ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Комп'ютерна гра "Lost Element" 2D платформер із сайтом вікіпедії та використанням алгоритмів генерації залів, інтелектуальної поведінки ботів (комплексна робота)»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп’ютерні науки»
Освітня програма «Комп’ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Вадим КОЛЕСНИК
Керівник: зав. кафедри комп'ютерних наук
к.е.н., с.н.с, доцент
Сергій МІЧКІВСЬКИЙ

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп’ютерних наук
_____Сергій МІЧКІВСЬКИЙ
«____»____20___р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Колесник Вадим Олегович

Тема роботи	Комп'ютерна гра "Lost Element" 2D платформер із сайтом вікіпедії та використанням алгоритмів генерації залів, інтелектуальної поведінки ботів (комплексна робота)
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Розробити 2D комп’ютерну гру, платформер з використанням алгоритмів генерації залів, інтелектуальної поведінки ботів, а також UI/UX для гри. Розробити сайт вікіпедію з внутрішнім магазином, а також UI/UX для сайту.
Посилання на джерела інформації (не більше п’яти найменувань, які рекомендує науковий керівник)	1. Моделювання поведінки неігрових персонажів у комп'ютерних іграх : Н.О. СОКОЛОВА, В.В.ГНАТУШЕНКО, М.С. МІЩЕНКО, О.А. АТАМАНЧУК, Національний технічний університет «Дніпровська політехніка» 2. Чеботарьова І. Б., Черкашина Г. І. Основні тренди UI/UX дизайну 2024. Поліграфічні, мультимедійні та web-технології : матеріали Молодіжної школи-семінару IX Міжнар. наук.-техн. конф., 14-28 травня 2024 р. – Харків : ТОВ «Друкарня Мадрид», 2024. – Т. 2. – С. 40-47
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп’ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2024 р.

Керівник Сергій МІЧКІВСЬКИЙ

Здобувач освітнього ступня бакалавра Вадим КОЛЕСНИК

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
Основний етап			
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
Завершальний етап			
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Сергій МІЧКІВСЬКИЙ

Здобувач освітнього ступня бакалавра Вадим КОЛЕСНИК

Колесник В.О. Комп'ютерна гра "Lost Element" 2D платформер із сайтом вікіпедії та використанням алгоритмів генерації залів, інтелектуальної поведінки ботів (комплексна робота).

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 - Комп'ютерні науки (освітня програма - Комп'ютерні науки) СО Бакалавр. - ВНЗ "Університет економіки та права "КРОК", Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2025.

Розроблено комп'ютерну гру «Lost Element» – 2D платформер із сайтом вікіпедії та використанням алгоритмів генерації залів, інтелектуальної поведінки ботів.

Ключові слова: відеогра, розробка гри, 2D, Unity

Рис. 24. Бібліограф.: 29 найм.

Kolesnyk V.O. Computer game "Lost Element" 2D platformer with a Wikipedia site and the use of algorithms for generating halls, intelligent behavior of bots (complex work)

Project explanatory note by specialty 122 - Computer science. - "KROK" University, Educational and Scientific Institute of information and communication technologies, Department of Computer Science, Kyiv, 2025.

The computer game "Lost Element" has been developed - a 2D platformer with a Wikipedia site and the use of room generation algorithms and intelligent bot behavior.

Keywords: video game, game development, 2D, Unity

Fig. 24. Bibliography: 29 Items.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі цифрових технологій індустрія відеоігор стрімко набирає обертів та демонструє стабільне зростання. Особливу популярність останніми роками отримали саме 2D ігри, оскільки вони поєднують у собі простоту реалізації, зручність у використанні, цікаву візуальну стилістику та захопливий геймплей, який приваблює як новачків, так і досвідчених гравців. Створення таких ігор не вимагає надто великих ресурсів, тому є доступним варіантом для всіх заохочих.

Створення 2D піксельної гри в жанрі платформер під назвою «Lost Element» є актуальним і логічним напрямком для практичного дослідження в рамках навчального процесу. Проєкт дозволяє застосувати теоретичні знання на практиці, а також продемонструвати вміння з програмування, проєктування ігор, роботи з анімацією, побудови ігрової логіки та впровадження деяких алгоритмів.

Мета дослідження. Розробка гри у жанрі 2D платформер під назвою «Lost Element» та сайт вікіпедію до гри. У проєкті реалізується динамічна механіка руху, бойова система, штучний інтелект для NPC, генерація залів, система відродження та збереження, сайт вікіпедія.

Завдання дослідження:

- дослідити існуючі комп’ютерні ігри жанру платформер та визначити їх переваги та недоліки;
- створити концепт гри;
- спроектувати основну архітектуру гри та алгоритмів;
- розробити програмне забезпечення комп’ютерної гри «Lost Element» з використанням алгоритмів генерації залів та інтелектуальна поведінка ботів.

Об’єкт дослідження є ігровий процес комп’ютерної 2D гри в жанрі платформер.

Предмет дослідження комп’ютерні ігри та їх механіки, візуальні стилі 2D ігор в жанрі платформер. В досліджені здійснюється аналіз від ідеї до реалізації та тестування.

Методи дослідження. Підходи та алгоритми генерації залів та інтелектуальної поведінки ботів.

Практичне значення. Розроблена комп’ютерна 2D гра «Lost Element» в жанрі платформер зі сайтом вікіпедія, з використанням алгоритмів генерації залів та інтелектуальної поведінки ботів.

Структура роботи: Кваліфікаційна робота складається з вступу, трьох розділів, висновків та списку посилань (29 найменування). Пояснювальна записка містить 24 рисунків. Загальний обсяг пояснювальної записки становить 47 сторінок, основний зміст викладено на 42 сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ

1.1. Предметна область

Програмні продукти, що належать до категорії 2D ігор, створюються з метою забезпечення інтерактивної взаємодії між користувачем та віртуальним середовищем у двовимірному просторі. Основне призначення таких ігор полягає у створенні ігрового досвіду, що базується на ігрових механіках, візуальному стилі та сценарних елементах.

2D ігри мають низку переваг, серед яких: менші технічні вимоги до обладнання, відносна простота у створенні, висока гнучкість у реалізації художніх стилів та можливість зосередитись на ігрових механіках і сюжеті. Саме тому вони часто використовуються як у комерційних проєктах, так і в навчальних цілях, під час вивчення основ геймдизайну, програмування та комп’ютерної графіки.

Сфера застосування 2D ігор є досить широкою і охоплює такі напрямки:

Розважальні проєкти. Найбільш поширена сфера це створення ігор для розваг та відпочинку. 2D ігри тут займають окрему нішу завдяки своїй доступності, різноманітності жанрів (платформери, головоломки, аркади, RPG тощо) та ностальгічному стилю, що нагадує класичні ігри 80–90-х років.

Освітні продукти. Ігри активно використовуються як засіб для засвоєння нових знань і розвитку навичок. Завдяки простоті реалізації, викладачі або студенти можуть створювати власні освітні ігри навіть без великого досвіду у програмуванні. Unity, Godot, Construct та інші движки активно використовуються студентами для створення 2D ігор як частини дипломних чи курсових проєктів. Це дозволяє краще засвоїти принципи об’єктно-орієнтованого програмування, роботи з графікою, логікою взаємодії, фізикою та інтерфейсами користувача.

Реклама та маркетинг. Багато компаній замовляють невеликі 2D ігри для популяризації бренду чи продукції. Такі ігри можуть бути частиною рекламної

кампанії, інтерактивними банерами або окремими застосунками для мобільних платформ.

Терапевтичні ігри. Існують 2D ігри, спеціально розроблені для реабілітації, зниження рівня стресу або підтримки психоемоційного стану користувача. Вони можуть включати прості дії, заспокійливу музику та завдання, що сприяють розслабленню.

Культурні та художні проєкти. Деякі 2D ігри є носіями культурного чи філософського змісту. Через просту графіку автори мають змогу зосередитись на художній ідеї, передавати емоційні стани, осмислювати складні теми, а іноді навіть розповідати історії через інтерактивну форму мистецтва.

Таким чином, 2D ігри не обмежуються лише функцією розваги. Їх використання поширене у навчанні, науці, культурі, медицині та бізнесі. Вони слугують ефективним інструментом передачі інформації, розвитку навичок і створення емоційного зв'язку з користувачем. Саме завдяки універсальності та простоті реалізації 2D ігри залишаються актуальними навіть у сучасному світі високих технологій та тривимірної графіки.

1.2. Огляд аналогів

Щоб зрозуміти місце «Lost Element» серед інших платформерів, важливо проаналізувати ринок подібних ігор. У жанрі 2D-платформерів існує велика кількість успішних проєктів, що вплинули на формування сучасного геймдизайну.

Hollow Knight – одна з найпопулярніших метроїдваній, що пропонує глибоку історію, ретельно опрацьований світ і складну бойову систему. Ця гра відзначається атмосферністю та незабутньою візуальною стилістикою. (рис. 1.1) [1].

У грі гравець керує маленьким безіменним лицарем, який досліджує підземний світ Халлоунеста — занедбаного королівства, що колись було могутнім, а тепер наповнене небезпеками, ворогами та секретами. Головна

увага приділяється дослідженню, битвам і платформінгу. Гра має складну, але чесну бойову систему, цікаву нелінійну структуру та багатий візуальний стиль, виконаний у ручній анімації. Hollow Knight є прикладом якісної інді-розробки, яка отримала визнання як серед гравців, так і серед критиків.

Рисунок 1.1 – Гра «Hollow Knight»

Джерело [1].

