персонажа у 3D просторі (наприклад, за допомогою контролера персонажа), що дозволяє

гравцю вільно переміщатися по ігровій карті. Керування здійснюється за допомогою клавіатури та миші.

2. Система здоров'я (HP): Відстеження поточного рівня здоров'я гравця, його максимального значення. Реалізація логіки отримання урону (зменшення HP при контакті з ворогами) та, за потреби, механіки відновлення здоров'я (наприклад, через покращення).
3. Магічні атаки: Проектування системи стрільби магічними снарядами. Це включає визначення типу снарядів (дальність, швидкість, урон), їх візуалізації, механіки прицілювання та взаємодії з ворогами. Кожен тип магічної атаки матиме свої алгоритми розрахунку урону та взаємодії.

2.3.2. Механіка ворогів та система спавну

Вороги є основним джерелом виклику в грі. Їхнє проектування передбачає:

1. Поведінка ворогів: Реалізація базового штучного інтелекту ворогів, що дозволяє їм переслідувати гравця. Алгоритми руху ворогів мають забезпечувати їхнє наближення до гравця та можливість нанесення йому урону.
2. Система здоров'я ворогів: Аналогічно гравцю, кожен ворог має свій рівень здоров'я, що зменшується при отриманні урону від магічних атак гравця. Після досягнення нульового значення HP, ворог знищується.
3. Система спавну хвиль: Проектування алгоритму генерації хвиль ворогів. Це включає:
    3.1. Процедурна генерація спавну: Вороги з'являються у випадкових (або псевдовипадкових) місцях на карті в межах визначених зон. Це забезпечує унікальність кожної хвилі та підвищує реграбельність.
    3.2. Наростання складності: З кожною новою хвилею кількість ворогів, їхній тип та/або характеристики (здоров'я, урон, швидкість)

збільшуються. Це вимагає динамічної адаптації гравця.

3.3. Управління хвилями: Алгоритми відстеження кількості живих ворогів у поточній хвилі та ініціації наступної після знищення всіх попередніх.

2.3.3. Система прогресії та кастомізації (карти покращень)

Ключовим елементом адаптивного геймплею та кастомізації є система покращень через карти. Її проектування включає:

1. Механіка вибору карт: Після завершення кожної хвилі гравцю пропонується на вибір три випадкові карти покращень. Алгоритм вибору карт має забезпечувати випадковість їхньої появи з певного пулу доступних карт.
2. Типи покращень: Проектування різноманітних типів покращень, що впливають на характеристики гравця або його здібності. Це можуть бути:
   2.1. Збільшення базового урону магічних атак.
   2.2. Збільшення максимального здоров'я гравця.
   2.3. Збільшення швидкості руху персонажа.
   2.4. Зміни в механіках магічних атак (наприклад, зміна візуалізації, площі ураження).
3. Застосування покращень: Алгоритм застосування обраної карти, що миттєво змінює відповідні характеристики гравця або активує нові механіки.
4. 2.3.4. Алгоритми збереження та завантаження даних
5. Для забезпечення неперервності ігрового процесу та збереження прогресу гравця використовується система локального збереження даних.
6. Збереження даних: Алгоритм запису ключових ігрових параметрів (наприклад, досягнута хвиля, кількість вбитих ворогів, кращий час проходження) у локальне сховище (PlayerPrefs) після завершення гри або

між ігровими сесіями.

7. Завантаження даних: Алгоритм читання збережених параметрів з PlayerPrefs при запуску гри або перезапуску нової сесії, що дозволяє відобразити попередні досягнення гравця.

Отже в другому розділі було здійснено детальний аналіз вимог до розроблюваної 3D гри в жанрі Magic Survival, які були розділені на функціональні та нефункціональні. Чітко визначено ключові ігрові можливості, такі як система гравця, система ворогів та спавну, система прогресії через карти покращень, а також загальна логіка ігрового процесу та візуальні й звукові компоненти. Серед нефункціональних вимог акцентовано на забезпеченні стабільного робочого білду для ПК, зручності використання, надійності, сумісності з Windows та розширюваності для легкої модифікації ігрових параметрів.

Проектування архітектури програмного продукту базувалося на компонентно-орієнтованому підході Unity, де кожен ігровий об'єкт є сукупністю взаємодіючих компонентів. Було визначено основні логічні модулі гри – модуль гравця, модуль ворогів та спавну, модуль прогресії та кастомізації (система карт), модуль ігрового менеджменту, модуль користувацького інтерфейсу та модуль аудіо – та описано їхню взаємодію за допомогою, зокрема, системи подій. Розглянуто використання PlayerPrefs для збереження даних, підкресливши, що ця операція не є ресурсно затратною.

Детальне проектування ігрових механік охопило алгоритми руху та атаки гравця, реалізацію системи здоров'я, поведінку ворогів (переслідування та атаки), процедурну генерацію їхнього спавну з наростанням складності, а також логіку функціонування системи карт покращень та алгоритми локального збереження/завантаження даних.

