Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Жукової Анастасії Владиславівни*
(ПІБ)

академічної групи *122-20-4*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка ігрового застосунку "Paws & Magic" у середовищі Unreal Engine*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В*			
розділів:				
спеціальний	*доц. Кабак Л.В*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Каштан В.Ю.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-4* *Жукової А.В.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка ігрового застосунку "Paws & Magic" у середовищі Unreal Engine*

затверджена наказом ректора НТУ «ДП» від *23.05.2024 р.* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2024 р.***

Завдання видав _____ *доц. Кабак Л.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Жукова А.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *10,06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 64 с., 16 рис., 3 дод., 3 табл., 20 джерел.

Об'єкт розробки: Ігровий застосунок "Paws & Magic" у середовищі Unreal Engine.

Мета дипломного проекту: створити ігровий застосунок для ПК, що представляє собою поєднання класичного ігрового досвіду з сучасними технологіями розробки.

У вступі до кваліфікаційної роботи проведено аналіз сучасного стану ігрової індустрії та визначено актуальність теми кваліфікаційної роботи. Чітко сформульовано мету роботи та окреслено галузь застосування даного продукту, а також визначено конкретні завдання, які необхідно вирішити для досягнення поставленої мети.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до фінального продукту, технологій та програмних засобів.

У другому розділі описано платформу для розробки, виконано проектування і розробка програми, описана робота програми і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано очікуваний час на його створення.

Практичне значення полягає у створенні інформаційної системи, що базується на сучасному ігровому рушії Unreal Engine 5. Результатом роботи є функціональний ігровий застосунок, який може бути використаний для розваги та навчання.

Актуальність даного програмного забезпечення зумовлена зростаючою популярністю жанру та необхідністю розвивати українську ігрову індустрію. Створення якісних ігрових продуктів сприяє її подальшому зростанню та зміцненню позицій на світовому ринку.

Список ключових слів: 2D, UNREALENGINE, BLUEPRINTS, C++, GAMING, ANIMATION, USERINTERFACE.

ABSTRACT

Explanatory note: 64 pages, 16 figures, 3 appendices, 3 table, 20 sources.

Object of development: Game application "Paws & Magic" in the Unreal Engine environment.

The purpose of the graduation project: to create a PC game application that combines classic gaming experience with modern development technologies.

The introduction to the qualification work analyzes the current state of the gaming industry and determines the relevance of the topic of the qualification work. The purpose of the work is clearly formulated and the field of application of this product is outlined, as well as specific tasks that need to be solved to achieve the goal are determined.

The first chapter analyzes the subject area, determines the relevance of the task and the purpose of the development, formulates the task statement, indicates the requirements for the final product, technologies and software.

The second chapter describes the development platform, the design and development of the program, describes the work of the program and the structure of its functioning, as well as the call and loading of the program, determines the input and output data, characterizes the composition of the parameters of technical means.

In the economic section, the labor intensity of the developed information system is determined, the cost of creating the program is calculated, and the expected time for its creation is calculated.

The practical significance lies in the creation of an information system based on the modern Unreal Engine 5 game engine. The result of the work is a functional game application that can be used for entertainment and education.

The relevance of this software is due to the growing popularity of the genre and the need to develop the Ukrainian game industry. The creation of high-quality gaming products contributes to its further growth and strengthening of its position in the global market.

List of keywords: 2D, UNREALENGINE, BLUEPRINTS, C++, GAMING, ANIMATION, USERINTERFACE.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

ПК - Персональний комп’ютер

2D – Двовимірна графіка

BP – Blueprint

IDE –Integrated Development Environment

UI – User Interface

UMG – Unreal Motion Graphics

UE5 – Unreal Engine 5

UE4 – Unreal Engine 4

ЗМІСТ

ВСТУП

У сучасному світі відеоігри продовжують займати особливе місце у житті та культурі молоді, пропонуючи унікальний ігровий досвід, що поєднує в собі елементи пригод, екшену та головоломок. А саме головне, що поринути у цю культурну ланку можливо кожному, маючи лише смартфон, комп’ютер або навіть смарт-годинник. Ця доступність та захопливість робить їх привабливими для широкої аудиторії, незалежно від віку та ігрових уподобань. Завдяки розвитку технологій, сучасні ігрові рушії надають потужні інструменти для створення захопливих та візуально привабливих ігор.

З розвитком технологій розробки ігор з'явилися нові можливості для створення більш захопливих та візуально привабливих ігор. Сучасні ігрові рушії надають потужні інструменти для реалізації різноманітних ігрових механік, створення високоякісної графіки та анімацій, а також оптимізації продуктивності гри. Кожен із них має свої переваги та недоліки, які даються та досліджуються перед розробкою ігрового застосунку.

Метою даної кваліфікаційної роботи є розробка ігрового застосунку під назвою "Paws and Magic", що поєднує в собі класичні елементи жанру 2D-платформер з сучасними технологіями розробки. Гра запропонує гравцям захопливу подорож фентезійним ігровим світом, де їм доведеться долати перешкоди, збирати корисні предмети та боротися з ворогами. Ігрові механіки поєднуватимуть в собі найкращі практики із класичних ігор даного жанру. Гра повинна мати сучасний та актуальний ігровий рушій, який надаватиме широкі можливості для розробки та тестування. Також гра повинна мати високу якість графіки, плавну анімацію та зручний інтерфейс користувача.

У рамках цієї кваліфікаційної бакалаврської роботи буде детально описано весь процес розробки гри "Paws and Magic". Це включатиме вибір та обґрунтування використання Unreal Engine 5 як основного інструменту розробки, детальне проектування ігрових рівнів з урахуванням особливостей жанру, створення унікальних персонажів з використанням передових технік анімації, а

також реалізацію захопливої ігрової логіки, що включатиме різноманітні механіки взаємодії з оточенням, систему збору предметів та прогресу гравця. Особлива увага буде приділена створенню інтуїтивно зрозумілого та зручного інтерфейсу користувача, який забезпечить комфортну взаємодію гравця з грою.

Беручи це все до уваги, було сформовано тему роботи: «Розробка ігрового застосунку "Paws & Magic" у середовищі Unreal Engine». Ця робота представляє собою детальний опис процесу розробки гри, включаючи всі етапи від проектування до тестування, що дозволить створити якісний та цікавий продукт для кінцевих користувачів.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

У світлі процвітання світової ігрової індустрії, важливо дослідити унікальний внесок та досягнення окремих країн. Серед цих країн, що розвиваються, Україна виділяється значним зростанням своєї ігрової індустрії за останні роки.

Зародження ігрової індустрії відбулося наприкінці 1950-х — на початку 1960-х років. Перші новатори, такі як Вільям Хігінботем з його грою «Теніс для двох», а згодом Ральф Баер з «Коричневою коробкою», заклали основу для того, що перетворилося на глобальне явище.

Спочатку ігри розроблялися переважно невеликими командами або навіть окремими ентузіастами, часто як експерименти або додаткові проекти в рамках великих технологічних компаній. Ці ранні ПК ігри були простими за сучасними мірками, обмежені обчислювальною потужністю та пам'яттю доступного на той час обладнання. Проте, вони захопили уяву перших гравців та стали основою для майбутнього розвитку індустрії.

Монетизація в зароджуваній ігровій індустрії була переважно пов'язана з продажем фізичних носіїв з іграми. Розробники та видавці покладалися на продаж картриджів, дискет чи касет споживачам, не маючи жодного уявлення про цифрову дистрибуцію чи мікротранзакції. Така модель створювала значні бар'єри для входу на ринок дрібних розробників та обмежувала потенційну аудиторію [1].

Однак на початку 1980-х років індустрія зіткнулася з кризою, спричиненою перенасиченням ринку, низькою якістю ігор та подальшим крахом. Це призвело до широкого скептицизму щодо життєздатності відеоігор як сталого бізнесу.

Ситуація змінилася з появою домашніх ігрових консолей. Завдяки таким культовим іграм, як Super Mario Bros, Pac-Man та їжачок Сонік, ігрова індустрія пережила відродження популярності.

Перехід від фізичних носіїв до цифрової дистрибуції, у поєднанні з технологічним прогресом і зростанням інтернет-зв'язку, ще більше революціонізував індустрію. Це відкрило шлях для нових розробників та бізнес-моделей, таких як послуги передплати та внутрішньоігрові покупки [1].

Ігрова індустрія сьогодні є однією з найбільш динамічних та інноваційних галузей у світі. Вона об'єднує мільйони людей, включаючи розробників, гравців, дизайнерів, маркетологів і багатьох інших професіоналів. За даними інтернет-видань, у У 2023 році дохід ігрової індустрії сягає $390 млрд, а кількість гравців перевищила 3 мільярди осіб [2]. Це свідчить про те, що ігри вже давно перестали бути просто розвагою для дітей і підлітків. Вони стали важливою частиною культури. Сучасні ігри можуть похвалитися високоякісною графікою, захоплюючими сюжетами та складними механіками, що дозволяє гравцям повністю зануритися в віртуальні світи.

Індустрія ігор це не тільки продаж ігор та пристроїв, а й величезний ринок віртуальних задоволень, на якому гравці з різних країн витрачають мільярди доларів. Наприклад, гравці з Китаю та США витратили майже по 30 мільярдів доларів на ігрові продукти. Крім того, індустрія заробляє на створенні контенту та організації змагань. Наприклад, у лютому 2018 року шанувальники Fortnite витратили понад 118 мільярдів годин на перегляд відео, пов'язаного з грою, лише на платформі Twitch. Професійні гравці також заробляють мільйони доларів, беручи участь у турнірах, а українські кіберспортсмени Роман Фомінок і Володимир Міненко отримали понад мільйон доларів призових кожен. Українські компанії вже понад 10 років створюють віртуальні світи та займають значне місце в ігровій індустрії, розробляючи ігри, які стають відомими та популярними [3].

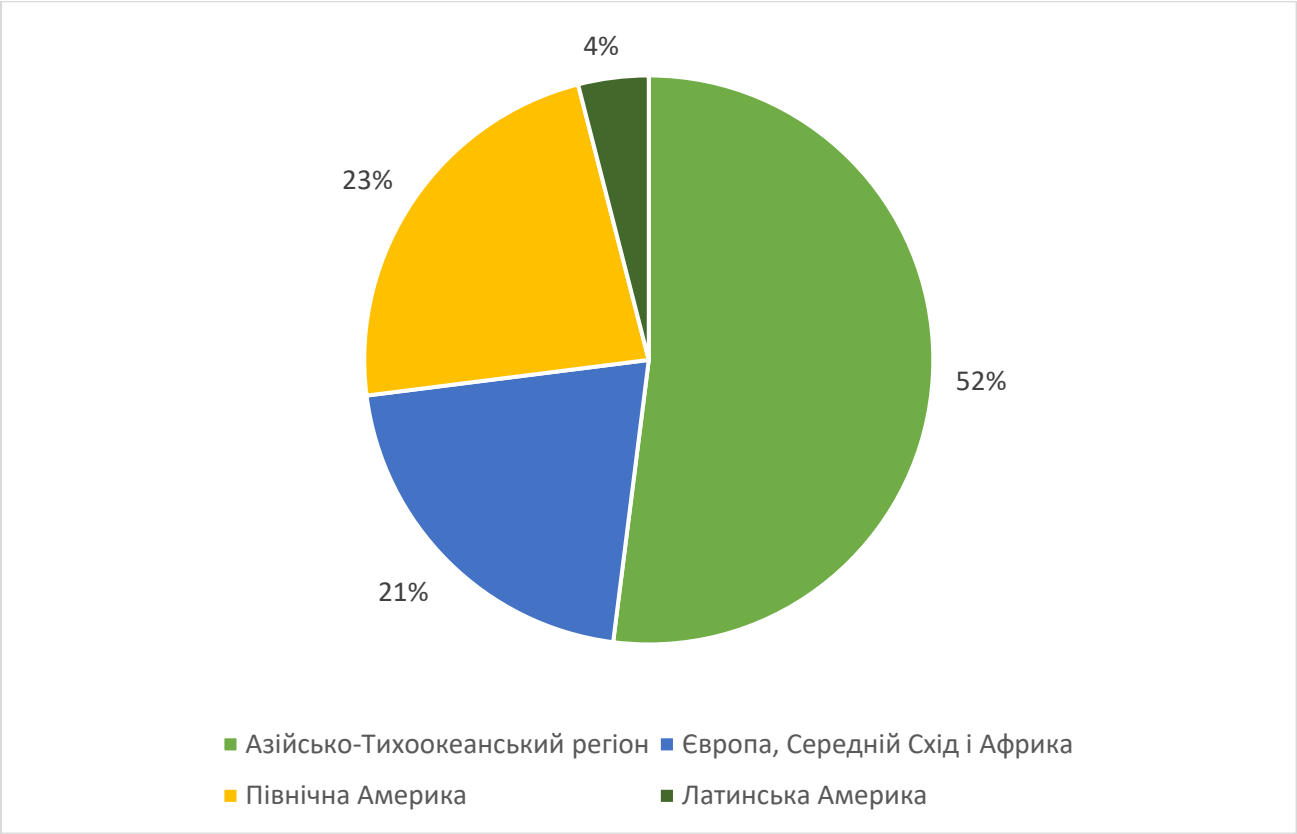

Рис. 1.1 Діаграма відносного розподілення ринку ігрових продуктів по регіонам світу [3]

Ринок відеоігор є одним із найбільших сегментів розважальної індустрії, перевершуючи за прибутками кіно та музичну індустрію [4]. Кожного року випускаються тисячі нових ігор, які продаються мільйонами копій по всьому світу. Гравці витрачають значні суми грошей на придбання ігор, додаткового контенту та обладнання для геймінгу. Змагання з кіберспорту збирають величезні аудиторії, а професійні гравці стають справжніми зірками, заробляючи мільйони доларів на призових та спонсорських контрактах.