Celeste – платформер, який фокусується на складних паркур елементах. Його особливістю є точне управління та інтригуючий сюжет, який змушує гравця заглиблюватися у внутрішній світ персонажа. Гра розповідає історію дівчини на ім’я Медлін. Вона намагається піднятися на вершину гори Селеста, долаючи не лише фізичні перешкоди, але й власні психологічні труднощі. Геймплей зосереджений на точному управлінні, стрибках і використанні здібності ривка. Хоча гра має високий рівень складності, вона мотивує гравця через підтримуючий сюжет і добре продуманий дизайн рівнів. Celeste відома своїм глибоким сенсом і якісним геймдизайном, що доводить, що інді-ігри

можуть зачіпати серйозні теми й залишатися при цьому захопливими. (рис. 1.2) [2].

Рисунок 1.2 – Гра «Celeste»

Джерело [2].

Dead Cells – гра, яка поєднує динамічні бої з процедурно генерованими рівнями. Вона пропонує великий вибір зброї та можливість адаптувати стиль гри під власні вподобання. Гравець керує істотою, яка вселяється у тіла загиблих воїнів, намагаючись втекти з постійно змінюваного острова. Основні особливості гри — динамічні бої, процедурно згенеровані рівні та постійне вдосконалення навичок після кожної спроби. Гра поєднує елементи метроїдванії та roguelike, дозволяючи гравцю відкривати нові шляхи та покращення з кожним проходженням. Dead Cells демонструє, як інноваційні механіки та стиль піксельної графіки можуть поєднуватися у високоякісний геймплей. (рис. 1.3) [3].

Shovel Knight – класичний платформер у піксельному стилі, що нагадує ігри 90-х. (рис. 1.4) [4].

Рисунок 1.3 – Гра «Dead Cells»

Джерело [3].

Рисунок 1.4 – Гра «Shovel Knight»

Джерело [4].

Його бойова система та рівні натхненні ретро-іграми, що робить його привабливим для ностальгуючих геймерів. Головний герой лицар із лопатою,

який вирушає у пригоду, щоб врятувати свою кохану і перемогти злу Чародійку. Гра натхненна класикою 8-бітної епохи, такою як Mega Man та Castlevania, але має сучасні елементи управління, збереження прогресу та глибину геймплею. Shovel Knight отримала багато позитивних відгуків за стильну графіку, музику та відданість духу старих ігор, при цьому залишаючись цікавою для нової аудиторії.

1.3. Визначення потенційних конкурентних переваг розробки

Конкурентні переваги «Lost Element» визначають її привабливість для гравців та виділяють серед інших платформерів. Основні особливості гри, що можуть забезпечити успіх проєкту:

Оригінальний сюжет – на відміну від багатьох платформерів, що зосереджуються лише на геймплеї, «Lost Element» має добре продуману історію головного персонажу. (рис 1.5).

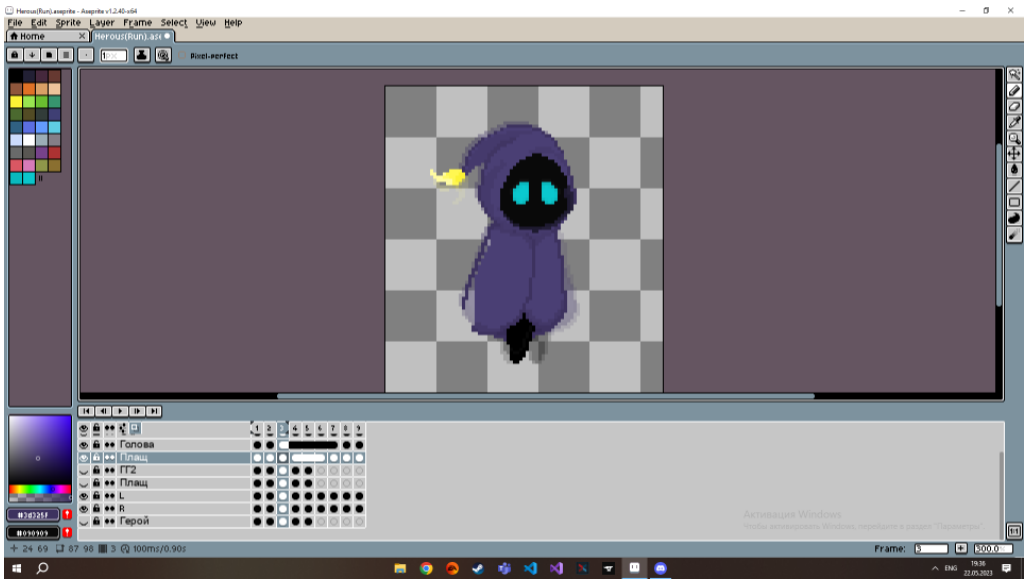

Рисунок 1.5 – Головний герой гри «Lost element»

Джерело розроблено автором

Стилізована піксельна графіка створює унікальну атмосферу та додає грі впізнаваний стиль, що приваблює фанатів багатьох аналогів.

Комбінація платформінгу та бойових елементів. Гра пропонує як складні стрибкові секції, так і динамічні сутички з ворогами, що забезпечує різноманітність геймплею.

Глибока система бою – персонаж має кілька типів ударів, що робить бої більш тактичними та динамічними.

Ці характеристики дозволяють «Lost Element» конкурувати з іншими платформерами, пропонуючи унікальний ігровий досвід. (рис 1.6)

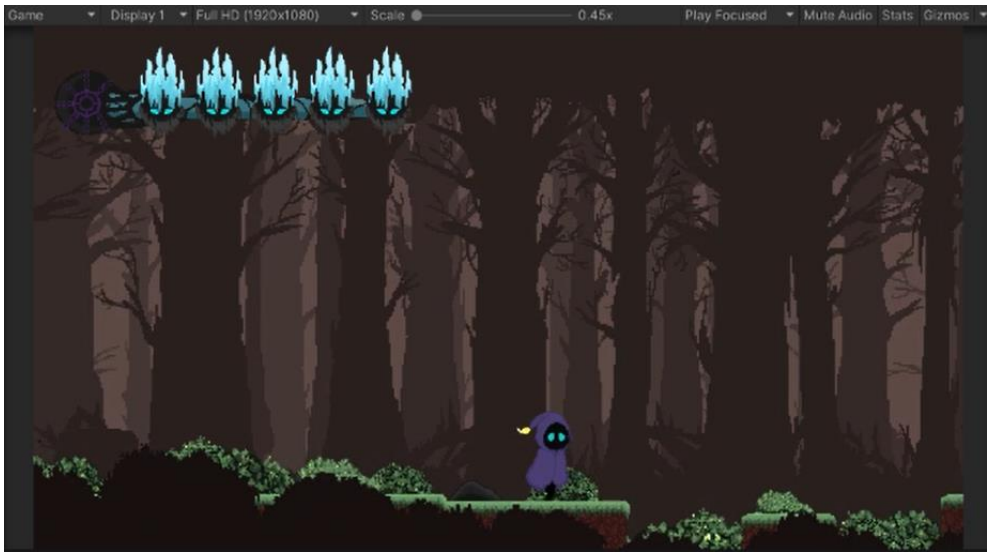

Рисунок 1.6 – Ігровий момент в «Lost element»

Джерело розроблено автором

1.4. Постановка задачі

Гра «Lost Element» має базові механіки: ходьба, стрибки, біг та атака ближнього бою. Дослідження локацій: гравець зможе самостійно обирати маршрути для проходження. Локації будуть мати кімнати з загадками, які персонаж повинен вирішити щоб отримати нагороду. Також будуть присутні секретні кімнати, знайти їх можна досліджуючи локацію. Взаємодія: гравець може взаємодіяти з деякими елементами або NPC. Додаткові механіки: система збереження буде реалізована у вигляді точок з якими гравець має

взаємодіяти. При загибелі персонажу, його буде перенесено до останньої точки збереження.

Геймплей передбачає проходження основних локацій, кожна з яких має свої унікальні виклики. У цих локаціях гравець стикається з основами механіки гри: потрібно пересуватися платформами, стрибати через прірви, уникати падінь у хащі або агресивно налаштованих NPC та вчитися контролювати персонажа, щоб не витрачати «життя» що може призвести до закінчення гри.

Гравець керує персонажем за допомогою таких клавіш:

A – рух ліворуч, D – рух праворуч, Space – стрибок, Q – удар.

Також під всі дії персонажу є своя анімація головного персонажу, що робиться в Unity. (рис 1.7).

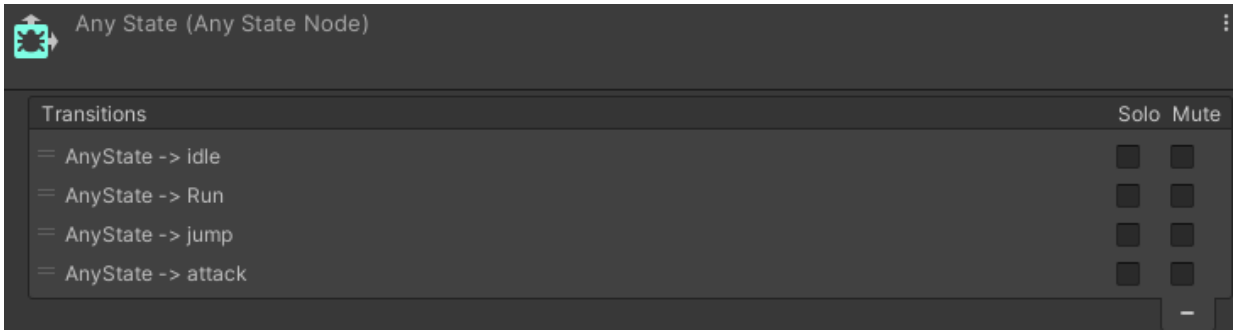

Рисунок 1.7 – Анімації головного персонажу

Джерело розроблено автором

Учасниками гри люди віком від 15-35 років, до їхнього числа можуть входити як фанати ретро та платформерів, так і люди, які полюбляють дослідження та відкритий світ.