Результати проектування стали основою для подальшої практичної

реалізації демонстраційного прототипу гри, що дозволить привернути увагу до ідеї продукту.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1. Створення ігрового середовища та основних ігрових об'єктів

Етап реалізації розпочинається зі створення базового ігрового середовища та інтеграції основних ігрових об'єктів у рушій Unity. Це формує візуальну основу, на якій будуються всі подальші механіки та взаємодії.

Створення ігрової сцени: Ігрове середовище було спроектовано як 3D сцена в Unity. Для цього використовувались вбудовані інструменти Unity та імпортовані 3D-моделі оточення. Це включає створення ландшафту (Terrain), розміщення статичних об'єктів (декорації, перешкоди), налаштування освітлення (Directional Light) та камер (Main Camera). Для навігації ворогів була використана система Unity NavMesh, що забезпечує коректне переміщення об'єктів штучного інтелекту .

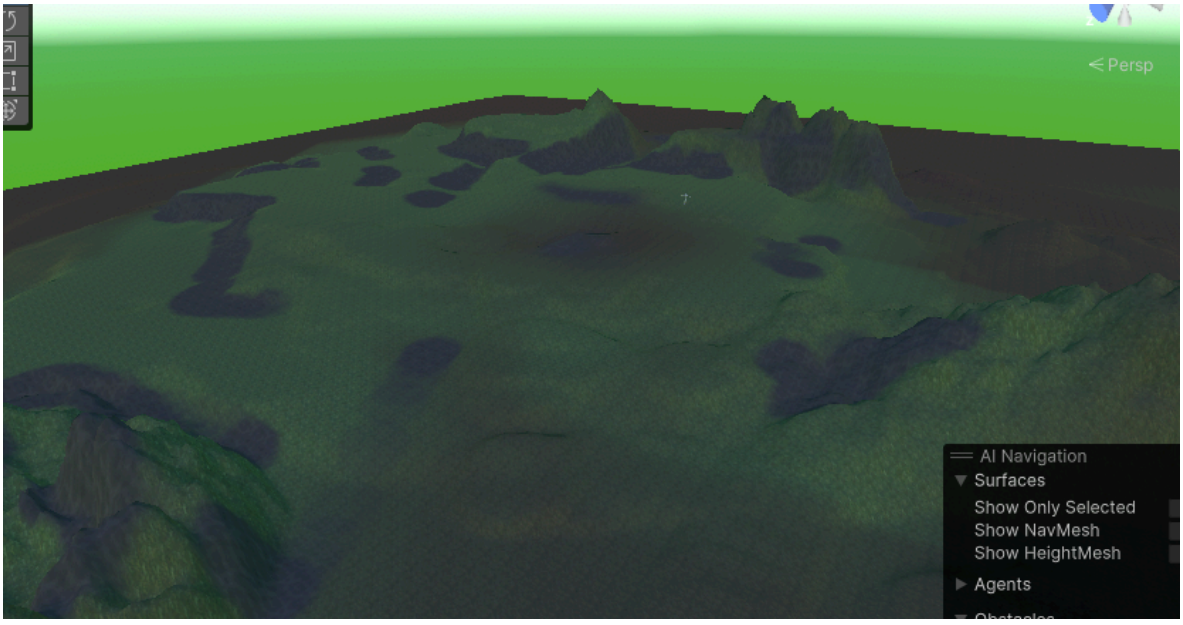

Рис. 3.1. Загальний вигляд ігрової сцени у Unity Editor.

Інтеграція гравця: Модель персонажа-чаклуна була імпортована в Unity. Для забезпечення його руху та взаємодії з оточенням було налаштовано контролер персонажа (Character Controller) та розроблено скрипт MouseLook.cs

для управління рухом (WASD) та орієнтацією (мишею). Анімації (рух, атаки) були імпортовані з сервісу Mixamo та інтегровані через систему Unity Animator.