У технологічному плані, ігрова індустрія є одним з рушіїв прогресу. Розробка ігор стимулює розвиток новітніх технологій, таких як віртуальна і доповнена реальність, штучний інтелект та графічні процесори. Наприклад, платформи на зразок Unreal Engine та Unity дозволяють створювати реалістичні тривимірні світи з високим рівнем деталізації, які можуть конкурувати з реальними зображеннями. Ці технології знаходять застосування не лише в іграх,

але і в інших галузях, таких як медицина, архітектура, освіта, кіноіндустрія та військова справа.

Ігрова індустрія також значно впливає на соціальні аспекти життя. Ігри стають платформою для соціальної взаємодії, де люди можуть знайомитися, спілкуватися та спільно проходити випробування. Вони допомагають розвивати різні навички, такі як стратегічне мислення, координація рухів та робота в команді. Крім того, ігри все частіше використовуються для освітніх цілей, надаючи інтерактивний спосіб вивчення нових знань та навичок.

Український ринок відеоігор, незважаючи на складні умови, пов'язані з повномасштабним вторгненням, продовжує демонструвати потенціал до розвитку. У 2022 році обсяг ринку склав $287 млн [4]. Незважаючи на все, українські ігрові компанії продовжують працювати та випускати нові продукти, а деякі з них навіть збільшили свої команди та обсяги виробництва.

Три чверті ігрових компаній, що працюють в Україні, мають і штаб-квартиру в Україні, отже маємо можливість створювати ідеї, розробляти і керувати процесом розробки і продажів ігор. Крім цього, наші компанії надають послуги закордонним геймдев-студіям, займаються аутсорсингом, аутстафінгом. Тобто компанії більше покладаються на якісну роботу наших програмістів, аніж на створення брендів і захоплюючих ідей [3]. Нижче наведена діаграма співвідношення часток компаній в ігровій індустрії України, де розглянуто 6 основних сфер діяльності:

– Паблішинг - Процес підготовки, випуску та поширення відеоігор або іншого програмного забезпечення. Паблішер може займатися маркетингом, продажем, локалізацією та підтримкою продукту.

– Консалтинг - Надання експертних порад та допомоги компаніям у вирішенні конкретних проблем або досягненні певних цілей. У сфері розробки програмного забезпечення консалтинг може включати аналіз вимог, проектування архітектури, тестування та оптимізацію.

– Сервіс для інших компаній - Послуги, що надаються однією компанією іншій компанії, такі як розробка програмного забезпечення на замовлення, хмарні обчислення, маркетингові дослідження тощо.

–Інді-розробка - Процес створення відеоігор або іншого програмного забезпечення невеликими незалежними командами або окремими розробниками, без фінансової підтримки великих видавців.

–Аутсорсинг - Передача певних бізнес-процесів або завдань зовнішньому підряднику або компанії. Це може включати розробку програмного забезпечення, підтримку клієнтів, бухгалтерію тощо.

–Аутстафінг - Форма аутсорсингу, при якій зовнішні спеціалісти тимчасово залучаються до роботи над проектом компанії-замовника, але залишаються працівниками компанії-провайдера.

– Власний продукт - Програмне забезпечення або інший цифровий продукт, розроблений та повністю контрольований однією компанією, призначений для продажу або використання в рамках власної діяльності.

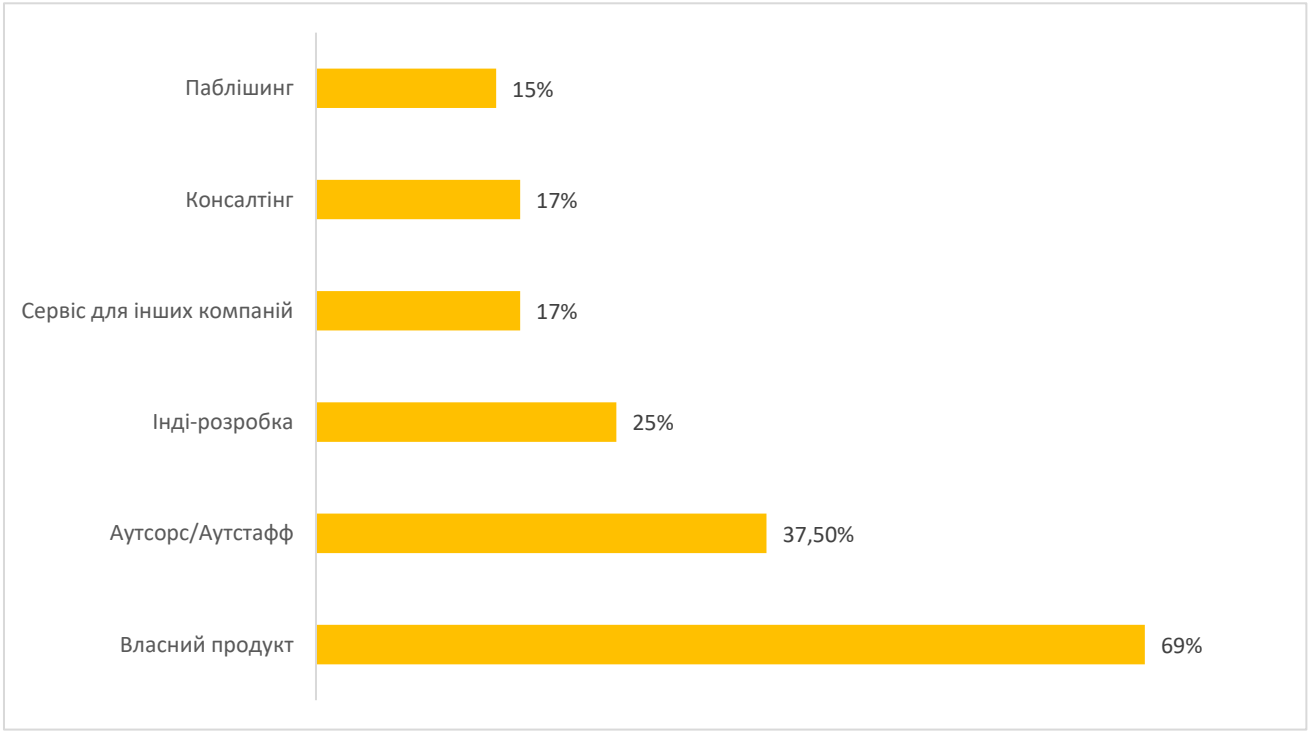

Рис. 1.2 Частка українських геймдев-компаній за профілем їхньої діяльності [3]

Українські геймдев-компанії набагато більше, ніж в середньому по світу, орієнтовані на створення ігор для мобільних пристроїв і PC, а ось до приставок відношення більш стримане. Наша країна має найбільшу в світі кількість спеціалістів, які працюють з ігровим рушієм Unity 3D і не дивно, що більшість компанії обирають його для розробки ігор, інші технології менш популярні [3].

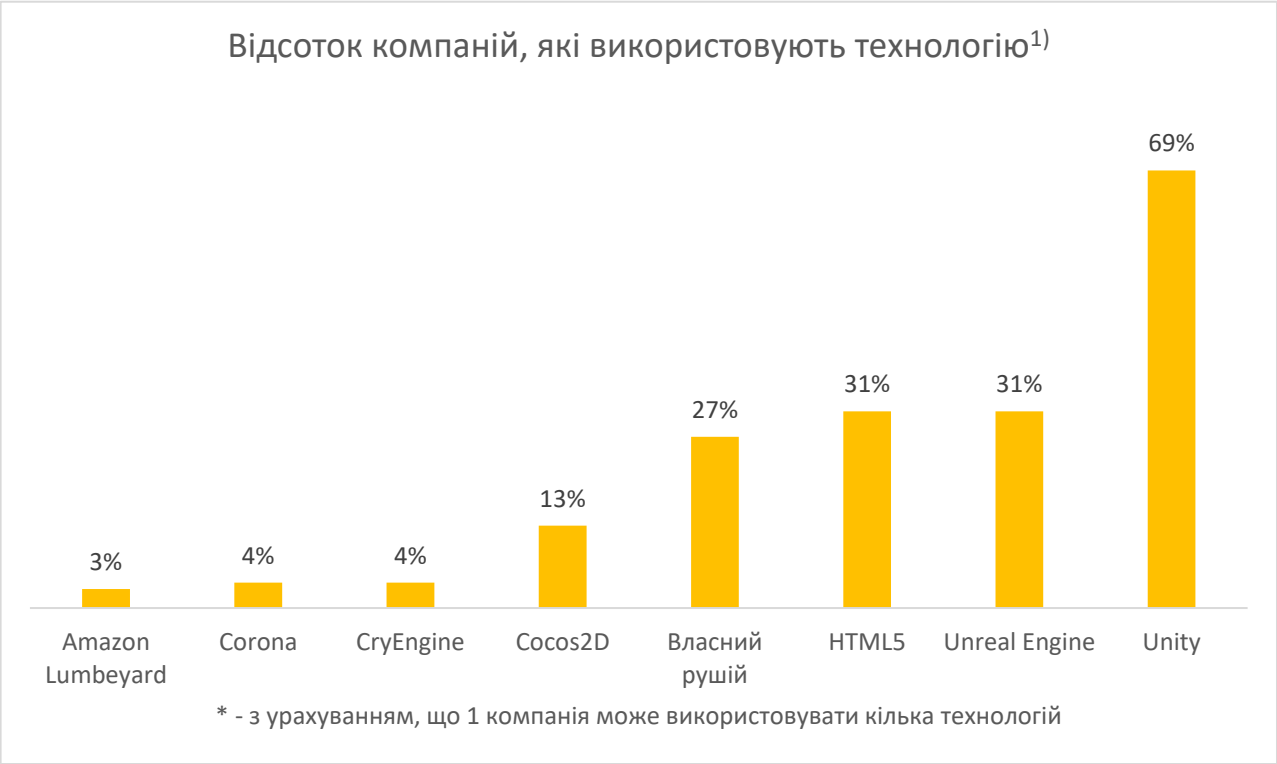

Рис. 1.3 Технології, які найчастіше використовують українські компанії [3]

Примітка.

Відсотки підраховані з урахуванням, що 1 компанія може використовувати кілька технологій.

Повномасштабне вторгнення створило значні виклики для української ігрової індустрії, включаючи втрату персоналу, проблеми з логістикою та фінансуванням. Проте, українські розробники проявили стійкість та адаптивність, знаходячи нові способи роботи та підтримки бізнесу. Багато компаній перемістили свої офіси в безпечніші регіони України або за кордон, а деякі перейшли на віддалений режим роботи.

Війна також спонукала українських розробників до створення ігор, які відображають реалії війни та підтримують українську культуру та ідентичність. Наприклад, гра "Zero Losses" розповідає історію українських військових, які захищають свою країну від російської агресії [5].

Незважаючи на труднощі, українська ігрова індустрія має значний потенціал для зростання. Українські розробники відомі своєю креативністю, технічними навичками та відданістю своїй справі. З підтримкою міжнародної спільноти, український ринок відеоігор може стає важливим гравцем на світовій арені.

Отже, ігрова індустрія сьогодні є комплексним і різноманітним явищем, яке значно впливає на економіку, технології та суспільство. Вона не тільки розважає мільйони людей по всьому світу, але й стимулює розвиток нових технологій та соціальних відносин. Розуміння цього дозволяє оцінити масштаб і значення ігрової індустрії в сучасному світі, а також перспективи її подальшого розвитку в Україні.

1.2. Призначення розробки та галузь застосування

Призначення даного застосунку полягає у наданні користувачам унікального ігрового досвіду, що відзначається високою якістю графіки, захоплюючим сеттінгом і зручними ігровими механіками. Основна мета полягає у залученні широкої аудиторії, яка може включати як звичайних гравців, так і ентузіастів, що шукають нові виклики та враження у віртуальному світі.

Ігровий застосунок для ПК буде використовуватися для забезпечення розважального контенту, надаючи користувачам можливість зануритися у віртуальний світ, де вони зможуть взаємодіяти із середовищем та відточувати свої навички у грі. Ігри можуть мати значний освітній потенціал, сприяючи розвитку когнітивних здібностей, покращенню реакції, стратегічного мислення та вирішення проблем.