Система життів представлена у вигляді 5 життів, які поступово втрачаються у разі отримання шкоди або падіння в прірву. Якщо персонаж падає за межі карти або втрачає все свої життя, з’являється спеціальне меню з

можливістю respawn або виходу в menu. Це дозволяє гравцю не починати гру спочатку, а повернутися до попереднього контрольного пункту.

Бойова система включає можливість ближнього бою, що дозволяє атакувати противників за допомогою магічних ударів. Вороги мають різні типи поведінки та рівні складності, що вимагає від гравця адаптації до тактики бою.

Ігровий процес доповнюється інтерактивними елементами, такими як відеовставки, які допомагають розкрити сюжет і краще зрозуміти мотиви головного героя. Завдяки детальній піксельній графіці та добре опрацьованому саундтреку гра створює унікальну атмосферу, що сприяє глибшому зануренню у світ «Lost Element».

Функціональні вимоги:

- персонаж може взаємодіяти з об’єктами;
- користувач може керувати головним персонажем;
- ворожий NPC попадаючи атакою по головному герою знімає здоров’я;
- система збереження;
- після втрати всього здоров'я персонаж відроджується на останній точці збереження;
- головне меню та HUD персонажу.

Нефункціональні вимоги:

- кожна дія має звуковий супровід;
- зміна звукового супроводу в залежності від локації або місця;
- гра на операційній системі Windows;
- гра повинна працювати стабільно та не мати критичних помилок, які б зупиняли проходження гри;
- підтримка клавіатури.

Взагалом в проєкті реалізовуються такі автоматизовані функції:

- зміна стану анімацій персонажу;
- оновлення карти при дослідженні;

- генерація локацій при новій грі;
- поява ворогів у відповідних зонах;
- поведінка ворогів: патрулювання, атака, переслідування;
- збереження статусу ворогів – вбиті вороги не відроджуються, поки гравець не взаємодіє з об'єктом збереження;
- спеціальні тригери подій для зустрічі з босом та запуску кат-сцени;
- система діалогів.

Висновки до розділу 1

Проведено дослідження предметної області, аналіз напрямків та вибір напрямку для проєкту, а також пояснення інших напрямків.

Проведено аналіз потенційних конкурентів, їх опис, а також переваги та недоліки майбутнього продукту.

Визначені та описані основні функціональні та нефункціональні вимоги, вхідні та вихідні дані, автоматизовані функції.

РОЗДІЛ 2
ПРОЄКТУВАННЯ

2.1 Моделювання поведінки

Діаграма прецедентів (Use Case Diagram) показує візуальне зображення основних функціональних характеристик продукту та методів взаємодії користувача із системою. Діаграма дозволяє зрозуміти, які конкретні дії будуть доступні для користувачів, а також які варіанти поведінки система повинна бути здатна підтримувати. (рис 2.1).

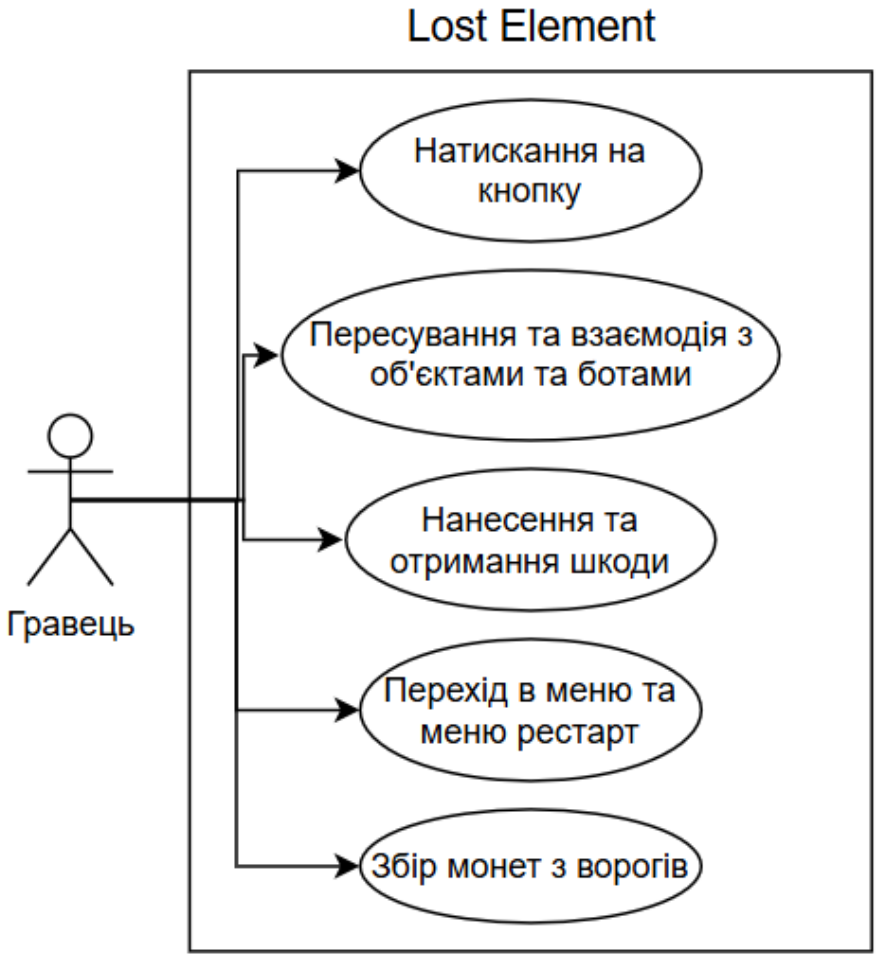

Рисунок 2.1 – Use Case Diagram (Lost Element)

Джерело розроблено автором

Інструкція до рисунку 2.1:

- гравець – актор, який взаємодіє з системою;
- стрілка – пов’язує актора з функцією, з якою взаємодіє;
- овал – основна функція.

У діаграмі показано, що гравець, а тобто актор має такі взаємодії з системою гри:

- Гравець має доступ до базових механік: ходьба, біг, стрибок, атака. Всі дії виконуються при натисканні на кнопки;
- Актор може взаємодіяти: з другорядними персонажами для активації діалогу та об’єктами локації, які потрібні для вирішення загадок або для збереження гри;
- Атакуючи ворожого бота (NPC) гравець наносить шкоду, ворожий бот атакуючи знімає життя героя;
- Гравець має деякий HUD: кнопки щоб вийти в меню або вийти з гри та меню для рестарту при поразці.

Діаграма діяльності (Activity Diagram) відображає послідовність дій, що виконуються під час гри. Ця діаграма дозволяє наочно продемонструвати логіку ігрового процесу, зокрема поведінку головного персонажа під час взаємодії з оточенням (рис 2.2).

В діаграмі показано як актор, ворожий бот (NPC) починає зі стану патрулювання, у грі бот ходить між двома точками. Коли головний персонаж входить в деякий діапазон біля бота, бот премикає стан на переслідування, якщо головний герой виходить з діапазону бот знову повертається в стан патрулювання. В стані переслідування бот ходить за героєм і може вийти зі свого діапазону патрулювання, поки не зазнає критичної шкоди або герой не вийде з діапазону видимості головного персонажу. В бота є деякий запас життів як і у головного героя, якщо запас життів буде 0 бот загине.

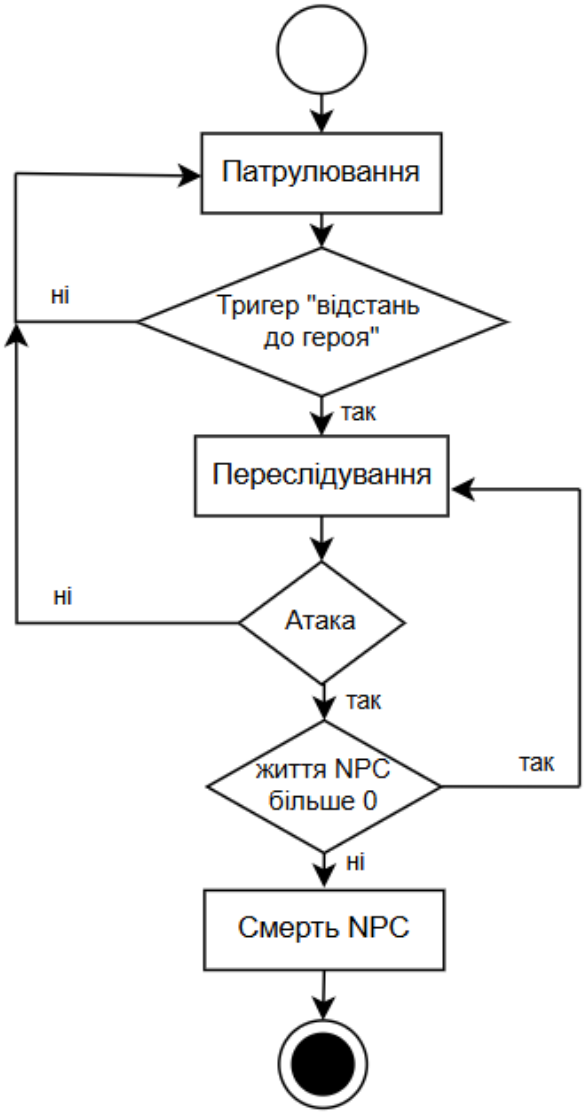

Рисунок 2.2 – Diagram Activity

Джерело розроблено автором

Інструкція до рисунку 2.2:

- коло – актор, в даному випадку актор є ворожим ботом;
- прямокутник – функція, яку виконує актор;
- ромб – перевірка на дію (так чи ні?);
- стрілка – актор задає дію;
- полукруг – кінцева точка гри.