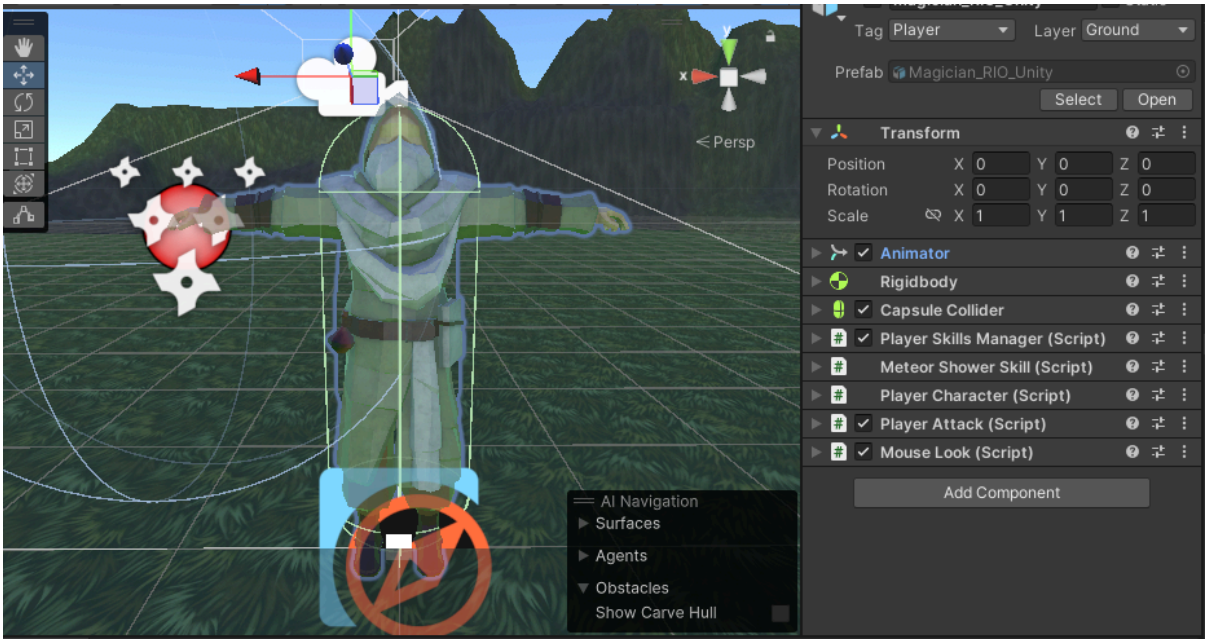

Рис. 3.2. 3D-модель персонажа-чаклуна та його компоненти в Unity.

На рис. 3.2. представлено 3D-модель персонажа-чаклуна у початковій Т-позі, яка використовується для імпорту та налаштування анімацій у Unity. Фрагмент вікна Inspector праворуч демонструє інтеграцію ключових компонентів та розроблених скриптів до ігрового об'єкта гравця, забезпечуючи його повний функціонал.

Фрагмент вікна Inspector праворуч демонструє інтеграцію ключових компонентів та розроблених скриптів до ігрового об'єкта гравця, забезпечуючи його повний функціонал:

1. Character Controller – компонент, який забезпечує фізичне переміщення персонажа у 3D-просторі та його взаємодію з оточенням без необхідності складної фізичної симуляції.
2. Animator – системний компонент Unity, що керує анімаціями моделі. Інтегрує анімації, імпортовані з сервісу Mixamo, та дозволяє створювати переходи між ними згідно з ігровою логікою.
3. PlayerCharacter (Script) – базовий скрипт, що інкапсулює основні

характеристики гравця, такі як рівень здоров'я та урон, а також керує його загальним станом.

4. MouseLook (Script) – скрипт для обробки введення гравця з клавіатури (WASD для руху) та миші (для орієнтації та прицілювання), забезпечуючи контроль над анімаційним контролером персонажа.
5. PlayerAttack (Script) – компонент, відповідальний за реалізацію механіки магічних атак гравця. Він ініціює стрільбу магічними снарядами, контролює їх появу та управління перезарядкою.
6. PlayerSkillsManager (Script) – централізовано керує системою скілів гравця, включаючи відстеження кулдаун та ініціацію активації різних магічних здібностей.
7. MeteorShowerSkill (Script) – реалізує специфічну логіку скіла “Метеоритний Дощ", керуючи його візуальними ефектами, механізмом нанесення AOE-урону та взаємодією з об'єктами сцени.

Реалізація ворогів: Для ворогів були імпортовані відповідні 3D-моделі. Кожен тип ворога був налаштований як окремий ігровий об'єкт зі своїми компонентами (скриптами для поведінки, коллайдерами для взаємодії). Для забезпечення їхньої навігації та взаємодії з гравцем використовувались компоненти Unity NavMesh Agent та спеціалізовані скрипти штучного інтелекту, що дозволяють переслідувати та атакувати ціль