Галузь застосування розроблюваного ігрового застосунку охоплює як домашнє використання, так і напівпрофесійний кіберспорт. В домашньому

середовищі гра служить джерелом розваги та релаксації, допомагаючи користувачам відволіктися від повсякденних турбот та зануритися у захоплюючі пригоди. У сфері кіберспорту ігровий застосунок може використовуватися для проведення змагань, тренувань та розвитку гравців. Висока якість гри та її конкурентні механіки сприятимуть популяризації серед професійних геймерів та кіберспортивних команд.

Крім того, ігровий застосунок може стати важливим інструментом для соціальної взаємодії, створюючи можливості для співпраці та комунікації між гравцями. Соціальні функції, такі як змігання за результатами гри між собою. Це дозволяє не лише покращити ігровий досвід, але й сприяти формуванню соціальних зв'язків та розвитку спільних інтересів серед користувачів.

1.3. Підстава для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проєкт). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему ”Розробка ігрового застосунку "Paws & Magic" у середовищі Unreal Engine” є наказ від Національного технічного університету «Дніпровська політехніка» від 23.05.2024.

1.4. Постановка завдання

Метою розробки мого ігрового застосунку для пк є створити ігровий застосунок для ПК, що представляє собою поєднання класичного ігрового досвіду з сучасними технологіями розробки, яке має в собі розважальну та культурну функції.

Розробка ігрового застосунку "Paws & Magic" включає в себе комплекс завдань, що охоплюють різні аспекти створення гри.

По-перше, це дизайн та розробка ігрових рівнів, що передбачає створення унікальних рівнів з різноманітними перешкодами, розробку механік взаємодії персонажа з оточенням та впровадження системи збереження прогресу гравця.

По-друге, це створення та анімація персонажів, включаючи розробку головного героя з базовим набором рухів, створення ворогів з певною поведінкою та реалізацію анімації головного персонажа.

По-третє, важливим аспектом є розробка ігрової логіки, що включає створення системи збору предметів, які впливають на геймплей, а також впровадження системи рахунку та збереження прогресу гравця.

Нарешті, створення інтерфейсу користувача передбачає розробку головного меню, екранів паузи, перемоги та поразки, відображення інформації про рахунок гравця та забезпечення зручної навігації та управління грою.

Для успішної реалізації проєкту необхідно виконати такі ключові етапи:

– Спроектувати детальну концепцію гри, яка включатиме опис ігрових механік, візуального стилю та інших важливих аспектів.

– Реалізувати ігрову логіку, використовуючи такі інструменти, як C++, Blueprints та інші можливості Unreal Engine 5.

– Провести ретельне тестування гри для забезпечення її коректної та стабільної роботи.

Завдяки комплексному підходу до розробки та використанню сучасних технологій, ігровий застосунок "Paws & Magic" зможе запропонувати гравцям захопливий та інтерактивний досвід, що відповідає сучасним стандартам якості та вимогам цільової аудиторії.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Запропонована система має наступні функціональні характеристики.

– Гравець повинен мати можливість керувати персонажем за допомогою клавіатури, використовуючи кнопки для пересування та одну клавішу для стрибка (або двійного стрибка).

– Кожен рівень повинен мати чітко визначену мету: зібрати всі зірки та дістатися до виходу.

– Перешкоди повинні включати статичні елементи, які ускладнюють пересування по рівню.

– Гравець повинен мати можливість знищити ворога, стрибнувши на нього зверху.

– Зібрані зірки повинні давати доступ до наступних рівнів.

– Гра повинна мати простий та інтуїтивно зрозумілий інтерфейс користувача.

– Інтерфейс повинен включати меню паузи, де гравець може повернутися назад в гру.

– Розробка гри повинна бути здійснена в середовищі Unreal Engine 5 або новішої версії.

– Під час розробки в середовищі, повинен бути застований плагін Paper2D для роботи із спрайтами та рівнями.

1.5.2. Вимоги до інформаційної безпеки

Вимоги до інформаційної безпеки у "Paws and Magic" є мінімальними, оскільки гра не збирає, не обробляє та не зберігає персональні дані користувачів. Проте, для забезпечення безпеки та стабільності роботи застосунку необхідно дотримуватися наступних вимог:

– Захист від несанкціонованого доступу: Код гри повинен бути захищений від несанкціонованого доступу та модифікації з метою запобігання втручанню в ігровий процес та збереження цілісності даних.

– Захист від шкідливого програмного забезпечення: Необхідно регулярно оновлювати програмне забезпечення, включаючи рушій Unreal Engine 5, для виправлення вразливостей та захисту від шкідливого програмного забезпечення.

– Контроль якості коду: Регулярне тестування та перевірка коду на наявність помилок та вразливостей дозволить забезпечити стабільну роботу гри та запобігти можливим проблемам безпеки.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для розробки ігрових додатків на базі Unreal Engine потрібна потужна робоча станція з сучасним багатоядерним процесором, високопродуктивною відеокартою та великим об’ємом оперативної пам’яті. Це забезпечить швидку компіляцію коду, реалістичний рендеринг і ефективне тестування гри в різних режимах. Важливим аспектом є використання швидких твердотільних накопичувачів (SSD) для скорочення часу завантаження проекту та підвищення загальної продуктивності системи.

1.5.4. Вимоги до інформаційної та програмної сумісності

Сумісність для користувача передбачає, щоб програма коректно функціонувала на різних конфігураціях комп'ютерів, з урахуванням різних операційних систем, апаратного забезпечення та параметрів відображення. З урахуванням умов ринкового середовища, необхідною вимогою є сумісність з операційними системами Microsoft Windows 10 або 11.

Програмна сумісність проєкту для розробки, у свою чергу, вимагає, щоб ігровий застосунок був розроблений і налаштований з використанням таких технологій та бібліотек, які є сумісними з Unreal Engine 5. Це включає узгоджені версії підтримуваної мови програмування (C++ або Blueprint), а також відповідність сторонніх бібліотек та інтеграцій до рекомендованих стандартів інтерфейсів програмного забезпечення.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Ігровий продукт являтиме собою 2D-платформер – це жанр відеоігор, де ігровий процес відбувається на двовимірній площині (2D) і головним елементом є переміщення персонажа по платформах різної висоти та форми [9]. Вибір даного жанру запропонує гравцям захопливий та інтерактивний досвід, сповнений динамічних випробувань та яскравих вражень. Гравці зможуть досліджувати різноманітні ігрові світи, долаючи перешкоди, збираючи цінні предмети та взаємодіючи з унікальними елементами оточення.

Основною метою гри буде досягнення кінця кожного рівня, де на гравців чекатимуть захопливі випробування. Для успішного проходження рівнів гравцям необхідно буде проявити спритність, логічне мислення та вміння використовувати різноманітні механіки, такі як стрибки, подвійні стрибки, атаки та збір предметів. Успішне завершення кожного рівня відкриватиме доступ до нових, ще більш захопливих та складних випробувань, що стимулюватиме гравця до подальшого проходження гри та дослідження її світу.

2.2. Опис застосованих математичних методів

Так як особливості предметної області розв'язуваної задачі не передбачають застосування складних математичних методів, при розробці інформаційної системи математичні методи не використовувалися. Однак, є деякі основні математичні та логічні операції, які використовувалися. До прикладу: інкрементація, перевірка умови, прості арифметичні операції тощо.

2.3 Опис використаних технологій та мов програмування

У процесі розробки моєї дипломної гри, вибір ігрового рушія став одним з ключових рішень. Як зазначалося в першому розділі кваліфікаційної роботи

найпопулярнішими варіанти були Unreal Engine та Unity. Вони обидва пропонували потужні інструменти та можливості, але також мали свої особливості та обмеження. Кожен з них мав свої переваги та недоліки, що ускладнювало вибір. Детальний аналіз цих рушіїв став необхідним кроком для визначення оптимального варіанту, який би найкраще відповідав потребам мого проекту та моїм навичкам розробника. Результати аналізу занесені до таблиці 2.1 нижче.

Таблиця 2.1

Порівняльна характеристика рушіїв для розробки ігрових застосунків [6]

Критерій	Unity	Unreal Engine
Основні мови програмування	C#	C++, Blueprints
Платформи	PC, консолі, мобільні, VR/AR, веб	PC, консолі, мобільні, VR/AR
Ліцензування та вартість	Безкоштовна, Pro-версія $150/міс	Безкоштовна до $1 млн доходу, 5% роялті
Графічні можливості	Добре для 2D, 3D менш потужне	Висока деталізація 3D, технології Lumen та Nanite
Простота використання	Нижчий поріг входу	Вищий поріг входу
Інструменти для дизайну	Зручні для мобільних і 2D	Blueprints для візуального скриптування
Документація та підтримка	Велика інтернет-спільнота, що означає багато інформаційних ресурсів	

Продовж. табл. 2.1

Розширюваність	Велика кількість плагінів	Відкритий вихідний код, модифікація коду
Сфери використання	Широко використовується для мобільних ігор	Популярний для AAA-проектів та VR/AR
Оптимізація	Легка оптимізація для мобільних пристроїв	Багаті інструменти, але складні в освоєнні

В результаті порівняння було прийняте рішення обрати рушій Unreal Engine у зв’язку з ширшим функціоналом та перспективою розвитку проєкту на майбутнє. Серед мінусів було зазначено менші графічні можливості для 2D ігор, але рішення цієї проблеми вже знайдене. Оскільки для цього рушія створюються тисячі плагінів для різноманітних потреб, із недавніх пір було винайдено й такий, який допомагає в розробці 2D ігор.

Під час розробки даної кваліфікаційної бакалаврської роботи застосовувалась мова програмування C++. Будучи потужною та гнучкою мовою програмування, вона залишається однією з найпопулярніших мов у розробці ігор.

Більшість популярних ігрових рушіїв, таких як Unreal Engine та більша частина Unity, побудовані на C++. Це дозволяє забезпечити високу продуктивність та ефективне використання ресурсів, що критично важливо для сучасних ігор з реалістичною графікою та складними симуляціями.

C++ широко використовується для розробки графічних підсистем, шейдерів та візуальних ефектів. Бібліотеки, такі як DirectX та OpenGL, надають низькорівневий доступ до графічного обладнання, дозволяючи розробникам створювати вражаючу візуальну складову ігор [8].

Здатність до написання кодів низького рівня робить її незамінною для створення системного коду, базових ігрових механізмів та високопродуктивних

алгоритмів, що забезпечує оптимальне використання апаратних ресурсів та плавний ігровий процес.

Blueprints, у свою чергу, дозволить створювати складну ігрову логіку та поведінку об'єктів без необхідності написання великої кількості коду, що прискорить процес розробки та зробить його більш гнучким.

Додатковим інструментом для розробки мого ігрового продукту були Blueprints. Blueprints – це система візуального ООП-програмування в середовищі Unreal Engine. Вона більш доступна, швидка та вважається легшою для опанування, однак підходить не для всього. Розробка у Blueprints у 8+ разів швидша, ніж розробка такої самої функціональності на C++ [7]. Навіть передові геймдев-компанії використовують їх при розробці своїх продуктів в комбінації із C++.

Найчастіше Blueprints застосовуються для скриптування анімацій, матеріалів, звуків, інтерфейсів тощо. Нижче наведена таблиця переваг та недоліків Blueprints в Unreal Engine, за допомогою якої обирався спосів для розробки всіх необхідних елементів гри.

Таблиця 2.2

Переваги та недоліки Blueprints в Unreal Engine

Blueprints	
Переваги	Недоліки
Розробка і компіляція в рази швидша, ніж на C++	Вимагає більше пам’яті накопичувача
Просте використання готових функцій, які можна легко під’єднати у правильному порядку	Повільна робота циклів порівняно із C++
Автоматичні перевірки	Не всі методи із C++ доступні
Динамічна робота	Дещо гірша робота з математикою

Продовж. табл. 2.2

Зручність роботи з посиланнями на моделі чи матеріали	Проблематично оптимізувати

2.4 Опис структури програми та алгоритмів її функціонування

Узагальнена схема роботи мого ігрового застосунку виглядає як на схемі (рис. 2.1):

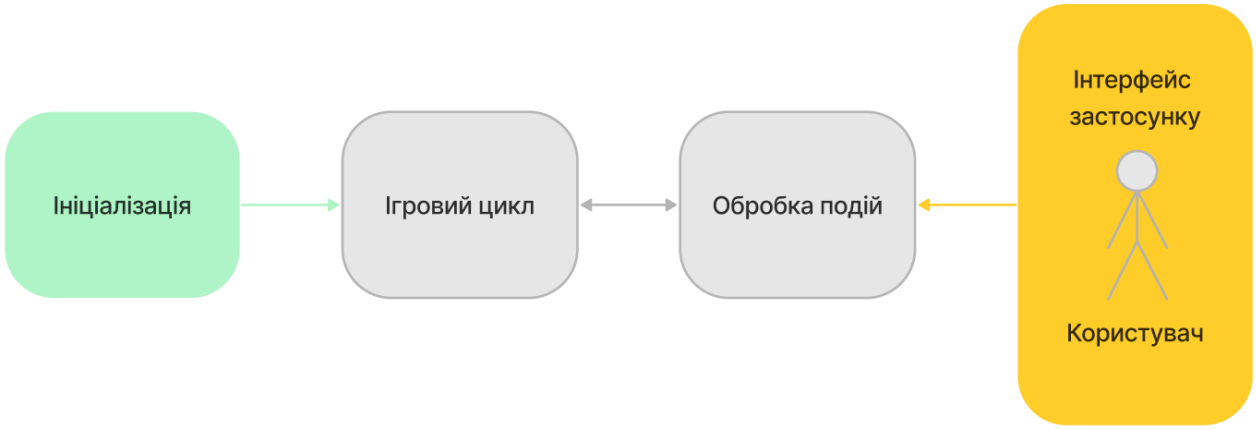

Рис. 2.1. Схема роботи ігрового застосунку

Алгоритм функціонування ігрового продукту включає наступні етапи:

Ініціалізація. На початку гри відбувається завантаження необхідних ресурсів, включаючи Blueprint-класи, C++ класи, ігрові рівні, звукові ефекти та текстури. Далі відбувається створення екземплярів ігрових об'єктів на сцені, таких як гравець, тригери, вороги та платформи. Одночасно здійснюється ініціалізація змінних, що відповідають за стан гри, здоров'я гравця, рахунок та інші параметри.