2.2 Моделювання поведінки

Діаграма класів (Class Diagram) є одним із основних інструментів структурного моделювання в рамках мови UML. Діаграма відображає логічну архітектуру системи, а саме: основні класи, їхні атрибути, методи, а також взаємозв'язки між ними.

Основна мета побудови діаграми класів це забезпечення візуального уявлення про об'єктно-орієнтовану структуру проєкту. Це дозволяє більш чітко спланувати подальшу реалізацію функціоналу, а також уникнути дублювання коду, зменшити залежність між компонентами та підвищити масштабованість застосунку (рис 2.3).

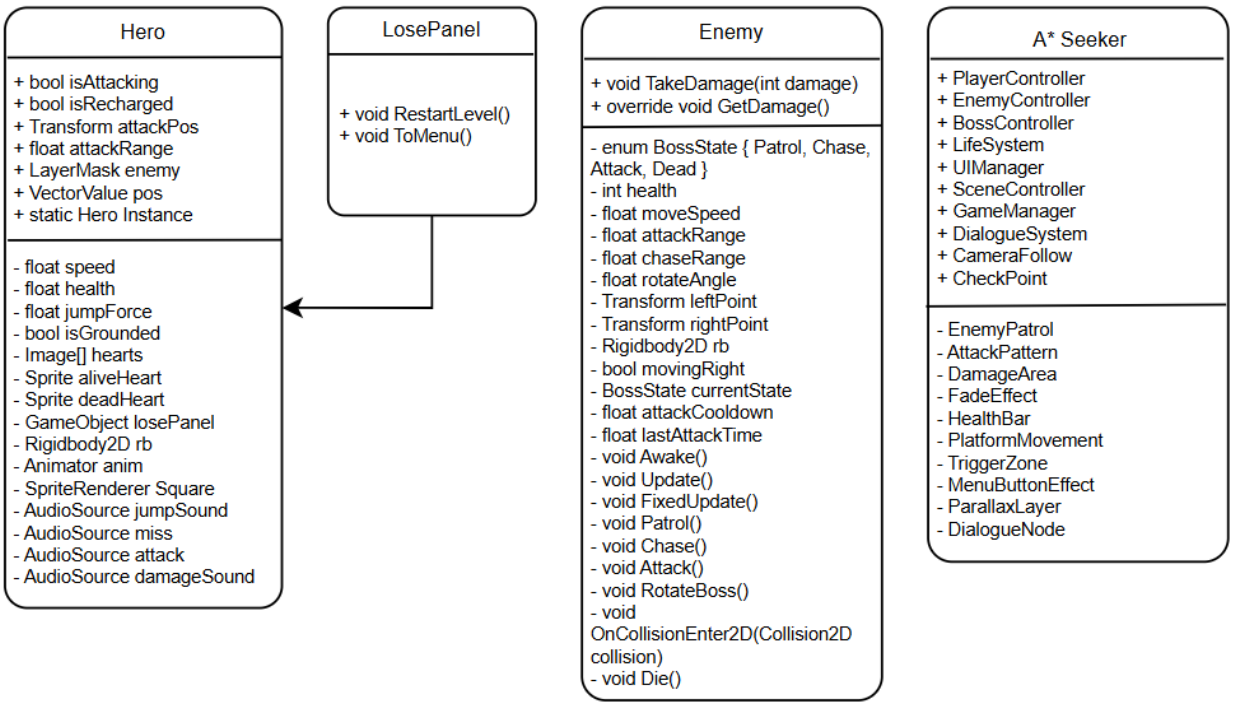

Рисунок 2.3 – Class Diagram

Джерело розроблено автором

Інструкція до рисунку 2.3:

- квадрат – це функцію яку задає гравець;
- стрілка – це коди які прикріпленні до батьківського об'єкта;

- + - це публічна змінна;
- - - це приватна змінна.

Рисунок показує головний код героя Hero до якого прикріплений дочірній об’єкт LosePanel.

Hero – головний ігровий персонаж. Цей клас відповідає за логіку руху, стрибків, атаки, отримання пошкоджень, а також взаємодію з елементами інтерфейсу та звуками: (додаток А.1)

- static Hero Instance — об'єкт героя в сцені, який дає можливість іншим скриптам взаємодіяти з героєм (наприклад, змінювати його здоров'я чи перевіряти його стан);
- float speed — швидкість переміщення героя, яка впливає на його переміщення по сцені;
- float health — поточне здоров'я героя, яке можна змінювати в процесі гри;
- float jumpForce — сила стрибка героя, визначає, на яку висоту герой може стрибнути;
- bool isGrounded — перевірка, чи стоїть герой на землі або на платформі. Важливо для запобігання виконання стрибка в повітрі;
- Image[] hearts — масив, який використовується для відображення сердець на інтерфейсі користувача, що показують здоров'я героя;
- Sprite aliveHeart, deadHeart — спрайти, які відображають серце героя в різних станах: живе і мертве;
- GameObject losePanel — панель, що з’являється, коли герой помирає, і пропонує варіанти перезапустити рівень або повернутися в головне меню;
- Rigidbody2D rb — фізичне тіло героя, яке відповідає за взаємодію з фізичними об'єктами на сцені (наприклад, сила тяжіння);
- Animator anim — аніматор для відображення анімацій героя, наприклад, для бігу, стрибка чи атаки;

- AudioSource jumpSound, miss, attack, damageSound — звукові ефекти для різних дій героя (стрибок, атака, отримання шкоди тощо);
- void RestartLevel() — метод для перезапуску рівня після смерті героя;
- void ToMenu() — метод для повернення на головне меню гри після смерті героя.

LosePanel — клас, що керує інтерфейсом поразки. Відображає UI при втраті всіх життів:

- RestartLevel() — перезапускає рівень;
- ToMenu() — повертає в головне меню.

Цей клас взаємодіє з Hero, коли гравець програє.

Далі на рисунку можна побачити клас ворога Enemy.

Enemy — базовий ворог у грі. Відповідає за рух, стан, отримання та нанесення шкоди: (додаток А.2)

- void TakeDamage(int damage) — метод для прийому шкоди від героя або іншого об'єкта в грі;
- void GetDamage() — метод для обробки отриманої шкоди, наприклад, зменшення здоров’я;
- void Awake() — метод для ініціалізації ворога в момент його появи на сцені;
- void Update() — метод для оновлення логіки ворога кожного кадру (наприклад, рух чи атака);
- void FixedUpdate() — метод для обробки фізичних взаємодій, наприклад, рух ворога за допомогою Rigidbody2D;
- void Patrol() — метод для патрулювання ворога в межах певної зони;
- void Chase() — метод для переслідування героя, якщо він потрапляє в зону агресії ворога;
- void Attack() — метод для атакування героя або іншого об'єкта;

- void RotateBoss() — метод для обертання ворога (можливо, босса) в напрямку героя або іншого об'єкта;
- Transform leftPoint, rightPoint — точки, між якими ворог пересувається під час патрулювання;
- float health, moveSpeed, attackRange, chaseRange, rotateAngle — змінні для здоров’я, швидкості руху, діапазону атаки, діапазону переслідування та кута повороту ворога;
- Rigidbody2D rb — фізичне тіло ворога, яке відповідає за його рухи та зіткнення з іншими об'єктами;
- BossState currentState — поточний стан ворога (наприклад, патрулювання, переслідування, атака);
- float attackCooldown — час між атаками ворога.

Останній код A* Seeker це алгоритм A*, який буде описаний в архітектурі продукту:

- PlayerController — керує рухом, анімацією та діями гравця. Приймає ввід з клавіатури або контролера;
- EnemyController — централізовано керує логікою всіх ворогів: рухом, атаками, поведінкою;
- BossController — клас для керування босом: його фазами, атаками, логікою переходів між станами (Patrol, Attack, Dead тощо);
- LifeSystem — відповідає за систему життів гравця. Може оновлювати інтерфейс, перевіряти смерть, викликати LosePanel;
- UIManager — керує усіма елементами UI: оновлення життів, очок, повідомлень, панелей тощо;
- SceneController — здійснює завантаження, перезапуск та перехід між сценами гри;
- GameManager — головний менеджер гри. Відповідає за логіку рівня, збереження станів, перемогу або поразку, паузу;
- DialogueSystem — показує діалоги, взаємодіє з персонажами та гравцем, може призупиняти геймплей під час діалогу;

- CameraFollow —реалізує слідування камери за гравцем з певним лагом або межами;
- CheckPoint — точка збереження. Гравець повертається до неї після смерті.

2.3 Опис архітектури продукту

Архітектура програмного проєкту базується на принципах об'єктно-орієнтованого програмування з використанням компонувальної моделі, що реалізується в середовищі Unity.

На першому етапі розробки було створено базову структуру проєкту з використанням Unity. До проєкту було додано необхідні скрипти, спрайти та звукові ефекти. Створено основну сцену гри, яка виступає початковою локацією. В межах цієї локації реалізовано стартову точку появи гравця (спавн), а також побудовано платформи, кожна з яких має компонент BoxCollider2D, що відповідає за фізичну взаємодію з об'єктами середовища.

Другий етап передбачав додавання головного героя на ігрову сцену. До об’єкта гравця було прикріплено компонент Rigidbody2D для забезпечення фізики руху, а також написано скрипт керування, який відповідає за обробку натискань клавіш і взаємодію з навколишнім середовищем. Цей етап є критично важливим, оскільки вся логіка поведінки персонажа реалізується саме через відповідний скрипт.