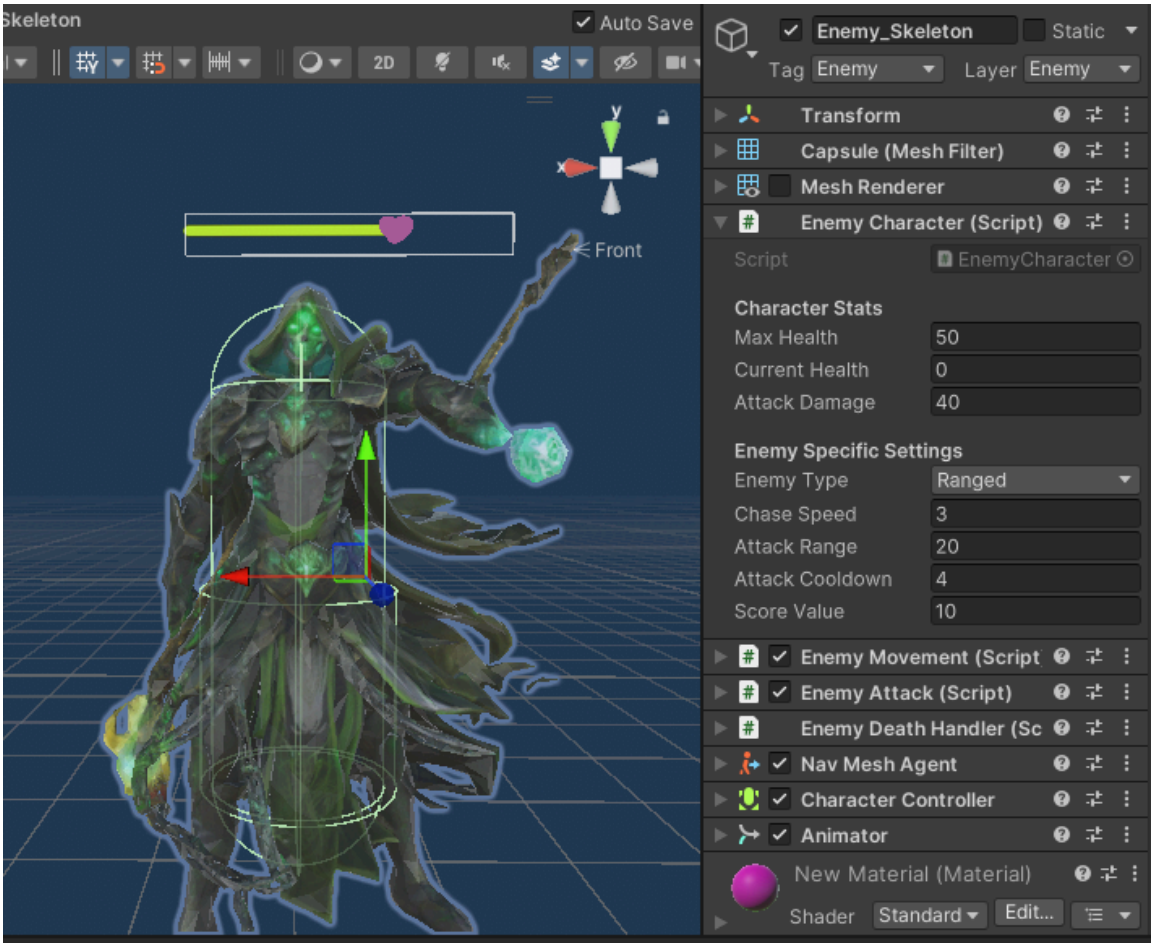

Рис. 3.3. Приклад 3D-моделі ворога з налаштованими компонентами.

На рис. 3.3. представлено 3D-модель ворога та фрагмент його Inspector, що демонструє ключові компоненти, які забезпечують його функціональність та взаємодію з ігровим світом.

Nav Mesh Agent – компонент Unity, що дозволяє ворогу здійснювати навігацію по запеченому NavMesh, переслідуючи гравця та оминаючи перешкоди.

1. Collider – використовується для виявлення зіткнень з магічними снарядами гравця та для взаємодії з оточенням.
2. Animator – керує анімаціями ворога (рух, атаки, смерть) відповідно до його поведінки.
3. EnemyCharacter (Script) – базовий скрипт, що інкапсулює основні характеристики ворога: рівень здоров'я, урон, швидкість переслідування

та кулдаун атаки.

4. EnemyMovement (Script) – реалізує логіку руху ворога, дозволяючи йому переслідувати гравця та зупинятися на певній відстані для атаки.
5. EnemyAttack (Script) – відповідає за логіку атаки ворога, включаючи визначення типу атаки (ближній чи дальній бій), її ініціацію та нанесення урону гравцю.
6. EnemyHealthBar (Script) – керує візуальним відображенням смуги здоров'я ворога над його головою, яка завжди орієнтована на камеру гравця.

Створення візуальних ефектів магії: Для магічних атак гравця були розроблені та інтегровані відповідні візуальні ефекти (частинкові системи, шейдери) , що супроводжують застосування заклинань та зіткнення з ворогами. Ці ефекти підкреслюють візуальну форму та механіки магічних здібностей.