Ігровий цикл. Основний цикл гри складається з двох ключових фаз: оновлення стану та рендеринг. Під час фази оновлення стану відбувається обробка вхідних даних від гравця (керування персонажем, атаки), розрахунок нової позиції та стану гравця, оновлення стану ворогів та інших об'єктів,

перевірка колізій між акторами, а також оновлення камери та інтерфейсу користувача.

Фаза рендерингу забезпечує візуалізацію ігрової сцени на екрані, відображаючи персонажа, ворогів, платформи, фон та інші елементи (рис. 2.2).

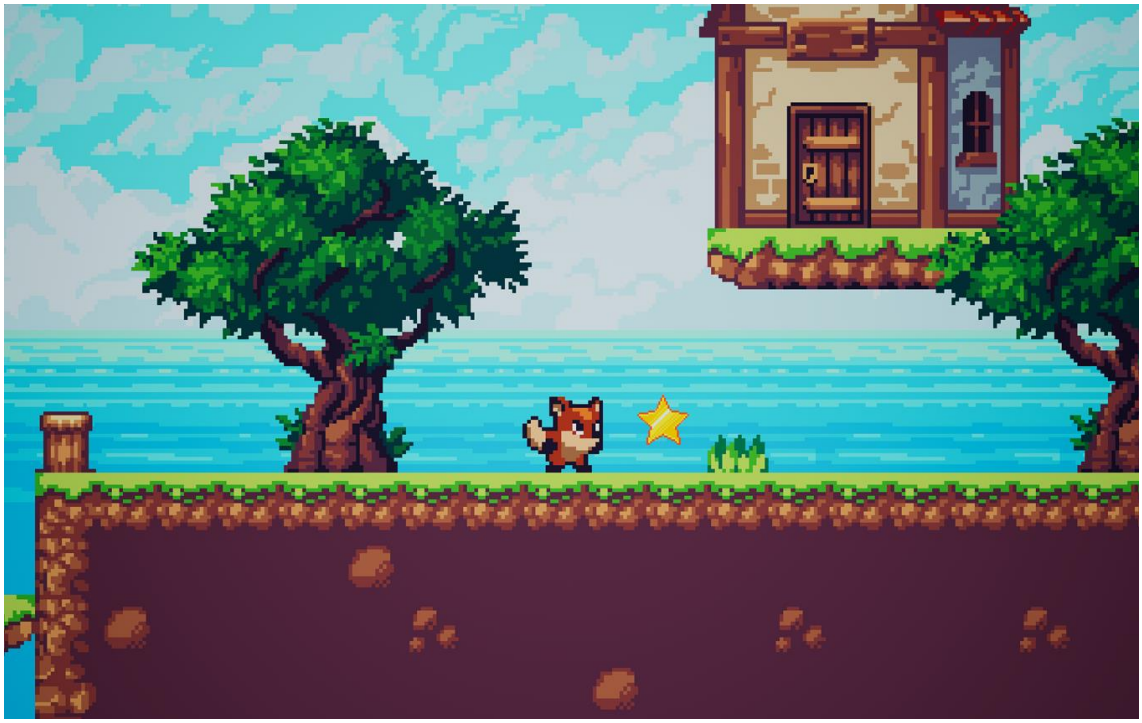

Рис. 2.2. Візуалізація ігрової сцени першого рівня та її елементів

Обробка подій. У процесі гри відбувається обробка різноманітних запрограмованиз подій. Процес взаємодії із рівнем гри відбувається безпосередньо через головного героя, або Character. Він реалізований за допомогою Blueprints. Оскільки гра розроблена в стилі 2D, то персонаж гравця являє собою спрайт, який має свою поведінку руху вліво-вправо, стрибків вгору та анімації. В грі присутня також механіка подвійного стрибка, яка доступна деякий час поки персонаж знаходиться в стані звичайного стрибка. Такий стрибок виконується за допомогою подвійного натискання клавіші за короткий час. Дана механіка буде корисною аби діставатися недоступних місць на рівні.

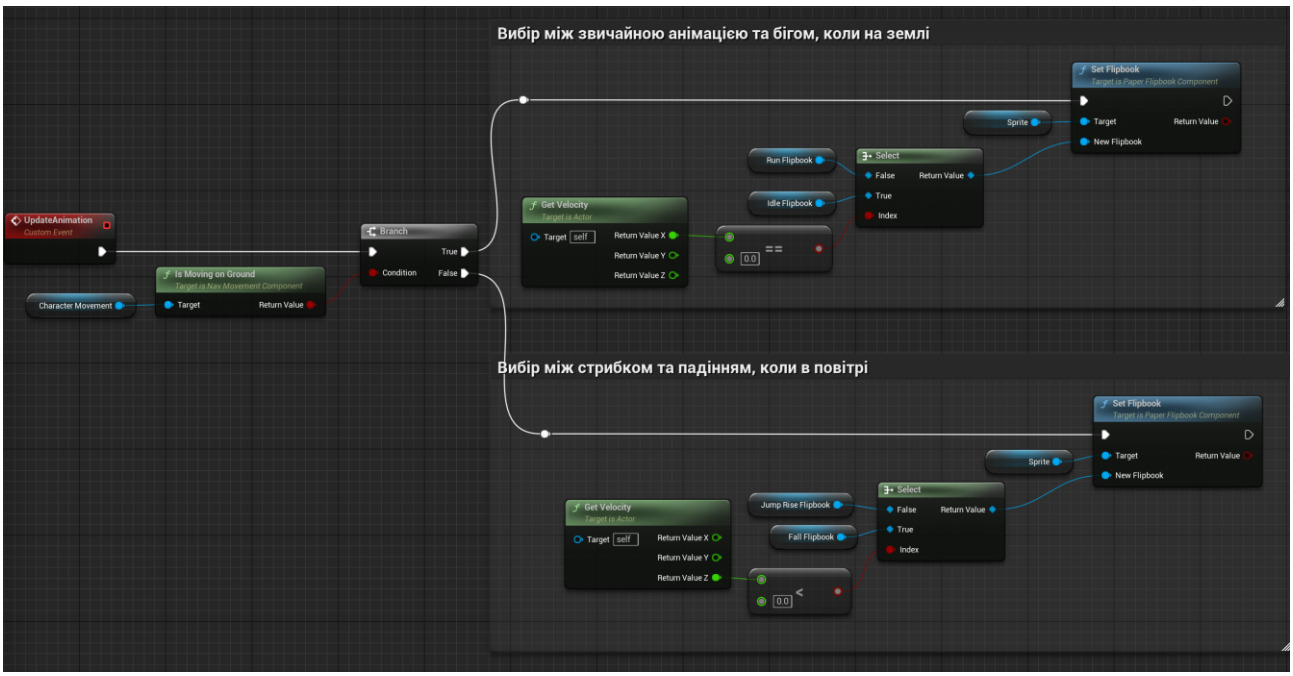

Рис. 2.3. Робота анімації персонажа в Blueprint

Ключове значенян серед обробки подій мають колізії. У контексті розробки ігор, колізія – це процес виявлення та реагування на зіткнення між об'єктами в ігровому світі. Це може бути зіткнення між персонажем гравця та стіною, підлогою, ворогом або іншим об'єктом. Колізії використовуються для того, щоб об'єкти не проходили один крізь одного, а реагували на зіткнення відповідним чином. У Unreal Engine колізії обробляються за допомогою компонентів колізії, які можна додавати до акторів. Ці компоненти визначають форму об'єкта та його фізичні властивості, такі як маса та тертя. Рушій автоматично перевіряє колізії між об'єктами та викликає відповідні події, які можна використовувати для реалізації ігрової логіки.

При зіткненні гравця з тригерами гра реагує по різному. Наприклад, при перетині тригера падіння за межами рівня призводить до повернення на точку спавну.

В грі присутня механіка збору предметів, так званих зірок, які розташовані на кожному з рівнів гри. Класс ItemStar відповідає за принцип роботи цієї ігрової механіки. Основна функція предмета зірка полягає в тому, щоб головний персонаж міг збирати його, після чого вона зникає з рівня, а кількість зібраних предметів зберігається. Зірка з’являється на рівні як окремий об’єкт. Вона, як і

майже всі тригери має компонент для візуалізації, який визначає її зовнішній вигляд та компонент для обробки колізій, де відбувається взаємодія з головним персонажем.

Механізм збереження гри дозволяє зберігати стан гри на певний момент часу, щоб гравець міг повернутися до цього стану пізніше. У нашому випадку, збереження використовується для фіксації кількості зібраних зірок. Збереження дозволяє зафіксувати досягнення гравця, такі як зібрані зірки, пройдені рівні тощо. Це важливо для мотивації гравця і відчуття прогресу в грі.

Компонент для обробки зіткнень зірки має функцію-обробник, яка викликається при зіткненні з іншими об'єктами. Ця функція перевіряє, чи інший об'єкт є головним персонажем. Якщо інший об'єкт є головним персонажем, обробник виконує логіку збирання зірки. При зіткненні з головним персонажем виконуються наступні дії:

1. Перевіряється, чи існує збереження гри за заданим іменем слота. Якщо збереження існує, воно завантажується. Якщо ні, створюється новий об'єкт збереження.
2. Після завантаження збереження кількість зірок збільшується на 1.
3. Оновлені дані (кількість зірок) зберігаються в слот.
4. Після успішного збирання зірка видаляється з рівня за допомогою методу Destroy.

Ще одним персонажем в ігровому застосунку є ворог головного героя. Клас Enemy відповідає за його поведінку у грі. Основний принцип роботи класу полягає в забезпеченні руху ворога вздовж платформи, реалізації взаємодії з головним персонажем та знищення ворога за певних умов. Рух ворога реалізується за допомогою зміни його позиції на ігровому полі. Він переміщується вліво та вправо вздовж платформи, змінюючи напрямок руху при зіткненні з краями платформи або іншими перешкодами. Це створює враження патрулювання території. Схожий принцип роботи вже використовувався в культових іграх цього жанру, тому користувачеві цього застосунку буде одразу зрозуміло як його побороти. Наслідування механік вважається гарною

практикою в ігровій індустрії, адже досвічені гравці мають певний досвід та свої паттерни проходження гри в кожному із жанрів ігор.

Взаємодія з головним персонажем відбувається через механізм колізій. При зіткненні з головним персонажем ворог може завдати йому шкоди, що призводить до зменшення здоров'я або навіть смерті персонажа. Однак, якщо головний персонаж стрибає на ворога зверху, відбувається знищення ворога. Ця механіка додає елемент стратегії та вимагає від гравця вправності та точності.

Гра завершується у випадку коли гравець успішно досягає кінця рівня, що означає перемогу. Після завершення одного рівня, гравець може отримати доступ до нового рівня, за умови збору певної кількості об’єктів зірок на рівні. Таким чином ігровий контент може розвиватися в майбутньому за рахунок створення та ускладнення нових рівнів, додаванню нових ігрових режимів тощо.