На третьому етапі було реалізовано головне меню гри як окрему сцену. Меню включає дві кнопки:

- Play — кнопка, яка здійснює перехід до основної ігрової сцени;
- Exit — кнопка, що завершує виконання програми.

Окрім елементів інтерфейсу, сцена головного меню містить фонове звукове оформлення, здебільшого розроблене в FL Studio20.

Четвертий етап розробки був присвячений створенню додаткових ігрових сцен із випадковою генерацією просторових структур (залів). Для

цього було заздалегідь підготовлено декілька варіантів конфігурацій об'єктів, які розміщуються у сцені за допомогою скрипту.

На п’ятому етапі було реалізовано ворожих ботів (NPC) та босів із базовим штучним інтелектом. Їх функціонал включає: патрулювання певної зони, виявлення гравця при вході в деякий діапазон, переслідування, виконання атак різних типів, а також смерть у разі повністю витрачених життів.

Реалізовано алгоритм A* (A-star) в деяких ботів, один із найбільш популярних і продуктивних алгоритмів для знаходження найкоротшого шляху у графі або двовимірному просторі. Алгоритм A* був обраний за аналізом інших аналогів алгоритму.

Алгоритм Дейкстри (Dijkstra’s Algorithm) шукає найкоротший шлях від стартової точки до всіх інших без використання евристик.

Переваги:

- гарантує знаходження оптимального шляху;
- простий для реалізації.

Недоліки:

- повільніший за A*, оскільки перевіряє всі вузли, навіть ті, що далекі від цілі;
- неефективний у великих або складних рівнях.

2. Жадібний пошук по найкращій оцінці (Greedy Best-First Search) обирає вузол із найменшою евристичною відстанню до цілі.

Переваги:

- працює швидше за Дейкстру в багатьох випадках;
- простий у реалізації.

Недоліки:

- не гарантує знаходження найкоротшого шляху;
- може визначити неправильних шлях через неточну евристику.

3. Breadth-First Search (Пошук у ширину) перевіряє всі сусідні вузли на одному рівні перед переходом до наступного.

Переваги:

- надійний для невеликих карт;
- завжди знаходить найкоротший шлях (у непозначених графах).

Недоліки:

- дуже повільний при великих розмірах карти;
- не використовує евристику, отже не ефективний для складних рівнів.

4. Depth-First Search (Пошук у глибину) рухається в одному напрямку до кінця, поки не знайде ціль.

Переваги: потребує менше пам’яті.

Недоліки:

- не гарантує найкоротшого шляху;
- може застрягнути в глибоких гілках графа.

A* є евристичним алгоритмом пошуку, який поєднує в собі переваги алгоритму Дейкстри (який гарантує знаходження найкоротшого шляху) та жадібного пошуку по евристиці (який пришвидшує процес за рахунок оцінки відстані до цілі). Основна ідея алгоритму полягає у використанні функції оцінки вартості шляху для кожної потенційної вершини. Ця функція обчислюється за формулою:

$$f(n) = g(n) + h(n)$$

де

- g(n) — вартість шляху від стартової точки до поточної вершини n;
- h(n) — евристична оцінка вартості шляху від n до цільової точки (наприклад, за допомогою манхеттенської або евклідової відстані);
- f(n) — сумарна оцінка вартості проходження через вершину n.

Алгоритм A* було використано для ботів, які виконують функцію переслідування гравця, особливо в умовах складної геометрії рівнів, де персонаж і противник розділені стінами, платформами або іншими об’єктами.

Застосування A* дозволило ботам будувати оптимальний маршрут до гравця навіть у тих випадках, коли шлях є непрямим і вимагає серії обхідних дій.

Реалізація алгоритму в Unity базується на розбитті ігрового простору на grid, де кожна клітинка є окремим вузлом графа. Кожен вузол зберігає інформацію про свою прохідність, координати, а також значення функцій g, h та f. На початку роботи алгоритму відкривається так званий список відкритих вузлів (Open List), куди додається стартова точка. На кожній ітерації вибирається вузол з найменшим значенням f, після чого алгоритм аналізує його сусідів, обчислює для них відповідні значення g, h, f, і додає їх до Open List, якщо ті ще не були опрацьовані. Паралельно формується список закритих вузлів (Closed List), щоб уникнути повторного аналізу одних і тих самих точок.

Завдяки такій логіці обчислень A* дозволяє ботам адаптивно реагувати на динамічні умови гри: зміну положення гравця, поява або зникнення перешкод, загибель інших ворогів тощо. Водночас алгоритм гарантує, що шлях до цілі, якщо він існує, буде знайдений із мінімальними витратами.

У реалізації використовувалася манхеттенська метрика як евристична функція h(n), оскільки вона ідеально підходить для двовимірної сітки з ортогональними переміщеннями. Таким чином алгоритм A* шукає оптимальний маршрут та бот може пересуватись в 8 різних сторін.

Алгоритм A* було здійснено в окремий клас, який надає метод FindPath(startNode, targetNode) та повертає список вузлів, які утворюють найкоротший шлях. Цей шлях зберігається в черзі та поступово обробляється ІІ-агентом. Бот переходить до наступного вузла в черзі, щойно наближається до поточного на задану відстань. Такий підхід забезпечує гладке та природне переміщення ворожого персонажа, мінімізуючи ризик застрягання в платформах або нелогічної поведінки.

На шостому етапі було створення сайту вікі (LostElementWIKI), який розміщується на комп'ютері і створений за допомогою XAMPP (набір для створення локального сервера сайту). Код сайту написаний мовою PHP. Він

має простий інтерфейс та дизайн, з великою кількістю інформації про сюжет гри та допоміжними статтями для проходження гри та управління героєм.

Висновок до розділу 2

За допомогою діаграми Use Case вдалося визначити взаємодію гравця з основними ігровими об'єктами, такими як герой, меню, вороги, локації. Діаграма дозволяє візуалізувати логіку взаємозв'язків між системою та користувачем, підвищуючи зручність для подальшого проектування та тестування функціональності.

Використання Activity Diagram дозволило структурувати весь ігровий процес на окремі активності: запуск гри, завантаження сцени, переміщення героя, виявлення колізій, взаємодія з ворогами, зміна локацій.

Діаграма класів стала ключовим інструментом для моделювання структури гри з точки зору об'єктно-орієнтованого підходу. У цьому підрозділі було проаналізовано основні класи, які формують логіку гри «Lost Element», їх атрибути, методи та зв'язки. Було визначено класи для таких об'єктів: Hero, LosePanel, Enemy та A* Seeker.

Впроваджено алгоритм пошуку шляху A* (A-star) та аналіз аналогів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ГРИ LOST ELEMENT

3.1 Особливості реалізації та конструювання програмного продукту

Мова програмування. На початку створення гри було прийнято рішення використовувати мову програмування C#, яка є однією з найефективніших та найпоширеніших мов для роботи з Unity. Основна причина такого вибору полягає в значному досвіді використання C#, а також її широкій підтримці у середовищі розробки Unity, що значно спрощує інтеграцію логіки гри з її візуальною та фізичною частинами. Також C# відзначається добрим балансом між продуктивністю і зручністю використання, що дозволяє ефективно реалізовувати системи керування, візуалізації, взаємодії з користувачем, а також алгоритми штучного інтелекту.

Аналіз альтернатив:

1) Java.

Переваги:

- кросплатформеність: код на Java працює практично на будь-якій ОС завдяки JVM;
- велика спільнота та багатий вибір бібліотек.

Недоліки:

- слабка підтримка в Unity: Java не є стандартом для ігрового рушія Unity, що ускладнює інтеграцію;
- гірша продуктивність для ресурсомістких ігор, особливо в обробці графіки в реальному часі.

2) Python.

Переваги:

- Простий синтаксис, що робить його ідеальним для швидкого прототипування;
- Активна спільнота та численні фреймворки.

Недоліки:

- Низька продуктивність для графічно насичених ігор;
- Відсутність прямої підтримки.

Середовище розробки. Як основний рушій для реалізації гри було обрано Unity Engine, потужне програмне середовище, призначене для створення інтерактивного контенту. Однією з найсуттєвіших переваг Unity є його модульність та відкритість до розширення за допомогою скриптів на C#, що дає змогу максимально налаштовувати поведінку ігрових елементів.

Unity підтримує візуальне редагування сцен, має гнучку систему компонентів, яка дозволяє будувати ієрархії об'єктів та швидко змінювати їхні властивості. Крім того, рушій має зручну інтеграцію з системами анімації, фізики, частинок, UI-компонентів та аудіо, що суттєво прискорює процес розробки.

Значну роль відіграє також активна спільнота користувачів Unity, а також доступ до великої кількості навчальних матеріалів і прикладів, які допомагають швидко знаходити рішення для різних задач.

Аналіз альтернатив:

1) Unreal Engine.

Переваги:

- Потужна графічна система, особливо для 3D-ігор (наприклад, Nanite, Lumen);
- Сильна підтримка C++ і візуального скриптингу через Blueprints.

Недоліки:

- Високий поріг входу для початківців;
- Значно вищі системні вимоги для розробки та тестування.

2) Godot Engine

Переваги:

- Легка вага та відкритий вихідний код (open-source);
- Проста структура проекту, зручна для 2D-ігор.

Недоліки:

- Відсутність повної підтримки C# у деяких версіях, особливо на мобільних платформах;
- Менш зріла екосистема, порівняно з Unity.