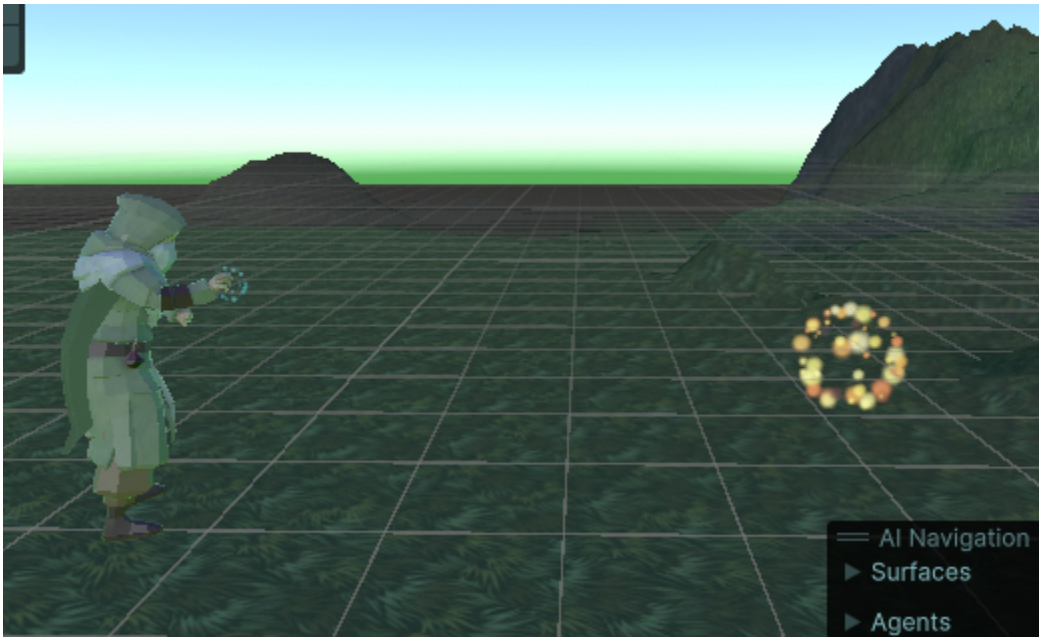

Рис. 3.4. Приклад візуального ефекту магічної атаки.

3.2. Розробка та інтеграція ігрових механік

На цьому етапі була реалізована основна логіка гри, що дозволила ігровим об'єктам взаємодіяти між собою згідно з розробленим проектом.

1. Реалізація механіки гравця:

1.1. Рух: Запрограмовано скрипт керування рухом гравця, який обробляє введення з клавіатури (WASD) та орієнтує персонажа в просторі. Алгоритм забезпечує плавне переміщення та обертання персонажа відповідно до напрямку руху та погляду миші.

1.2. Атаки магією: Розроблено скрипт для ініціації магічних атак. Це включає створення та появу (інстанціювання) магічних снарядів (префабів) у напрямку, куди дивиться гравець, а також логіку зіткнення цих снарядів з ворогами. Кожен тип магічної атаки має власні параметри: урон, швидкість, візуальний ефект.

1.3. Система здоров'я: Впроваджено скрипт, що відстежує поточне та максимальне значення HP гравця. При отриманні урону від ворогів, HP гравця зменшується. Реалізовано перевірку умови поразки (HP <= 0), після чого ігрова сесія завершується.

2. Реалізація механіки ворогів та спавну:

2.1. Поведінка ворогів: Створено скрипти штучного інтелекту для ворогів, що дозволяють їм переслідувати гравця. Це реалізовано за допомогою вбудованих засобів Unity (NavMesh Agent) простий алгоритмів руху до цілі. Вороги атакують гравця при досягненні певної відстані.

2.2. Система спавну: Розроблено менеджер хвиль, який контролює появу ворогів. Після завершення кожної хвилі, менеджер створює нову групу ворогів. Процедурна генерація появи ворогів забезпечується випадковим вибором місць їхньої появи в межах заданих зон ігрової сцени, що підвищує унікальність кожного проходження. З кожною хвилею збільшується кількість ворогів та/або їхні характеристики (сила, швидкість, HP).

2.3. Взаємодія з атаками гравця: Скрипти ворогів обробляють зіткнення з

магічними снарядами гравця, зменшуючи своє HP. Після знищення ворога (HP <= 0), він видаляється зі сцени.

Рис. 3.5. Момент ігрового процесу: чаклун атакує ворогів магічними снарядами.

3.3. Реалізація системи прогресії та кастомізації (карти покращень)

Система карт покращень є основним механізмом, що забезпечує адаптивний геймплей та високу реграбельність.

1. Візуалізація карт: Для кожної карти покращень створено відповідні візуальні асети: унікальні іконки, текстові описи та рамки, що відображають рідкість карт. Ці елементи інтегровано в інтерфейс користувача та представлено гравцеві після завершення кожної хвилі.
2. Логіка вибору карт: Розроблено скрипт, який випадковим чином обирає три унікальні карти з пулу доступних покращень, після чого відображає їх на екрані для вибору гравцем. Система обробляє вибір гравця (натискання кнопки або іконки карти), застосовуючи ефекти обраної карти до його

характеристик. Це включає збільшення базового урону магічних атак, збільшення максимального здоров'я гравця, збільшення швидкості руху персонажа, а також зміни в механіках магічних атак (зокрема, урону скілів, радіусу дії та тривалості скілів, зменшення кулдауна, збільшення швидкості стрільби) та активацію пасивної регенерації HP

3. Управління пулом карт: Створено структуру даних для зберігання інформації про всі доступні карти та їхні ефекти. Така архітектура дозволяє легко додавати нові типи покращень у майбутньому без зміни основної логіки програми, забезпечуючи розширюваність системи.

Рис. 3.6. Екран вибору карт покращень після завершення хвилі.

3.4. Розробка користувацького інтерфейсу (UI) та системи збереження даних

Ефективний та інтуїтивно зрозумілий користувацький інтерфейс (UI) є невід'ємною частиною будь-якої гри. Крім того, для збереження прогресу гравця реалізовано систему локального збереження даних.