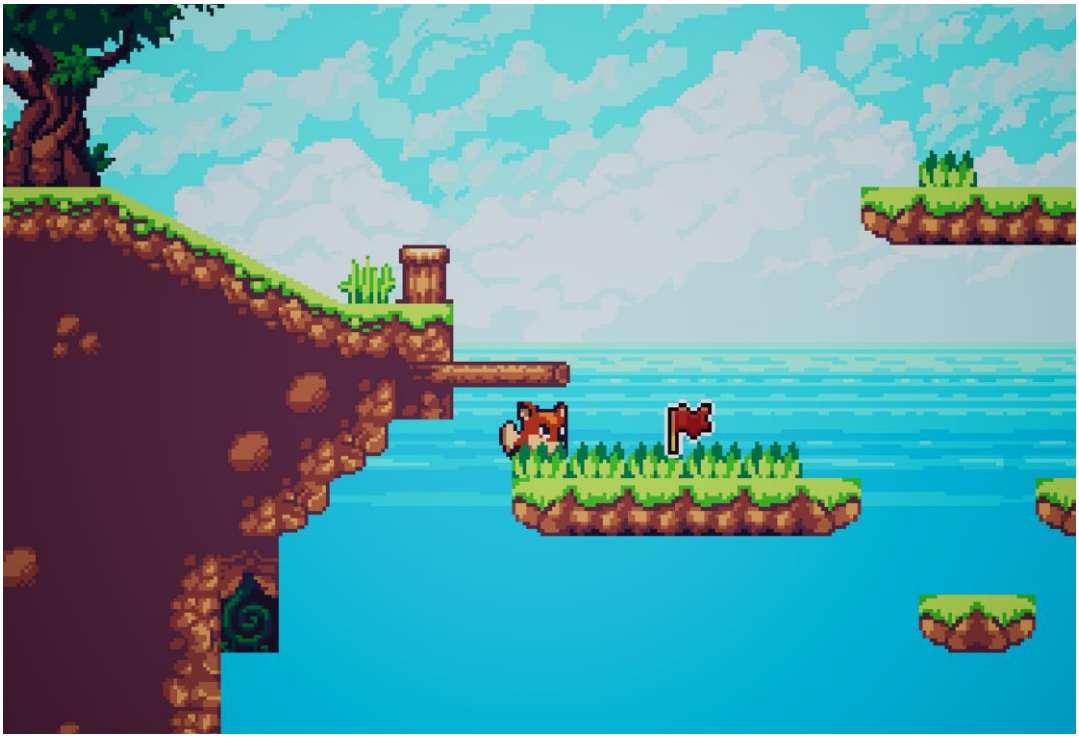

Рис. 2.4. Триггер “LevelEnd” розташований в кінці рівня

При успішному проходженні рівня, гравець бачить перед собою вікно, яке повідомляє про перемогу в грі. Після натискання на кнопку продовження, гра переміщує його до меню вибору рівня, на якому зображена кількість зібраних зірок та розташовані кнопки для переміщення на наступний рівень. Користувач

може проходити заново вже пройдені рівні без ніяких обмежень. А доступ до нових відкривається за умови накопичення певної кількості зірок. Так, наприклад, для рівня 2 потребується 10 зірок, а для рівня 3 – відповідно 30 зірок. Дана логіка не потребує створення окремого класу для цього, тому меню реалізовано за допомогою Blueprints.

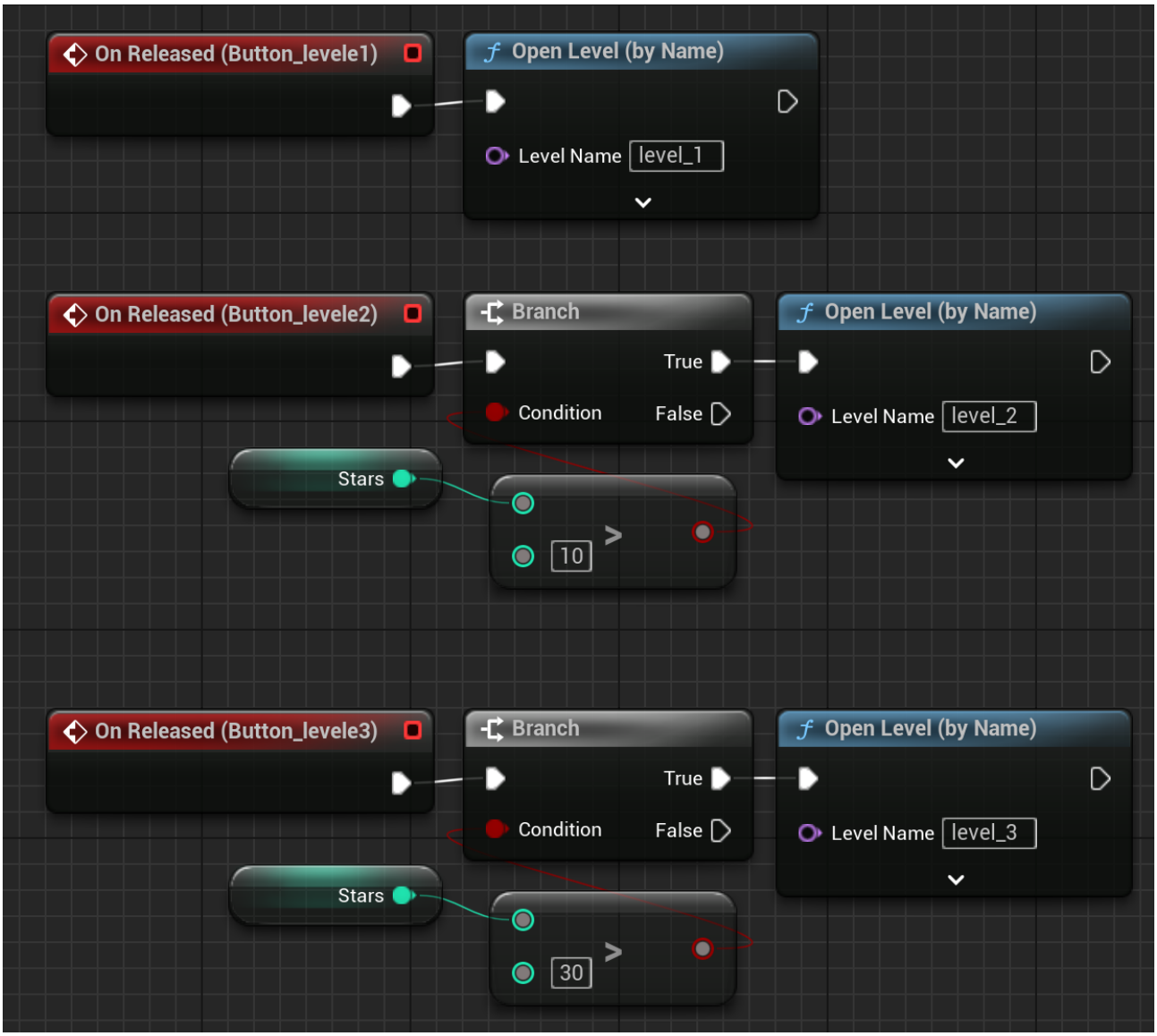

Рис. 2.5. Blueprint меню вибору рівнів

Таким чином під час розробки ігрового застосунку в рамках цієї кваліфікаційної бакалаврської роботи використовується комбінація C++ та Blueprints. C++ має відповідати за реалізацію складної ігрової логіки. Blueprints, у свою чергу, забезпечуватимуть економію часових ресурсів в процесі

візуального створення рівнів, налаштування поведінки акторів, створення інтерфейсу користувача та інших візуальних елементів гри.

Розроблений ігровий продукт являє собою 2D-платформер, де гравець керує персонажем, що переміщується по ігровому світу, збирає предмети та взаємодіє з оточенням. Основні компоненти ігрової системи та їх опис зображено на таблиці 2.1.

Таблиця 2.3

Компоненти гри та їх реалізація

Компонент гри	Опис компонента	Реалізація
Character	Гравець, що керується користувачем, з можливістю руху, стрибків, подвійних стрибків та збору предметів.	Blueprint
LevelEnd	Тригер, що відповідає за завершення рівня.	C++
Enemy	Об’єкт, що переміщуються по рівню та при зіткненні із гравцем, його вбивають.	C++
GameSave	Збереження даних гри (кількість зібраних об’єктів).	C++
Star	Об'єкт, який гравець може збирати для отримання очків.	C++
Maps	Тека рівнів, що мають розташовані платформи та перешкоди для Character.	Unreal Editor
Інтерфейс користувача (UI)	Відображає стан гри (головне меню, пауза, перемога).	Unreal Motion Graphics

		(UMG)

2.5 Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними в моєму ігрового застосунку є команди гравця, що передаються через клавіатуру та мишу. Ці команди визначають рух персонажа та його стрибки. Для управлінням головним героєм зарезервовані клавіші “A” для руху вліво та “D” для руху вправо відповідно. Для стрибка або подвійного стрибка вгору використовується класична для всіх ігор клавіша “Space” або “Пробіл”. Також до вхідних даних належать події колізій, які виникають при взаємодії гравця з об'єктами ігрового світу, такими як вороги, тригери та предмети.

Вихідними даними гри є візуальне відображення ігрового рівня на екрані, включаючи позицію та анімацію ігрового персонажа та ворогів, стан інших об'єктів, зміни в оточенні та інтерфейс користувача. Інтерфейс зустрічає користувача із самого запуску ігрового застосунку. Він відображає поточний стан гри, такий як кількість зібраних зірок, пауза та результат гри.

2.6 Опис розробленої системи

2.6.1.Використані технічні засоби.

Оскільки застосунок розроблений за допомогою комбінації Blueprints та класів C++, програма матиме дещо більший розмір на диску, але не занадто великий. Не зважаючи на це, алгоритми програми не потребують великих обчислень та рендерингу, тому широке коло користувачів може зіграти в цьому застосунку без потреб в потужному обладнанні.

У розробці даного ігрового застосунку вистачає одного комп’ютера. Для роботи рекомендовані наступні засоби:

- персональний комп’ютер чи ноутбук;
- монітор;
- клавіатура;
- маніпулятор “миша”;
- 4 Гб вільного місця на диску;
- не менше 3 Гб оперативної пам’яті;
- процесор із тактовою частотою 2.5 Ггц.

2.6.2 Використані програмні засоби

Для розробки ігрового продукту було обрано рушій Unreal Engine 5, який надає потужні інструменти та можливості для створення високоякісних 2D- та 3D-ігор. Зокрема, в рамках проєкту використовувалися наступні компоненти Unreal Engine 5.

C++. Мова програмування C++ була застосована для реалізації основної ігрової логіки, управління станом гри, обробки колізій, взаємодії об'єктів та інших ключових аспектів геймплею. Завдяки використанню C++, забезпечується висока продуктивність та ефективність роботи гри.

Blueprints: Візуальна система скриптування Blueprints була використана для створення ігрових рівнів, налаштування поведінки персонажів та інших об'єктів, реалізації анімацій, а також для створення інтерфейсу користувача. Blueprints дозволили спростити та прискорити процес розробки, надаючи можливість візуально налаштовувати логіку гри без необхідності написання коду.

Paper2D – це плагін для рушія Unreal Engine 5 був використаний для створення 2D-графіки та анімацій персонажів, ворогів, платформ та інших елементів ігрового світу (рис. 2.6).

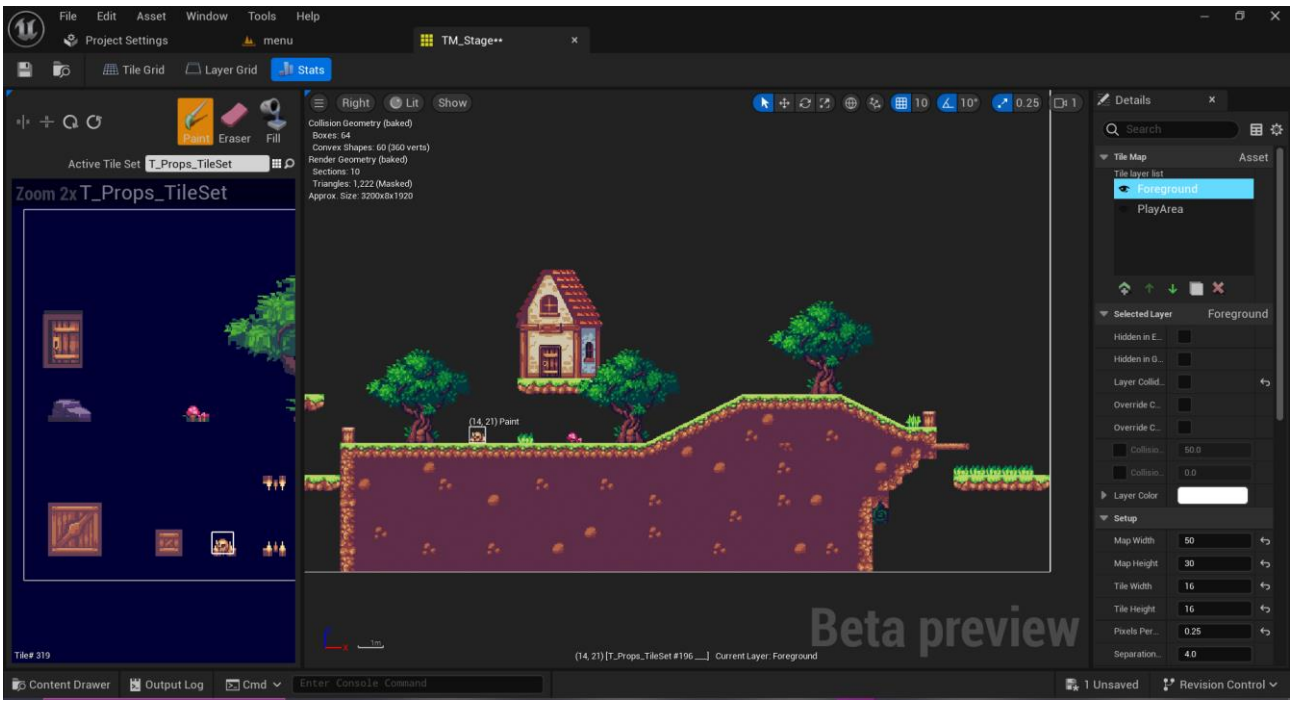

Рис. 2.6. Зовнішній вигляд редактору рівня Paper2D

Він надає зручні інструменти для роботи з спрайтами та анімаціями.[10] Також плагін був ключовим інструментом для створення спрайтів головного героя та його анімацій. Кожен спрайт - це окреме зображення персонажа в певній позі або фазі руху [12]. Для створення анімації бігу, стрибка та атаки було створено набір спрайтів, які послідовно змінюють один одного, створюючи ілюзію руху (рис. 2.7).