Платформа

Розробка гри здійснювалася з орієнтацією на операційну систему Windows, оскільки це дозволяє найшвидше реалізувати і протестувати основну механіку гри. Проте завдяки широким можливостям Unity щодо експорту проєкту на різні операційні системи, планується розширення функціоналу та адаптація гри для мобільних платформ — зокрема Android та iOS. Гнучкість Unity у плані кросплатформеності полягає в наявності готових інструментів для компіляції під різні архітектури, що дозволяє мінімізувати витрати часу при перенесенні гри на нову платформу. Це робить проєкт більш перспективним для майбутнього розвитку та розширення цільової аудиторії.

Бібліотеки та інструменти

Unity Engine — це основна платформа, яка використовувалась для створення гри. Вона надає потужні можливості для інтеграції графіки, фізики, анімацій та звукового супроводу в реальному часі. Завдяки широкій підтримці мультимедійного контенту, Unity дозволяє реалізувати як базові, так і складні функціональні компоненти гри без необхідності підключення сторонніх інструментів.

Cinemachine — інструмент для кінематографічного керування камерою. Дана бібліотека автоматизує переміщення камери, дозволяє налаштовувати її поведінку залежно від подій у грі, що забезпечує плавність рухів та додає динаміки при зміні сцен або під час боїв.

TextMesh Pro — надбудова для роботи з текстовою інформацією у Unity. Вона дозволяє застосовувати розширене форматування, ефекти накладення, тіней і стилізацію тексту, що значно покращує естетику інтерфейсу гри.

Tilemap — функціонал Unity, що дозволяє створювати двовимірні карти з використанням плиток. Це спрощує процес побудови рівнів, дозволяє

оперативно змінювати структуру локацій і знижує кількість помилок у ручному розміщенні об’єктів.

A Pathfinding Project* — зовнішня бібліотека для реалізації пошуку оптимального шляху. Вона широко застосовувалася в рамках розробки штучного інтелекту в грі, зокрема для забезпечення логіки руху ворогів та NPC, які повинні орієнтуватися в межах складних рівнів.

Physics2D — вбудований компонент Unity, призначений для симуляції фізичних взаємодій у двовимірному просторі. За його допомогою реалізовано зіткнення, падіння об'єктів, стрибки та інші елементи, що наближають поведінку об'єктів у грі до реального фізичного світу.

[SerializeField] — спеціальний атрибут, що надає можливість відображати приватні змінні в інспекторі Unity. Його використання дозволяє зберігати інкапсуляцію в коді та водночас забезпечити зручне редагування важливих параметрів — наприклад, швидкості гравця, рівня здоров’я чи часу затримки анімації.

Програми та інструменти для графіки та звуку

Aseprite — програма, яка була використана для створення піксельної 2D-графіки. Вона зручна для створення циклічних анімацій, обробки кадрів спрайтів, а також надає можливість швидко редагувати велику кількість графічних елементів.

Adobe Photoshop — використовувався для роботи з деталізованими текстурами, а також для створення фонових зображень і елементів інтерфейсу. Цей потужний редактор дозволяє застосовувати фільтри, маски, шари та інші інструменти професійної обробки.

Tiled Map Editor — допоміжний редактор, який дозволяє створювати карти з тайлів незалежно від середовища Unity. Його перевага полягає у гнучкому розподілі шарів, можливості призначати властивості для кожної плитки та експортувати карти у форматі, сумісному з Unity.

Audacity — вільне програмне забезпечення для роботи зі звуковими доріжками. У процесі розробки гри воно використовувалось для обрізання,

нормалізації, фільтрації та накладання ефектів на звуки, які були використані для озвучення ігрового світу, взаємодій та бойових сцен.

FL Studio 20 — це професійна цифрова аудіо-робоча станція (DAW), яка була використана під час розробки гри для створення та обробки музичного супроводу. Завдяки широкому набору вбудованих інструментів, плагінів і ефектів, FL Studio 20 дозволяє створювати унікальні композиції, що підкреслюють атмосферу гри. (рис 3.1)

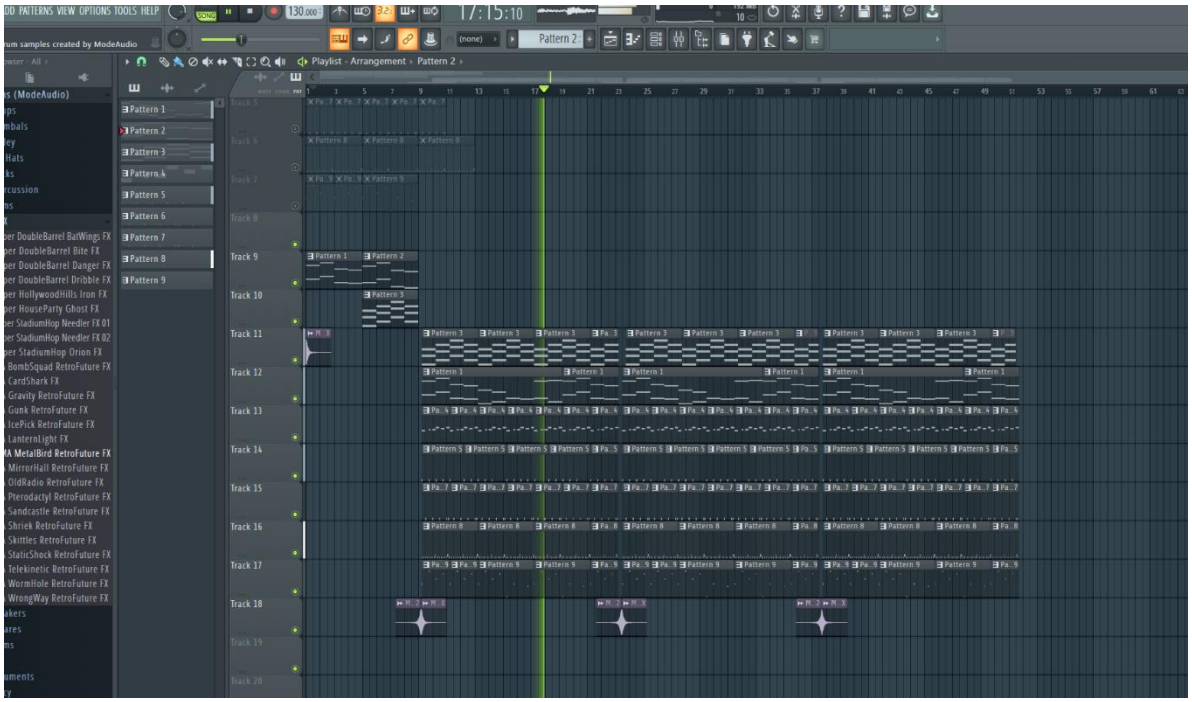

Рисунок 3.1 – Робота з музикою в FL STUDIO 20

Джерело розроблено автором

Такий вибір інструментів є оптимальним для створення багатофункціональних та конкурентоспроможних проектів у галузі відеоігор.

3.2. Тестування програмного продукту

Проведено тестування. Було перевірено правильність роботи автоматизованих функцій:

- зміна стану анімацій персонажу;

- оновлення карти при дослідженні;
- генерація локацій при новій грі;
- поява ворогів у відповідних зонах;
- поведінка ворогів: патрулювання, атака, переслідування;
- збереження статусу ворогів – вбиті вороги не відроджуються, поки гравець не взаємодіє з об'єктом збереження;
- спеціальні тригери подій для зустрічі з босом та запуску кат-сцени;
- система діалогів.

За результатами тестування критичних помилок, які заважали б проходженню гри або некоректної роботи механік не зафіксовано.

3.3. Використання програмного продукту

Початок гри починається з головного меню, можна натиснути кнопку «Почати гру» щоб перейти до першої локації гри або натиснути кнопку «Вийти» щоб закрити гру. (рис 3.2)

Рисунок 3.2 – Головне меню «Lost Element»

Джерело розроблено автором

Гравець керує персонажем за допомогою таких клавіш:

A – рух ліворуч, D – рух праворуч, Space – стрибок, Q – удар.

Також під всі дії персонажу є своя анімація головного персонажу, що робиться в Unity. (рис 3.3 та 3.4)

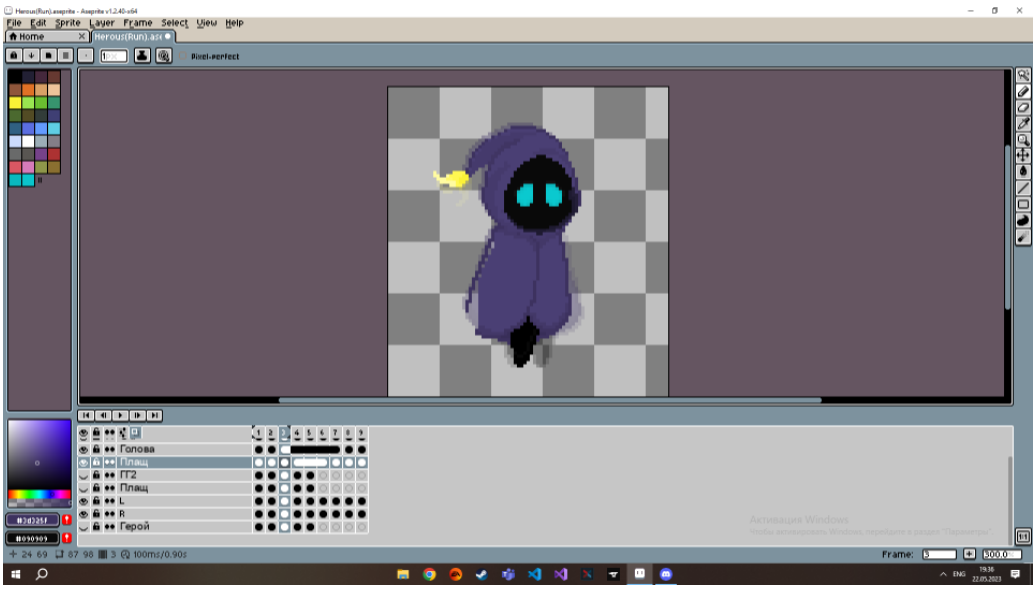

Рисунок 3.3 — Анімація бігу

Джерело зроблено автором

Рисунок 3.4 — Анімація ударів

Джерело зроблено автором

персонаж від початку має п’ять життів, які відповідають за життя персонажу, якщо герой не матиме життя він повинен бути натиснути кнопку respawn в меню яке з’являється при поразці головного героя. (рис 3.5 та 3.6)

На рисунку 3.5 показано:

- Зліва – повне життя;
- Зправа – втрачене життя.