1. Проектування та реалізація елементів UI: Створено елементи UI для відображення ключової ігрової інформації під час геймплею. До них належать: смуга здоров'я гравця, поточний номер хвилі, кількість вбитих ворогів. Ці елементи оновлюються в реальному часі. Додатково розроблено Health Bar, що відображається над головою ворогів та завжди орієнтований на гравця, а також UI для системи скілів з відображенням іконок та індикаторами перезарядки. Інтерфейс вибору карт покращень забезпечує візуалізацію опцій прогресії.
2. Меню та екрани статистики: Розроблено стартове меню, екран поразки/перемоги, а також екран зі статистикою гри (найкращий час проходження, загальна кількість вбитих ворогів за всі сесії).
3. Система збереження та завантаження даних (PlayerPrefs): Реалізовано алгоритми для збереження та завантаження ключових даних гравця (кращий час, кількість вбитих ворогів) за допомогою PlayerPrefs. Ця система забезпечує збереження прогресу між ігровими сесіями та при перезапуску гри.

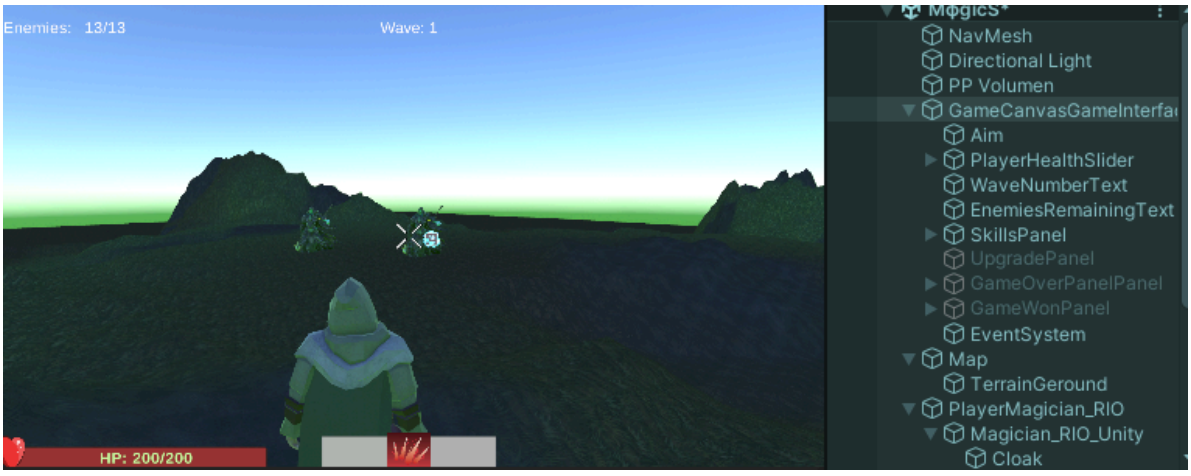

Рис. 3.7. Елементи користувацького інтерфейсу (HUD) під час ігрового процесу.

Отже в третьому розділі було детально розглянуто етапи та особливості практичної реалізації демонстраційного прототипу 3D гри у жанрі Magic

Survival. Описано процес створення базового ігрового середовища в Unity, включаючи налаштування сцени, інтеграцію моделей гравця та ворогів, а також розробку візуальних ефектів магії.

Ключовим аспектом розділу стала деталізація розробки та інтеграції основних ігрових механік. Зокрема, було описано реалізацію механіки гравця (рух, магічні атаки, система здоров'я), механіки ворогів (поведінка, спавн хвиль з процедурною генерацією та наростанням складності) та їх взаємодія з атаками гравця. Особливу увагу приділено системі прогресії та кастомізації через карти покращень, що включає візуалізацію карт, логіку їх вибору та застосування ефектів. Наостанок, описано реалізацію користувацького інтерфейсу (HUD, меню, екрани статистики) та системи локального збереження/завантаження даних за допомогою PlayerPrefs.

Всі ці кроки дозволили створити функціональний прототип гри, який демонструє основні ігрові концепції та є готовим для подальшого вдосконалення та презентації.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1. Демонстрація функціоналу розробленого прототипу гри

На цьому етапі кваліфікаційної роботи була проведена демонстрація функціоналу розробленого прототипу 3D гри у жанрі Magic Survival. Метою демонстрації є підтвердження працездатності реалізованих ігрових механік та загальної концепції продукту, а також візуальна презентація досягнутих результатів.

Прототип гри було запущено на цільовій платформі (ПК з ОС Windows), що дозволило оцінити його основні можливості. Демонстрація охопила такі ключові аспекти:

1. Запуск гри та стартове меню: Презентація початкового екрану, який дозволяє гравцю розпочати нову ігрову сесію.
2. Основний ігровий процес: Демонстрація керування персонажем-чаклуном, його магічних атак та взаємодії з ворогами. Показано динаміку бою та візуальні ефекти заклинань.
3. Система спавну ворогів: Відображення процедурного спавну хвиль ворогів, що наростають по кількості та/або силі. Демонстрація поведінки ворогів, їхнього переслідування гравця та отримання урону.
4. Система прогресії та вибору карт: Презентація екрану вибору покращень після пройденої хвилі. Показано, як гравець обирає одну з трьох випадкових карт та як застосовуються її ефекти на характеристики персонажа.
5. Користувацький інтерфейс (UI) та статистика: Відображення ігрового інтерфейсу, що включає показники здоров'я гравця, номер поточної хвилі, кількість вбитих ворогів. Після завершення ігрової сесії (перемога над босом або поразка) демонструється екран зі статистикою, що показує

досягнення гравця (кращий час, кількість вбитих ворогів).