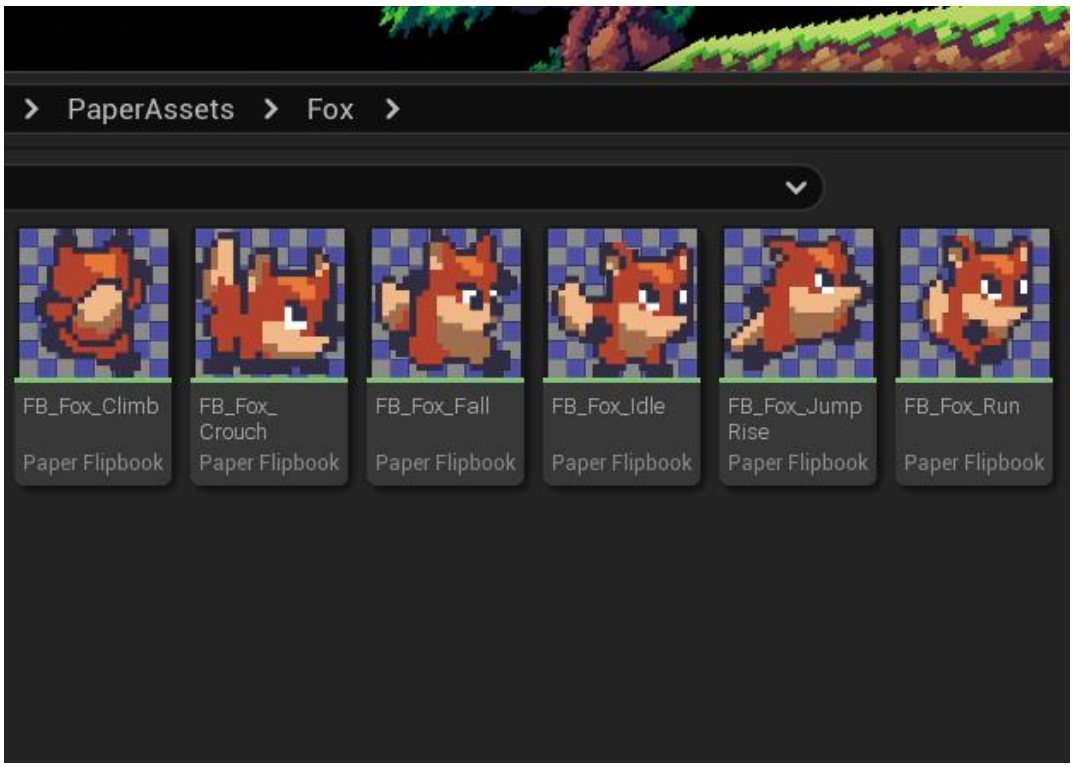

Рис. 2.7. Варіації спрайту головного героя в редакторі застосунку

Paper2D Flipbook - це інструмент, що дозволяє об'єднати ці спрайти в єдину анімацію. Налаштування частоти зміни кадрів та інших параметрів анімації відбувалося безпосередньо в Unreal Editor. Потім ці анімації були прив'язані до відповідних дій персонажа у Blueprint, забезпечуючи плавний та реалістичний рух героя під час гри [11].

Для розробки C++ коду використовувалося інтегроване середовище розробки (IDE) Visual Studio 2022, яке надає потужні інструменти для написання, налагодження та компіляції коду.

Для тестування та налагодження ігрового процесу використовувався вбудований в Unreal Engine 5 редактор рівнів та інструменти для профілювання продуктивності.

Ігровий застосунок стабільно працює на комп’ютерному обладнанні з операційною системою Windows 10, 11 32 чи 64-розрядних версій. З боку користувача застосунку, завантаження Unreal Engine та його плагінів не потребується.

2.6.3 Виклик та завантаження програми

Процес тестування та налагодження здійснюється безпосередньо в середовищі Unreal Editor, використовуючи його потужні вбудовані інструменти.

Режим гри (Play Mode) надає можливість запускати гру в реальному часі, що дозволяє ретельно перевірити ігрову логіку, механіки та візуальне відображення [14]. Крім того, редактор рівнів дозволяє створювати та редагувати ігрові рівні, розміщувати об'єкти, налаштовувати їх властивості та тестувати взаємодію між ними, забезпечуючи повний контроль над дизайном рівнів.

Консоль налагодження є незамінним інструментом для виведення повідомлень та змінних під час виконання гри, що допомагає відстежувати помилки та аналізувати роботу коду [15].

Важливою складовою процесу розробки є профілювання продуктивності, яке дозволяє аналізувати продуктивність гри, виявляти вузькі місця та оптимізувати код для досягнення плавного ігрового процесу, забезпечуючи найкращий користувацький досвід. Asset Loading Insights надає спосіб профілювання кількості часу, необхідного для завантаження ресурсів проекту в UnrealEngine. Це може дати вам детальне розуміння наборів даних для кожного типу активу. Asset Loading Insights базується на даних, отриманих із каналу трасування AssetLoadTime. Unreal Cooking Insights дозволяє збирати та відображати інформацію про спосіб “приготування” пакетів у проєкті. Відображаючи час, необхідний для приготування кожного пакета, можна спостерігати, на оптимізацію яких пакетів слід зосередити увагу [16].

Після завершення розробки та компіляції, запуск готового застосунку здійснюється через виконуваний файл (exe), що робить його доступним для кінцевих користувачів.

2.6.4 Опис інтерфейсу користувача

Інтерфейс користувача мого ігрового застосунку виконаний у ретро-стилі, що нагадує 8-бітну графіку [13]. Головне меню складається з фону, що зображує

блакитне небо з хмарами, та трьох основних кнопок: "Почати гру", "Вибір рівня" та "Вийти". Назва гри "Paws and Magic" розміщена у верхній частині екрана та виконана з використанням яскравих кольорів, що привертають увагу гравця. Кнопки меню мають зрозумілу для досвічених гравців структуру, що забезпечує зручність навігації. Загальний дизайн інтерфейсу створює атмосферу ностальгії та відсилає до класичних платформерів (рис. 2.8).

Рис. 2.8. Головне меню застосунку

Кнопка "Почати гру" реалізована за допомогою віджету Button в Unreal Motion Graphics (UMG). При натисканні на неї викликається подія "OnClicked", яка пов'язана з Blueprint-функцією, що виконує наступні дії: закриває головне меню, завантажує перший рівень гри та ініціалізує ігровий процес (рис. 2.9)

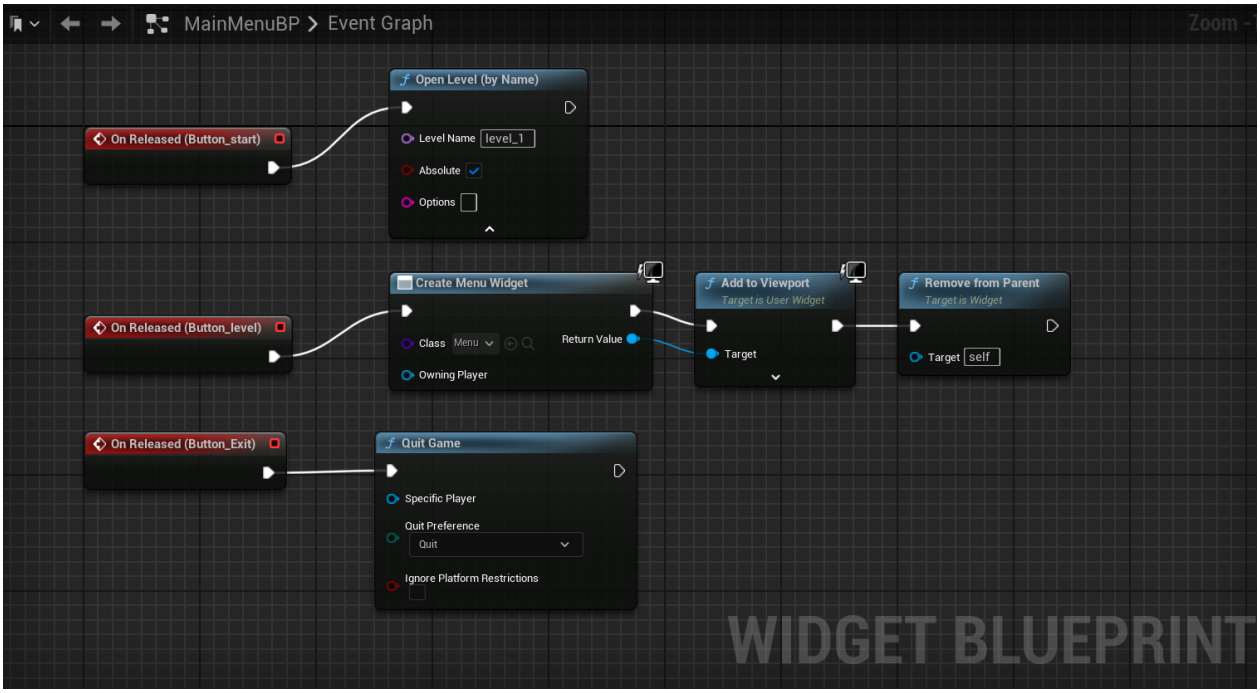

Рис. 2.9. Схема Blueprint для головного меню

Кнопка "Вибір рівня" також реалізована за допомогою віджету Button в UMG. При натисканні на неї викликається подія, яка відкриває новий віджет, що представляє меню вибору рівня. Це меню містить кнопки для вибору рівня та кількість набраних об'єктів "Зірок". Доступ до рівнів відкривається при певній кількості цих об'єктів. При натисканні на кнопку викликається Blueprint-функція, яка закриває меню вибору рівня, завантажує обраний рівень та ініціалізує ігровий процес (рис. 2.10).

Рис. 2.10. Меню вибору рівня

Кнопка "Вийти" також реалізована за допомогою віджету Button в UMG. При натисканні на неї викликається подія "OnClicked", яка пов'язана з функцією Quit Game в Unreal Engine. Ця функція негайно завершує роботу гри та повертає користувача до операційної системи.

Інтерфейс користувача рівня Level1 гри "Paws and Magic" виконаний у мінімалістичному стилі, щоб не перевантажувати візуально екран та забезпечити максимальне занурення гравця в ігровий процес. Всі необхідні маніпуляції, такі як пауза (клавіша P), здійснюються за допомогою клавіатури (рис. 2.11).

Рис. 2.11. Меню паузи

У разі смерті персонажа на екрані з'являється екран поразки (рис. 2.12). Він містить повідомлення про програш, інформацію про кількість залишених життів та можливість перезапуску рівня або повернення до головного меню. Дизайн екрану поразки також відповідає загальному стилю гри, використовуючи піксельні шрифти та графіку.