Рисунок 3.5 – Життя повне та втрачене

Джерело розроблено автором

Рисунок 3.6 – Меню для перероджження

Джерело розроблено автором

При старті гри користувача чекає перша локація де є ворожі боти, дружні боти, об’єкти або предмети на локації а також платформи через які потрібно перестрибувати. При досліджені локації гравець буде зустрічати об’єкти або предмети з якими персонаж може взаємодіяти. Для цього потрібно підійти до об’єкта поки не з’явиться напис, після чого натиснути клавішу E. (рис. 3.7)

При натисканні кнопки E, користувач побачить на екрані діалогове вікно та зможе прочитати вміст.

Щоб перемикнути діалогове вікно або закрити його, користувачу потрібно натиснути любу клавішу.

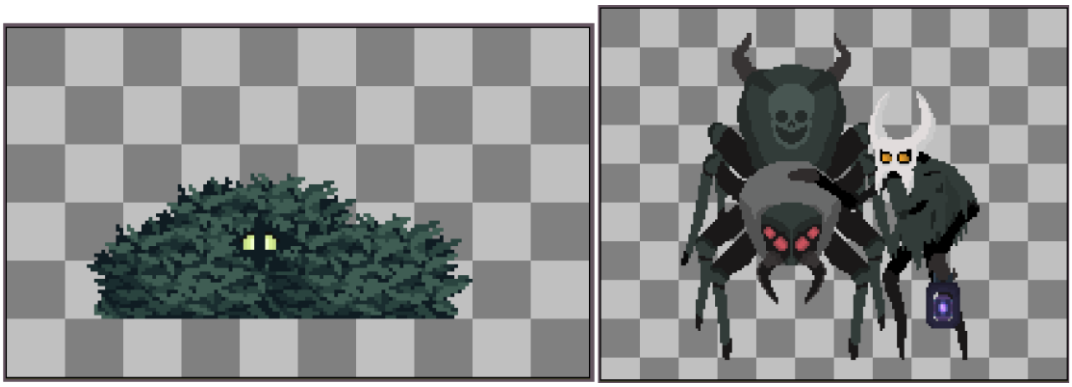

Рисунок 3.7 – Дружні боти (NPC)

Джерело розроблено автором

Локації будуть населені противниками. При входженні в поле зору ворога він почне атакувати, якщо гравець почне тікати, противник буде його переслідувати до певного моменту. (рис 3.8 та 3.9).

Також на другій локації знаходиться противник з алгоритмом A* (рис 3.10)

На локації є певні об’єкти які є системою збереження і відродження, при взаємодії з цим об’єктом гра буде автоматично зберігатися, вбиті вороги відроджуватися, а персонаж отримає точку відродження. Після смерті персонаж відродиться на кристалі з яким взаємодіям останній раз. (рис 3.11)

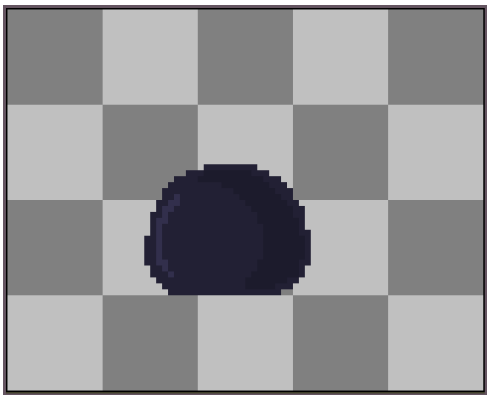

Рисунок 3.8 – Противник

Джерело зроблено автором

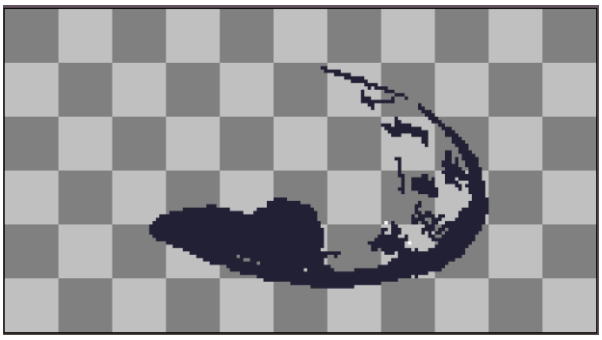

Рисунок 3.9 – Анімація атаки

Джерело зроблено автором

*Рисунок 3.10 – Шлях алгоритму A**

Джерело зроблено автором

Рисунок 3.11 – об’єкт збереження

Джерело зроблено автором

На останній локації гравець зустрінеться з фінальним босом «Божевільний послідовник». Він має дві фази. В першій атакує простими атаками. Коли персонаж наносить достатньо шкоди бос переходить у другу фазу атаки стають швидше, а також додається декілька нових прийомів. Після вбивства його гра завершується. (рис 3.12)

Рисунок 3.12 — Божевільний послідовник

Джерело розроблено автором

Висновки до розділу 3

Під час розробки гри в середовищі Unity було успішно створено основну структуру ігрових рівнів, що включає платформи, фонові елементи та об’єкти для взаємодії. Кожен рівень детально конфігурувався через інспектор Unity налаштовувалася фізика об’єктів, а також система переходу між сценами, що забезпечує безперервний ігровий процес.

Особливе значення приділялося створенню елементів інтерфейсу користувача: панелі відображення здоров’я гравця, лічильнику зібраних монет та іншим візуальним індикаторам, які допомагають орієнтуватися під час гри.

Головний персонаж керується за допомогою власного набору скриптів і здатний виконувати ближню атаку, що дає змогу ефективно знищувати ворогів. У грі реалізовано два типи супротивників — наземні та повітряні — кожен із яких має унікальну поведінку: патрулювання території, виявлення гравця та переслідування. Для ворогів створено окремі анімації, а також налаштовано параметри здоров’я, швидкість руху та сила атаки.

Завершення проходження рівня відбувається через спеціальну зону, яка надає можливість перейти до головного меню або завершити гру.

ВИСНОВКИ

Було досліджено роль і призначення комп’ютерних ігор, проведено аналіз основних ігрових жанрів. Розглянуто існуючі аналоги програмного продукту, визначено їхні сильні та слабкі сторони. Сформульовано основні правила гри та описано автоматизовані функції, що використовувалися у процесі розробки.

Визначено функціональні та нефункціональні вимоги до гри. Побудовано діаграми прецедентів (Use Case), які відображають основні можливості гри. Побудовано діаграму послідовності (Activity Diagram) для глибшого розуміння принципів зміни анімацій, роботи збереження та взаємодії об’єктів. Також було описано застосовані простори імен, компоненти, класи та основні методи в діаграмі класів (Class Diagram).

Проведено порівняльний аналіз різних ігрових рушіїв та графічного програмного забезпечення, що розглядалося на етапі планування. Зазначено причини, чому було обрано конкретні інструменти для реалізації гри, з урахуванням їхніх можливостей та переваг. Проведено тестування для виявлення критичних помилок і перевірки стабільності роботи гри.

Для повнішого уявлення про розроблений продукт описано ключові елементи геймплею, механіки, структуру гри та надано інструкцію для користувача.

У результаті створено 2D гру в жанрі платформера, що включає інтерфейс користувача, HUD персонажа, систему нанесення шкоди ворогам, кат-сцени, збереження прогресу, діалоги з NPC, поведінку ботів ШІ, генерацію ігрових рівнів та сайт вікіпедію до гри «Lost Element».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hollow Knight – URL: https://uk.wikipedia.org/wiki/Hollow_Knight

2. Celeste // Матеріал з Вікіпедії — вільної енциклопедії – URL: https://uk.wikipedia.org/wiki/Celeste_(%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D0%B3%D1%80%D0%B0)

3. Shovel Knight – URL: https://en.wikipedia.org/wiki/Shovel_Knight

4. Dead Cells – URL: https://en.wikipedia.org/wiki/Dead_Cells

5. Алгоритм A* – URL: https://en.wikipedia.org/wiki/A*_search_algorithm

6. A* Unity – URL: https://learn.unity.com/project/beginner-ai-pathfinding

7. A* 2D PathFinding // Brackeys – URL: https://www.youtube.com/watch?v=jvtFUfJ6CP8&ab_channel=Brackeys

8. Platformer – URL: https://en.wikipedia.org/wiki/Platformer

9. Shri Shivaji. EXPLORING THE POPULARITY OF 2D GAMES // International Research Journal of Modernization in Engineering Technology and Science. 07.04.April-2025. № - 8.187. - С. 1–3.