Наочна демонстрація підтвердила, що розроблений прототип відповідає поставленим функціональним вимогам і відображає основну ідею ігрового продукту. Візуальний та інтерактивний аспект дозволяє оцінити потенціал гри та її привабливість для цільової аудиторії, що є ключовим для подальшого залучення інвестицій та ресурсів.

4.2. Аналіз результатів та тестування

Після етапу реалізації та первинної демонстрації функціоналу, було проведено аналіз отриманих результатів та тестування розробленого прототипу гри. Метою цього етапу було підтвердження працездатності ключових ігрових механік та виявлення можливих недоліків.

Тестування проводилося шляхом ручної функціональної перевірки розробленого прототипу. Основна увага приділялася наступним аспектам:

1. Керування персонажем: Перевірялася плавність руху, коректність обробки вводу з клавіатури та миші, а також відсутність застрягань або непередбачуваних рухів.
2. Механіка атак гравця: Тестувалася можливість коректного використання магічних атак, їхня візуалізація та взаємодія з ворогами. Перевірялася коректність нанесення урону ворогам.
3. Поведінка ворогів: Аналізувалася логіка переслідування гравця, механіка їхніх атак та знищення після отримання достатнього урону.
4. Система спавну хвиль: Перевірялася коректність появи ворогів у нових хвилях, наростання їхньої кількості та/або сили з кожною хвилею, а також процедурна генерація місць спавну.
5. Система прогресії (карти покращень): Тестувалася коректність появи екрану вибору карт після хвилі, випадковість вибору карт, можливість їхнього обрання та правильне застосування ефектів до характеристик

гравця.

6. Користувацький інтерфейс: Перевірялася коректність відображення елементів UI (здоров'я, хвиля, вбивства), функціональність меню, екранів перемоги/поразки та статистики.
7. Збереження/завантаження даних: Тестувалася коректність збереження та завантаження ігрового прогресу (кращий час, вбиті вороги) між ігровими сесіями.
8. Звукове супроводження: Перевірялася наявність та коректність відтворення звукових ефектів відповідно до ігрових подій (атаки, отримання урону, спавн ворогів, вибір карт).

В процесі тестування було виявлено, що основні функціональні можливості, передбачені в проекті, працюють коректно. Розроблений прототип є стабільним робочим білдом, який демонструє основну ідею та концепцію гри. Всі критичні помилки, що могли б перешкоджати базовому ігровому процесу, були усунені. Незважаючи на те, що тестування проводилося вручну, воно підтвердило життєздатність проекту як демонстраційного зразка для подальшого вдосконалення та розвитку.

В процесі тестування було виявлено, що основні функціональні можливості, передбачені в проекті, працюють коректно. Розроблений прототип є працездатним, що демонструє основну ідею та концепцію гри. Всі критичні помилки, що могли б перешкоджати базовому ігровому процесу, були усунені. Незважаючи на те, що тестування проводилося вручну, воно підтвердило життєздатність проекту як демонстраційного зразка для подальшого вдосконалення та розвитку.

Отже в четвертому розділі було проведено демонстрацію та аналіз результатів розробленого прототипу 3D гри у жанрі Magic Survival. Демонстрація функціоналу підтвердила працездатність основних ігрових

механік, включаючи керування персонажем, магічні атаки, поведінку ворогів та систему їхнього спавну, а також механіку вибору карт покращень. Окремо було показано функціонування користувацького інтерфейсу (HUD, меню, екрани статистики) та системи локального збереження прогресу. Відеодемонстрація функціоналу проекту буде представлена під час захисту кваліфікаційної роботи.

Аналіз результатів та тестування, проведені шляхом ручної функціональної перевірки, засвідчили коректну роботу всіх основних компонентів гри. Прототип є працездатним і ефективно демонструє основну ідею та концепцію ігрового продукту. Всі критичні помилки, що могли б перешкоджати базовому ігровому процесу, були усунені, що підтверджує життєздатність проекту як демонстраційного зразка для подальшого вдосконалення та розвитку

ВИСНОВКИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

В процесі виконання даної кваліфікаційної роботи було успішно здійснено комплексний аналіз та практичну реалізацію демонстраційного прототипу 3D гри у жанрі Magic Survival з адаптивним геймплеєм та кастомізацією ігрових механік. Основна мета роботи – розробка функціонального прототипу, що демонструє ключові ідеї та потенціал продукту, – була повністю досягнута.