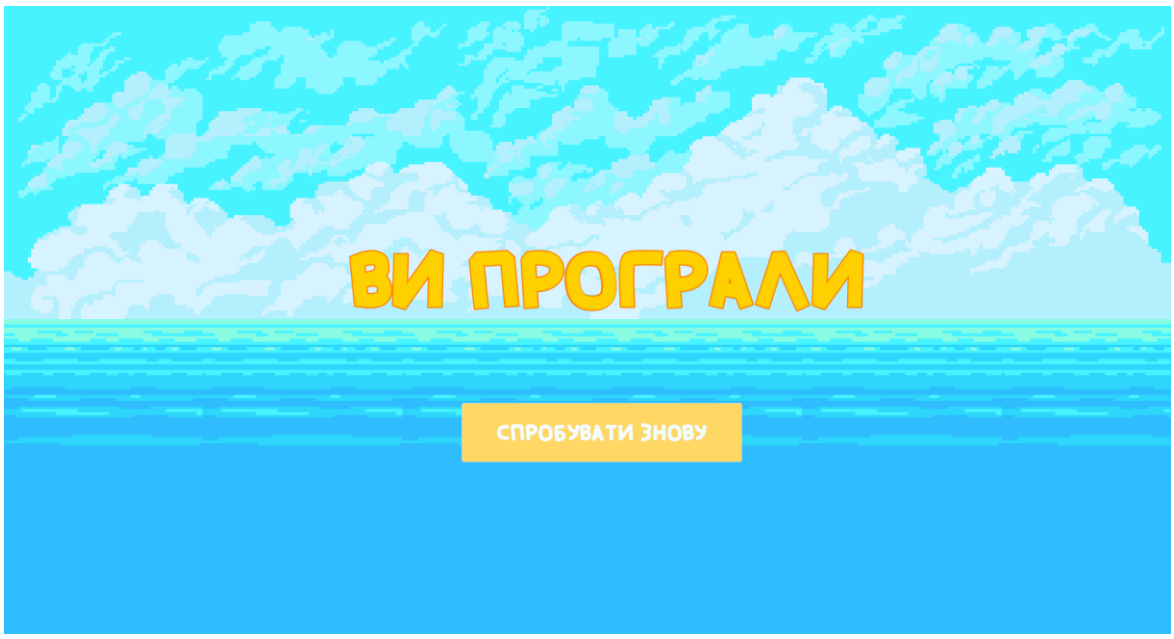

Рис. 2.12. Сповіщення про поразку

Після успішного завершення рівня на екрані з'являється екран перемоги (рис 2.13). Він містить привітання гравця з перемогою, інформацію про кількість зібраних монет та можливість переходу на наступний рівень. Екран виконаний зі збереженням основної кольорової гами.

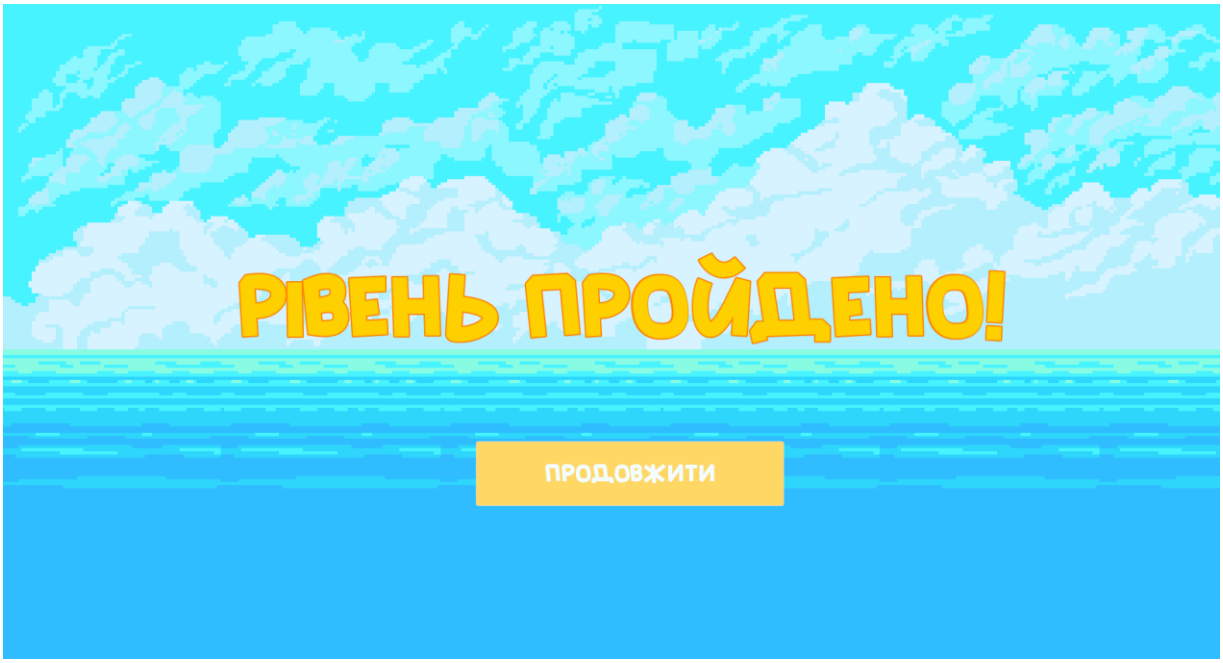

Рис. 2.13. Сповіщення про поразку

Інтерфейс застосунку передбачає всі стани та результати гри. Дизайн інтерфейсу зроблений, притримуючись принципів мінімалізму, адже так

користувачам гри стає легше користуватися застосунком та орієнтуватися в ньому. Кольорова гама дотримана за стилістикою навколинього світу в грі, а контрастність елементів роблять інтерфейс доступним для людей, які мають вади зору [17].

Із можливих покращень інтерфейсу можна зазначити додавання попереднього перегляду, або зображення фрагмента рівня для більшої наочності. Також можна оновити зовнішній вигляд елементів інтерфейсу, таких як кнопки та написи, щоб вони були також в піксельному стилі. Дані ідеї можуть бути успішно імплементовані в наступних оновленнях ігрового застосунку.

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані для розробки програмного забезпечення мають такі характеристики:

Початкові дані:

1. передбачуване число операторів програми – 750;
2. коефіцієнт складності програми – 1,2;
3. коефіцієнт корекції програми – 0,2;
4. годинна заробітна плата програміста – 920,7 грн/год;

За статистичними даними платформи Jobble щодо вакансій розробників на рушії Unreal Engine 4/5, станом на 21 червня 2024 середня заробітна плата позиції Unreal Engine розробник становить 3 655,43$ [19], що при курсі валют долара до гривні (40,45 грн), згідно НБУ прирівнюється до 147 313 грн. Отже, середня заробітна плата програміста в даній галузі становитиме 920,7 грн/год.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,4;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 1.1 грн/год.

Під час розрахунків було проведено наступний алгоритм: ціна за електроенергію для населення в Україні на момент 21 червня 2024 року у разі споживання понад 2000 кВт на місяць становить 4,32 грн кВт/год (з урахуванням ПДВ) [20]. На основі споживання електроенергії робочим пристроєм, а саме 170 Вт на годину, витрати під час роботи на електроенергію складають 0,170*4,32=0.7344. Додатково, підключення до інтернет – провайдеру «Київстар» проводиться за умовами тарифного плану «Домашній інтернет», щомісячна оплата якого складає 275 грн [18]. Таким чином, вартість використання погодинно становить 0,38 грн. Отже, загальна вартість ЕОМ дорівнює наближено 1.1 грн.

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Розрахунок трудомісткості розробки програмного забезпечення є складним завданням через творчий характер праці програміста. Для цього використовуються різні моделі, які дозволяють оцінити трудомісткість з різним ступенем точності.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин, (3.1)}$$

де t_o – витрати праці на підготовку й опис поставленої задачі

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

Де q – передбачуване число операторів;

C– коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1{,}2 * 750 * (1 + 0{,}2) = 1080;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин,} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1,35;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності становить 1,3;

$$t_u = (1080 * 1.3) / 80 * 1.2 = 14.625, \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K} \qquad (3.4)$$

$$t_a = 1080 / 20 * 1.2 = 45, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \qquad (3.5)$$

$$t_n = 1080 / 25 * 1.2 = 36, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K} \tag{3.6}$$

$t_{отл}$ = 1080 / 5 * 1.2 = 259.2, людино-годин,

- за умови комплексного налагодження завдання:

$$t^{k}_{отл} = 1{,}5 \cdot t_{отл} \text{ , людино-годин.} \tag{3.7}$$

$t^{k}_{отл}$ = 1,5 * 259.2 = 388.8, людино-годин

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до} \tag{3.8}$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K} \tag{3.9}$$

$t_{др}$ = 1080 / (16 * 1,2) = 56.25, людино-годин.

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др} \quad (3.10)$$

$t_{до}$ = 0,75 · 56.25 = 42.19, людино-годин.

$t_{д}$ = 56.25 + 42.19 = 98.44, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 14.625 + 45 + 36 + 259.2 + 98.44 = 503.27, людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}, \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн}, \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$З_{ЗП}$ = 503.27 * 920.7 = 463360.69, грн.

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$ЗМВ = tотл \cdot СМ, \text{ грн}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{мв} = 2592 * 0.93= 2410.56, \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$K_{ПО} = 2410.56 + 463360.69 = 465771.25, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де t – загальна трудомісткість, людино годин;

B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = 503.27 / 1 \cdot 176 \approx 2.86 \text{ міс.}$$

Висновки. Час розробки даного програмного забезпечення складає 503.27 людино-годин. Таким чином, очікувана тривалість розробки складе 2.86 місяців при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 465771.25 грн.

ВИСНОВКИ

Розробка ігрового застосунку «Paws & Magic» дала помітні результати і продемонструвала високий ступінь новизни та актуальності. Завдяки ретельному дослідженню, аналізу та впровадженню, було створено продукт, який забезпечує якісний та зрозумілий ігровий досвід з використанням сучасних технологій розробки.

Метою даної кваліфікаційної роботи була розробка ігрового застосунку "Paws and Magic", що поєднує в собі класичні елементи жанру 2D-платформера з сучасними технологіями розробки.

Практична цінність даного застосунку очевидна у його здатності поєднувати класичні елементи жанру платформера з сучасними графічними можливостями та інтерактивними ігровими механіками. Гра пропонує користувачам захопливу подорож ігровим світом, де вони можуть долати перешкоди, взаємодіяти з оточенням та боротися з ворогами. Це сприяє не лише розвазі, але й розвитку когнітивних навичок та стратегічного мислення у гравців.

Завдяки використанню потужного рушія Unreal Engine 5, гра має високу якість графіки, плавну анімацію та зручний інтерфейс користувача. Крім того, застосування комбінації мови програмування C++ та Blueprints дозволило досягти високої швидкості та гнучкості у розробці та тестуванні гри.

З наукової точки зору, дослідження, проведені в рамках даної роботи, сприяли поглибленню знань у сфері розробки ігрових застосунків. Проектування, впровадження та оцінка гри продемонстрували ефективність використання сучасних технологій та методів у створенні інтерактивних застосунків. Результати дослідження можуть стати основою для майбутніх проектів та розробок у галузі ігрової індустрії.

Економічний аналіз, проведений у рамках цієї роботи, надав цінну інформацію про трудомісткість (503.27 людино-годин) та вартість розробки програмного продукту (465771.25 грн). Оцінки та розрахунки допомогли визначити фінансову життєздатність проєкту, розподіл ресурсів та потенційну

рентабельність інвестицій. Це надає зацікавленим сторонам важливу інформацію для прийняття обґрунтованих рішень щодо подальшого розвитку проєкту.

З прогнозної точки зору, очікується, що ігровий застосунок "Paws & Magic" буде продовжувати розвиватися і адаптуватися до нових технологій та вимог ринку. Масштабованість та гнучкість гри дозволяють впроваджувати майбутні вдосконалення, такі як інтеграція нових більш складних алгоритмів штучного інтелекту для підвищення інтелектуального рівня ворогів, додавання нових складних рівнів із динамічними платформами та інших ігрових елементів. Застосунок має потенціал для успішного застосування у сфері розваг, роблячи вклад у розвиток української ігрової індустрії.

Таким чином, розробка ігрового застосунку "Paws & Magic" продемонструвала значні результати і має як практичну, так і наукову значимість. Завдяки своїм новим функціям та високій якості реалізації та потенціалу для подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зростання ігрового бізнесу та український геймдев / URL: https://minfin.com.ua/ua/2024/05/21/127364918/. дата звернення 15.05.2024.
2. How Many Gamers Are There? / URL: https://explodingtopics.com/blog/number-of-gamers (дата звернення 15.06.2024)
3. Як Україна виглядає на світовому ринку розробки ігор / URL: https://businessviews.com.ua/ru/tech/id/virobnictvo-igor-v-ukrajini-1948/. дата звернення 15.05.2024.
4. Last looks: The global games market in 2023 / URL: https://newzoo.com/resources/blog/last-looks-the-global-games-market-in-2023. дата звернення 15.05.2024.
5. Інтерв'ю з директором Marevo Collective — про гру Zero Losses з окупантом у головній ролі, вибір теми війни та успіх попереднього проєкту / URL: https://gamedev.dou.ua/articles/interview-with-indie-game-director-marevo-collective/. дата звернення 10.06.2024.
6. Unity проти Unreal Engine — який рушій обрати для гри та чому. Зважуємо всі за та проти з розробниками / URL: https://gamedev.dou.ua/articles/unity-or-unreal-engine/. дата звернення 10.06.2024.
7. Хто такий Unreal Engine Developer в ігровій індустрії. Кар'єра в геймдеві / URL: https://gamedev.dou.ua/articles/unreal-engine-developer-in-gamedev/. дата звернення 11.06.2024.
8. Getting started with Direct3D / URL: https://learn.microsoft.com/en-us/windows/win32/getting-started-with-direct3d (дата звернення 11.06.2024)
9. 10 Types of Platforms in Platform Video Games / URL: https://www.idtech.com/blog/10-types-of-platforms-in-platform-video-games. дата звернення 11.06.2024.
10. Paper 2D | Unreal Engine 4.27 Documentation / URL: https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/Paper2D/. .дата звернення 12.06.2024.