10. C# language documentation - URL: https://learn.microsoft.com/en-us/dotnet/csharp/

11. Unity Базовый курс: 2D платформер с нуля - URL: https://unity3dschool.com/unity-bazovyj-kurs-2d.html

12. Створення ігрової сцени і скрипу на Unity - URL: https://go-mother.com/creating-game-scene-and-squeak-on-unity/

13. Створення штучного інтелекту ворогів у іграх. Поради Senior Gameplay Designer компанії Remedy - URL: https://gamedev.dou.ua/blogs/enemy-ai-in-games/

14. Створення 2D-анімації в Unity - URL: https://uk.sharpcoderblog.com/blog/creating-2d-animations-in-unity

15. Unity 2D suite for game creators - URL: https://unity.com/solutions/2d

16. Adams, E. *(2013)*. *Fundamentals of Game Design*. Берклі: New Riders Publishing, 2013. С. 236.

17. Діаграма прецедентів – URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів?utm_source=chatgpt.com

18. Діаграма діяльності – URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності

19. Діаграма класів – URL: https://uk.wikipedia.org/wiki/Діаграма_класів

20. Генерація рівнів – URL: https://www.youtube.com/watch?v=QfzhYXr-U84&ab_channel=maxter

21. Моделювання поведінки неігрових персонажів у комп'ютерних іграх Н.О. СОКОЛОВА, В.В.ГНАТУШЕНКО, М.С. МІЩЕНКО, О.А. АТАМАНЧУК, Національний технічний університет «Дніпровська політехніка»

22. Чеботарьова І. Б., Черкашина Г. І. Основні тренди UI/UX дизайну 2024. Поліграфічні, мультимедійні та web-технології : матеріали Молодіжної школи-семінару IX Міжнар. наук.-техн. конф., 14-28 травня 2024 р. – Харків : ТОВ «Друкарня Мадрид», 2024. – Т. 2. – С. 40-47

23. Мічківський С. Microsoft Office (Word, Excel, Outlook …) : навч. посіб. / С. Мічківський, Д. Балдик, В. Головань; Східноукр. нац. ун-т ім. В. Даля, Аграр. ф-т. – Київ : [Вид-во Східноукр. нац. ун-т ім. В. Даля], 2023. – 128 с. – URL: https://dspace.snu.edu.ua/handle/123456789/1723

24. Vysochyn I., Michkivskyy S. Using machine learning for translation and speech generation in e-book reading applications. Держава, регіони, підприємництво: інформаційні, суспільно-правові, соціально-економічні аспекти розвитку: матеріали VI Міжнародної наукової конференції (5-6 грудня 2024 р., м. Київ). Київ: Університет "КРОК", 2024. С.62-64 – URL: https://dspace.krok.edu.ua/items/6e1488e1-9e21-4282-af10-2e3bca48d2bc

25. Алимова О. В., Мічківський С. М. Розробка антагоністичної комп'ютерної гри «Лабіринт на двох». XII Міжнародній науково-технічній конференції «Проблеми Інформатизації» (м. Київ, 13 грудня 2018 року). Київ: Державний університет телекомунікацій, 2018.

26. Алимова О.В. РАЗРАБОТКА АНТАГОНИСТИЧЕСКОЙ КОМПЬЮТЕРНОЙ ИГРЫ «ЛАБИРИНТ НА ДВОИХ» / О.В. Алимова, С.М. Мічківський // Матеріали І Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: Збірка наукових праць[Текст] / Під редакцією Г.О. Райко (м.Херсон, 30 листопада 2018 року). – Херсон: ХНТУ, 2018 - 299 с. – С. 112-114

27. Фещенко М.І. Кіберспорт та освітній процес / М. І. Фещенко, Л.Б. Ігнатова // Держава, регіони, підприємництво: інформаційні, суспільно-правові, соціально-економічні аспекти розвитку: тези доповідей V Міжнародної конференції (Київ, 7 грудня 2023 р.). - Київ: Університет "КРОК", 2023. URL: https://conf.krok.edu.ua/SRE/SRE-2023/paper/view/1989

28. Системи та методи прийняття рішень: методичні вказівки / С. М. Мічківський, О. В. Прігунов, П. В. Римар. Вінниця, ДонНУ імені Василя Стуса, 2019, 76 с.

29. Троцько В.В. Методи штучного інтелекту: навчально-методичний і практичний посібник / В.В. Троцько. – Київ: Університет економіки та права «КРОК», 2020. – 86 с. URL: https://library.krok.edu.ua/media/library/category/navchalni-posibniki/trotsko_0001.pdf

ДОДАТКИ А
ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Seeker.cs | Hero.cs | RoomManager.cs | SpawnBlockscr.cs | RandVariants.cs | Boss.cs

Assembly-CSharp — Hero

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

Unity Script (2 asset references) | 15 references
public class Hero : Entity
{
    [SerializeField] private float speed = 3f;
    [SerializeField] private float health;
    [SerializeField] private float jumpForce = 10f;
    private bool isGrounded = false;

    public bool isAttacking = false;
    public bool isRecharged = true;

    public Transform attackPos;
    public float attackRange;
    public LayerMask enemy;
    public VectorValue pos;




    [SerializeField] private Image[] hearts;

    [SerializeField] private Sprite aliveHeart;
    [SerializeField] private Sprite deadHeart;
    [SerializeField] private GameObject losePanel;



    private Rigidbody2D rb;
    private Animator anim;
    private SpriteRenderer Square;

    [SerializeField] private AudioSource jumpSound;
    [SerializeField] private AudioSource miss;
    [SerializeField] private AudioSource attack;
    [SerializeField] private AudioSource damageSound;
    15 references
    public static Hero Instance { get; set; }

    4 references
    private States State
    {
        get { return (States)anim.GetInteger("state"); }
        set { anim.SetInteger("state", (int)value); }
    }
    Unity Message | 0 references
    void Awake()
```

68 % | No issues found

Error List | Entire Solution | 0 Errors | 0 Warnings | 0 of 9 Messages | Build + IntelliSense

Рисунок А.1 – Скрипт Hero

Джерело розроблено автором

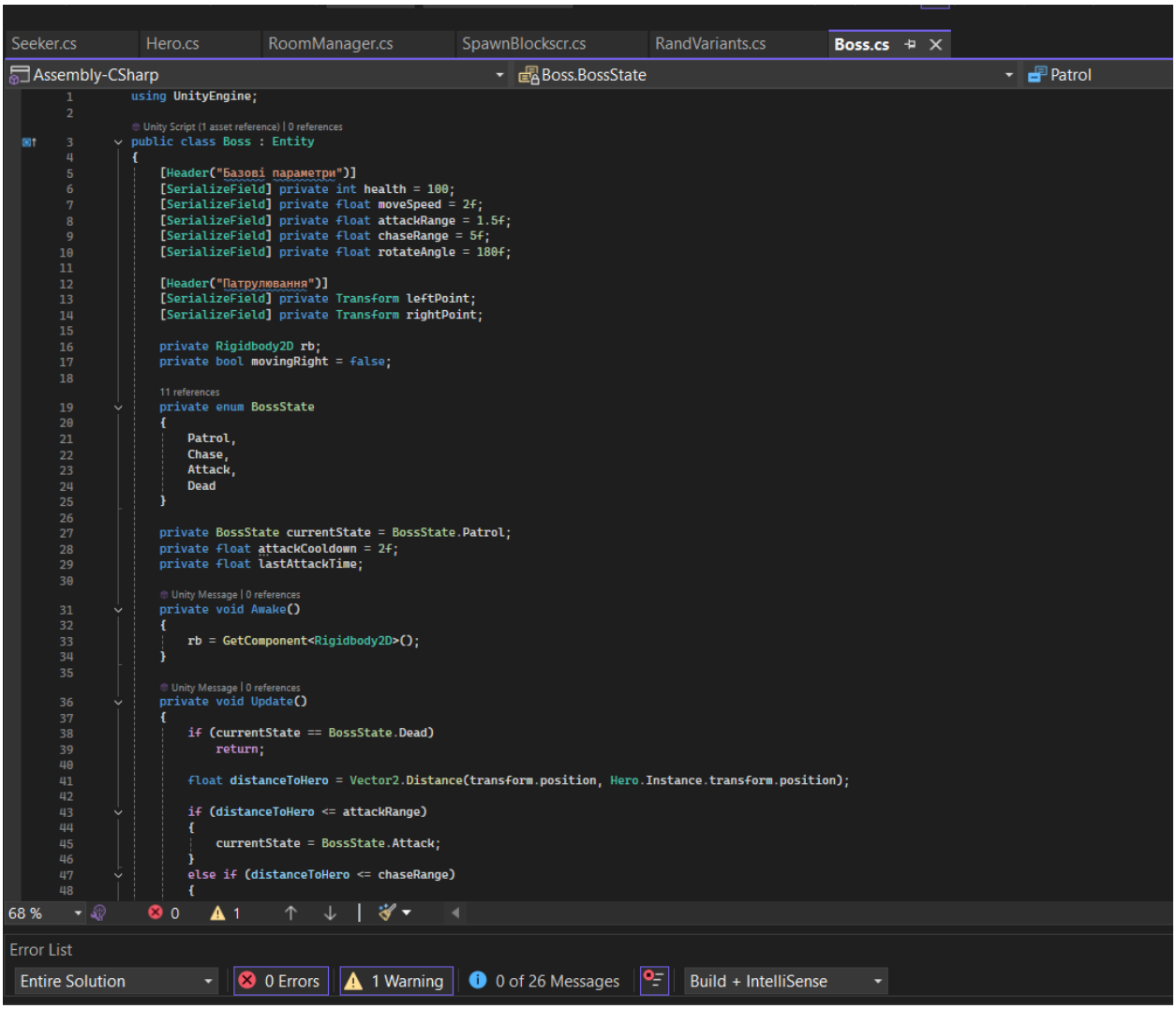

Рисунок А.2 — Скрипт Boss (Enemy)

Джерело розроблено автором