В ході роботи було:

1. Проаналізовано сучасний стан ігрової індустрії, особливості жанру *Magic Survival* та його актуальність на ринку.
2. Досліджено та обґрунтовано вибір ключових технологій та інструментів розробки, зокрема ігрового рушія Unity, мови програмування C#, середовища JetBrains Rider, а також Blender, Mixamo, Unity Animator, графічних редакторів та систем контролю версій GitHub та GitKraken.
3. Спроектовано архітектуру ігрового продукту, що базується на компонентно-орієнтованому підході Unity, з чітким розділенням на логічні модулі (гравця, ворогів, прогресії, UI, аудіо) та описано їхню взаємодію через систему подій.
4. Реалізовано основні ігрові механіки: керування гравцем, магічні атаки, поведінка ворогів, процедурний спавн хвиль з наростанням складності, систему прогресії та кастомізації через вибір покращень у вигляді карт, а також систему локального збереження даних за допомогою PlayerPrefs.
5. Проведена демонстрація та аналіз працездатності розробленого прототипу, що підтвердило його відповідність функціональним вимогам та життєздатність як основи для подальшого розвитку.

Розроблений прототип є працездатним і ефективно демонструє основну ідею та концепцію ігрового продукту. Всі критичні помилки, що могли б

перешкоджати базовому ігровому процесу, були усунені. Це дозволило отримати практичний досвід у розробці 3D ігор, поглибити знання у сфері геймдеву та застосувати сучасні підходи до проектування та реалізації програмних комплексів. Отримані результати свідчать про потенціал проекту як основи для подальшої комерційної розробки або для демонстрації концепції потенційним інвесторам та зацікавленим сторонам.

Перспективи подальшого розвитку проекту є значними та охоплюють наступні ключові напрямки:

1. Розширення контенту: Додавання нових типів ворогів, босів, магічних заклинань та покращень, що значно урізноманітнить ігровий процес та збільшить реграбельність.
2. Сюжетна лінія та квестова система: Інтеграція більш глибокого наративу, який може адаптуватися до дій гравця, та розробка системи завдань для підвищення занурення.
3. Система розвитку персонажа: Розширення механік кастомізації, додавання дерева навичок або системи артефактів для більш глибокої персоналізації ігрового досвіду.
4. Оптимізація та полірування: Подальша оптимізація продуктивності, покращення графіки та візуальних ефектів, а також доопрацювання користувацького інтерфейсу для забезпечення вищої якості та зручності.
5. Мультиплеєр: Дослідження можливостей додавання багатокористувацького режиму для спільної гри або змагань.
6. Розробка монетизаційних моделей: Для комерційного успіху прототипу необхідно розробити стратегії монетизації, наприклад, через внутрішньо ігрові покупки або преміум-контент.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Методичні вказівки та приклади оформлення:

1. GameDev. Analysis of Games by Engine in 2024. URL: https://gamedev.dou.ua/news/analysis-games-by-engine-2024/ (date of access: 04.10.2023).
2. YouTube. Kayak Plussize Jimmy and Jon Test Their Strength - YouTube, 2023. URL: https://www.youtube.com/watch?v=tU0BEwhjb5U&t=1s.
3. Учасники проектів Вікімедіа. Blender – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Blender (дата звернення: 11.06.2025).
4. Mixamo. *Mixamo*. URL: https://www.mixamo.com/#/ (date of access: 11.06.2025).
5. Bevor Sie zu YouTube weitergehen. URL: https://www.youtube.com/shorts/7BwZgeBsYuk (date of access: 16.06.2025).
6. Digital Hub. Advance Photoshop Training Class 1 | Digital Hub, 2022. *YouTube*. URL: https://www.youtube.com/watch?v=tfp2E8f10cw (date of access: 16.06.2025).
7. Unity Asset Store URL: https://assetstore.unity.com/search#q=3d&nf-ec_price_filter=0...0
8. Навчання перед монітором. GitHub: Реєстрація, 2025. *YouTube*. URL: https://www.youtube.com/watch?v=YYAun3Uwa1s (дата звернення: 16.06.2025).
9. GitKraken. GitKraken Desktop 11.1 Release: Auto-gen PR Title & Descriptions, Stash Messages, & Amend Commits, 2025. *YouTube*. URL: https://www.youtube.com/watch?v=jg3n_dg4IjM (date of access: 16.06.2025).
10. Що таке Trello та як ним користуватись » Блог хостера HOSTiQ.ua. *Блог хостера HOSTiQ.ua*. URL: https://hostiq.ua/blog/ukr/what-is-trello-2/?gad_source=1&gad_campaignid=22221308613&gbraid=0AAAAAC7A2B86DPGtk1dDdkghB5vdP57LU&

amp;gclid=CjwKCAjwgb_CBhBMEiwA0p3oOBLkr7L_MVhqVUbpfbLW5-u6MOyt8WL_yBn_oQaEtpPcLA0QbNV_bhoCOmgQAvD_BwE (дата звернення: 16.06.2025).

11. Schell J. Art of Game Design: A Book of Lenses, Second Edition. Taylor & Francis Group, 2017.
12. Вказано Н. Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих. Не вказано : Не вказано.