11. Paper 2D Sprites | Unreal Engine 4.27 Documentation / URL:: https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/Paper2D/Sprites/. дата звернення 12.06.2024.

12. 2D SPRITES IN GAME DESIGN: WHY GAME ARTISTS STILL USE THEM / URL: https://retrostylegames.com/blog/2d-sprites-in-game-design/. дата звернення 22.06.2024.

13. Pixel Art/8 bit Graphics-Style Use in Modern Games – Rising Trends / URL: https://logicsimplified.com/newgames/rising-trend-of-pixel-art8-bit-graphic-style-use-in-modern-games/. дата звернення 17.05.2024.

14. Playtesting and Simulating your game with the Play and Simulate options / URL: https://docs.unrealengine.com/4.27/en-US/Basics/PIE/. дата звернення 22.06.2024.

15. Unreal Engine | Console Commands / URL: https://docs.unrealengine.com/udk/Two/ConsoleCommands.html. дата звернення 22.06.2024.

16. Unreal Insights . / URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-insights-in-unreal-engine. дата звернення 22.06.2024.

17. Design and accessibility – use of colour / URL: https://www.accessibilityoz.com/2014/07/design-accessibility-use-colour/. дата звернення 17.06.2024.

18. Київстар | Домашній Інтернет - Тарифи / URL: https://kyivstar.ua/tariffs-home-internet. дата звернення 21.06.2024.

19. Unreal engine developer: середня зарплата в Україні: / URL: https://ua.jooble.org/salary/unreal-engine-developer?salarySearchSource=1. дата звернення 21.06.2024.

20. Тарифи на електроенергію для побутових споживачів з 01.06.2024 р.: / URL: https://yasno.com.ua/news/all_news/electricity-tariffs-for-household-consumers-from-01-06-2024. дата звернення 21.06.2024.

ДОДАТОК А

КОД ПРОГРАМИ

Enemycharacter.h // Заголовний файл

```
#pragma once

#include "CoreMinimal.h"
#include "PaperCharacter.h" // Підключаємо заголовковий файл PaperCharacter
#include "EnemyCharacter.generated.h"

UCLASS()
class TPTWODSIDEKSCROLLERBP_API AEnemyCharacter : public APaperCharacter
{
	GENERATED_BODY()

public:
	// Встановлює значення за замовчуванням для властивостей цього персонажа
	AEnemyCharacter();

protected:
	// Викликається, коли гра починається або коли цей персонаж з'являється
	virtual void BeginPlay() override;

public:
	// Викликається кожного кадру
	virtual void Tick(float DeltaTime) override;

	// Функція для обробки зіткнення з головним героєм
	UFUNCTION()
	void OnOverlapBegin(class UPrimitiveComponent* OverlappedComp, class AActor*
OtherActor, class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep,
const FHitResult& SweepResult);

private:
	// Напрямок руху
	FVector MovementDirection;

	// Швидкість ворога
	UPROPERTY(EditAnywhere)
	float Speed;

	// Функція для обробки смерті ворога
	void Die();
};
```

Enemycharacter.cpp // Ворожий персонаж

```
#include "EnemyCharacter.h"

#include "BP_SideScrollerCharacter.h" // Підключаємо заголовковий файл головного героя
```

```
#include "Components/CapsuleComponent.h"

// Встановлює значення за замовчуванням
AEnemyCharacter::AEnemyCharacter()
{
	// Встановлює, що цей персонаж викликає Tick() кожного кадру
	PrimaryActorTick.bCanEverTick = true;

	// Ініціалізує швидкість
	Speed = 200.0f;

	// Встановлює напрямок руху ліворуч спочатку
	MovementDirection = FVector(-1.0f, 0.0f, 0.0f);

	// Прив'язує функцію OnOverlapBegin до події OnComponentBeginOverlap
	GetCapsuleComponent()->OnComponentBeginOverlap.AddDynamic(this,
&AEnemyCharacter::OnOverlapBegin);
}

// Викликається, коли гра починається або коли цей персонаж з'являється
void AEnemyCharacter::BeginPlay()
{
	Super::BeginPlay();
}

// Викликається кожного кадру
void AEnemyCharacter::Tick(float DeltaTime)
{
	Super::Tick(DeltaTime);

	// Рухає персонажа ворога
	AddMovementInput(MovementDirection, Speed * DeltaTime);
}

// Функція для обробки зіткнення з головним героєм
```

```
void AEnemyCharacter::OnOverlapBegin(UPrimitiveComponent* OverlappedComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult)
{
	// Перевіряє, чи є інший актор головним героєм
	if (OtherActor->IsA(ABP_SideScrollerCharacter::StaticClass()))
	{
		ABP_SideScrollerCharacter* MainCharacter = Cast<ABP_SideScrollerCharacter>(OtherActor);
		if (MainCharacter)
		{
			// Перевіряє, чи головний герой знаходиться над ворогом
			if (MainCharacter->GetActorLocation().Z > GetActorLocation().Z)
			{
				// Вбиває ворога
				Die();
			}
			else
			{
				// Вбиває головного героя
				MainCharacter->Destroy(); // Гарантуємо, що MainCharacter має Destroy() функцію
			}
		}
	}
}

// Функція для обробки смерті ворога
void AEnemyCharacter::Die()
{
	Destroy();
```

LevelEnd.h // Заголовочний файл завершення рівня

```
#pragma once
```

```
#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "LevelEnd.generated.h"

UCLASS()
class TPTWODSIDEKSCROLLERBP_API ALevelEnd : public AActor
{
	GENERATED_BODY()

public:
	// Встановлює значення за замовчуванням для цього актора
	ALevelEnd();

protected:
	// Викликається при початку гри або при спавні актора
	virtual void BeginPlay() override;

public:
	// Викликається кожного кадру
	virtual void Tick(float DeltaTime) override;

	// Компонент для обробки зони завершення рівня
	UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Components")
	class UBoxComponent* BoxComponent;

	// Клас віджета для відображення при завершенні рівня
	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "UI")
	TSubclassOf<class UUserWidget> PeremogaWidgetClass;

	// Функція для обробки перекриття з гравцем
	UFUNCTION()
	void OnOverlapBegin(class UPrimitiveComponent* OverlappedComp, class AActor* OtherActor, class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult);
```

```
private:
	// Функція для відкриття рівня "menu"
	void OpenMenuLevel();
};
```

LevelEnd.cpp // Завершення рівня

```
#include "LevelEnd.h"
#include "Components/BoxComponent.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"

// Встановлює значення за замовчуванням
ALevelEnd::ALevelEnd()
{
	// Встановлює, що цей актор викликає Tick() кожного кадру
	PrimaryActorTick.bCanEverTick = true;

	// Створює і налаштовує BoxComponent
	BoxComponent = CreateDefaultSubobject<UBoxComponent>(TEXT("BoxComponent"));
	RootComponent = BoxComponent;

	// Прив'язує функцію OnOverlapBegin до події OnComponentBeginOverlap
	BoxComponent->OnComponentBeginOverlap.AddDynamic(this,
&ALevelEnd::OnOverlapBegin);
}

// Викликається при початку гри або при спавні актора
void ALevelEnd::BeginPlay()
{
	Super::BeginPlay();
}

// Викликається кожного кадру
void ALevelEnd::Tick(float DeltaTime)
{
```

```
	Super::Tick(DeltaTime);
}

// Функція для обробки перекриття з гравцем
void ALevelEnd::OnOverlapBegin(UPrimitiveComponent* OverlappedComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult)
{
	// Перевіряє, чи інший актор є головним героєм
	if (OtherActor->IsA(ABP_SideScrollerCharacter::StaticClass()))
	{
		// Створює та відображає віджет Peremoga
		if (PeremogaWidgetClass != nullptr)
		{
			UUserWidget* PeremogaWidget = CreateWidget<UUserWidget>(GetWorld(), PeremogaWidgetClass);
			if (PeremogaWidget != nullptr)
			{
				PeremogaWidget->AddToViewport();
			}
		}

		// Викликає функцію для відкриття рівня "menu"
		OpenMenuLevel();
	}
}

// Функція для відкриття рівня "menu"
void ALevelEnd::OpenMenuLevel()
{
	// Відкриває рівень "menu"
	UGameplayStatics::OpenLevel(this, FName("menu"));
}
```

MySaveGame.h // Заголовочний файл збереження гри

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/SaveGame.h"
#include "MySaveGame.generated.h"

UCLASS()
class TPTWODSIDEKSCROLLERBP_API UMySaveGame : public USaveGame
{
	GENERATED_BODY()

public:
	// Кількість зірок
	UPROPERTY(VisibleAnywhere, Category = Basic)
	int32 Stars;
};
```

MySaveGame.cpp // Збереження гри

```
#include "MySaveGame.h"

UMySaveGame::UMySaveGame()
{
	Stars = 0;
}
```

ItemStar.h // Заголовочний файл для зірки

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "ItemStar.generated.h"

UCLASS()
class TPTWODSIDEKSCROLLERBP_API AItemStar : public AActor
{
```

```
	GENERATED_BODY()

public:
	// Встановлює значення за замовчуванням для цього актора
	AItemStar();

protected:
	// Викликається при початку гри або при спавні актора
	virtual void BeginPlay() override;

public:
	// Викликається кожного кадру
	virtual void Tick(float DeltaTime) override;

	// Компонент для обробки зіткнень
	UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Components")
	class UBoxComponent* BoxComponent;

	// Компонент для візуального відображення зірки
	UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Components")
	class UPaperSpriteComponent* StarSprite;

	// Клас збереження гри
	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "SaveGame")
	TSubclassOf<class UMySaveGame> SaveGameClass;

	// Функція для обробки зіткнення з головним персонажем
	UFUNCTION()
	void OnOverlapBegin(class UPrimitiveComponent* OverlappedComp, class AActor* OtherActor, class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult);

private:
	// Функція для збереження кількості зірок
	void SaveStarCount();
```

```
};
```

ItemStar.cpp // Механіка роботи зірки та її накопичення

```
#include "ItemStar.h"
#include "Components/BoxComponent.h"
#include "PaperSpriteComponent.h"
#include "Kismet/GameplayStatics.h"
#include "MySaveGame.h"
#include "BP_SideScrollerCharacter.h"

// Встановлює значення за замовчуванням
AItemStar::AItemStar()
{
	// Встановлює, що цей актор викликає Tick() кожного кадру
	PrimaryActorTick.bCanEverTick = true;

	// Створює і налаштовує BoxComponent
	BoxComponent = CreateDefaultSubobject<UBoxComponent>(TEXT("BoxComponent"));
	RootComponent = BoxComponent;

	// Створює і налаштовує StarSprite
	StarSprite = CreateDefaultSubobject<UPaperSpriteComponent>(TEXT("StarSprite"));
	StarSprite->SetupAttachment(RootComponent);

	// Прив'язує функцію OnOverlapBegin до події OnComponentBeginOverlap
	BoxComponent->OnComponentBeginOverlap.AddDynamic(this,
&AItemStar::OnOverlapBegin);
}

// Викликається при початку гри або при спавні актора
void AItemStar::BeginPlay()
{
	Super::BeginPlay();
}
```

```
// Викликається кожного кадру
void AItemStar::Tick(float DeltaTime)
{
	Super::Tick(DeltaTime);
}

// Функція для обробки зіткнення з головним персонажем
void AItemStar::OnOverlapBegin(UPrimitiveComponent* OverlappedComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult)
{
	// Перевіряє, чи інший актор є головним героєм
	if (OtherActor->IsA(ABP_SideScrollerCharacter::StaticClass()))
	{
		// Зберігає кількість зірок
		SaveStarCount();

		// Видаляє зірку з рівня
		Destroy();
	}
}

// Функція для збереження кількості зірок
void AItemStar::SaveStarCount()
{
	// Перевіряє, чи існує файл збереження
	if (UGameplayStatics::DoesSaveGameExist(TEXT("test"), 1))
	{
		// Завантажує існуюче збереження
		UMySaveGame* LoadGameInstance = Cast<UMySaveGame>(UGameplayStatics::LoadGameFromSlot(TEXT("test"), 1));
		if (LoadGameInstance)
		{
			// Збільшує кількість зірок
			LoadGameInstance->Stars += 1;
```

```
                // Зберігає зміни
                UGameplayStatics::SaveGameToSlot(LoadGameInstance, TEXT("test"), 1);
            }
        }
        else
        {
            // Створює новий об'єкт збереження
            UMySaveGame* SaveGameInstance =
Cast<UMySaveGame>(UGameplayStatics::CreateSaveGameObject(SaveGameClass));
            if (SaveGameInstance)
            {
                // Встановлює кількість зірок
                SaveGameInstance->Stars = 1;

                // Зберігає зміни
                UGameplayStatics::SaveGameToSlot(SaveGameInstance, TEXT("test"), 1);
            }
        }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_Жукова.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Жукова.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми
Додаткові файли	
Pawsnmagic.exe	Файл демонстраційної моделі
Презентація	
Презентація_Жукова.pptx	Презентація дипломного проекту