Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія створення гри Weenfaster у жанрі платформер»

Виконав: студент 2 курсу, групи 2КН-23М
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Молдаванов В.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Озеранський В.С.
(прізвище та ініціали)

«05» 12 2024 р.

Опонент: к.т.н., доцент каф. АІІТ
Барабан М.В.
(прізвище та ініціали)

«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н. проф.
Яровий А.А.
«06» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 122 – Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ

Завідувач кафедри КН
д.т.н., проф. Яровий А.А
"11" 10 20 року

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу
студенту
Молдаванову Володимиру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: "Інформаційна технологія створення гри Weenfaster у жанрі платформер".

   Керівник роботи: Озеранський Володимир Сергійович, к.т.н., доц. каф. КН

   затверджені наказом вищого навчального закладу "11" 09 2024 року № 310

2. Строки подання студентом роботи: : 11. 11 2024 року
3. Вихідні дані для роботи:

Мова програмування – об'єктно-орієнтована; Операційна система – Windows 10; Ромір ігрового поля 1920*1080 пікселів. Обсяг оперативної пам'яті – не більше 100 МБ; Обсяг ресурсів процесора – не більше 10%; Кількість гравців - не менше 1 чол.

4. Зміст пояснювальної записки

аналіз існуючих розробок комп'ютерних ігор ; загальна характеристика ігрової індустрії; огляд та порівняльний аналіз ігрових движків; розробка гри на unity; тестування результатів роботи;

5. Перелік графічного матеріалу

UML-діаграми класів (діаграма основного персонажа, діаграма платформ). Архітектурна діаграма (відображення структури рівнів). Схеми сценаріїв тестування та зображення з ігрових сцен для пояснення тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Озеранський В.С. к.т.н., доцент каф. КН		
4	Адлер О.О. к.т.н., доцент каф. ЕПТВМ		

7. Дата видачі завдання

02.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Постановка задачі дослідження	02.09.24	14.09.24	
2	Аналіз предметної області та відомих реалізацій розробки платформерів	15.09.24	25.09.24	
3	Об'єктно орієнтоване програмування створення гри Weenfaster у жанрі платформер	26.09.24	02.10.24	
4	Розробка UML діаграм	03.10.24	18.10.24	
5	Тестування розробленої гри	19.10.24	28.10.24	
6	Оформлення пояснювальної записки, графічного матеріалу	29.10.24	04.11.24	
7	Захист МКР	05.11.24	11.11.24	

Студент _____ Молдаванов В.І
(підпис) (ініціали і прізви

Керівник роботи _____ Озеранський В.
(підпис) (ініціали і прізвище

АНОТАЦІЯ

УДК 004.4:004.42

Молдаванов В.В. Інформаційна технологія створення гри Weenfaster у жанрі платформер. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп’ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 104 стр.

Укр. мовою. Бібліогр.: 20 назв; рис.: 13 ; табл. 18.

Дана магістерська робота присвячена розробці інформаційної технології для створення комп’ютерної гри у жанрі платформер. Проект охоплює аналіз існуючих підходів до створення таких ігор, розробку механізмів гри, взаємодії з платформами, управлінню персонажами та фізичними аспектами гри, а також застосування оптимізаційних технік для досягнення низьких вимог до ресурсів.

Робота включає кілька розділів, які детально розглядають етапи проектування ігор у жанрі платформер, проблеми оптимізації, реалізації механік стрибків та взаємодії з платформами, а також застосування сучасних інформаційних технологій для створення інтерактивного досвіду для гравців. Зокрема, акцентовано увагу на важливості створення ігор з мінімальними вимогами до ресурсів комп’ютера, що є важливим аспектом для широкого доступу гравців.

В рамках роботи висвітлюються перспективи використання інтелектуальних моделей для покращення ігрового процесу, персоналізації досвіду гравця, а також вдосконалення пошуку та адаптації ігор до різних платформ і пристроїв. Висновки і рекомендації цієї роботи можуть бути корисними для подальших досліджень у галузі розробки ігор і допоможуть вдосконалити процеси створення платформерів, роблячи їх більш доступними та привабливими для гравців.

Ключові слова: інформаційні технології, комп'ютерна гра, платформер, розробка ігор, оптимізація, інтелектуальні моделі.

ABSTRACT

Moldavanov V.V. Information Technology for Creating the Platformer Game "Weenfaster." Master's Qualification Thesis in the Specialty 122 "Computer Science," Educational Program "Artificial Intelligence Systems." Vinnytsia: VNTU, 2024. 101 p. In Ukrainian. Bibliography: 20 titles; Figures: 13; Table 18.

This master's thesis is dedicated to the development of information technology for creating a platformer computer game. The project includes an analysis of existing approaches to game design, the development of game mechanics, interactions with platforms, character control, and the physical aspects of the game, as well as the application of optimization techniques to achieve low resource requirements.

The work includes several sections that thoroughly explore the stages of platformer game design, optimization issues, the implementation of jump mechanics, interactions with platforms, and the application of modern information technologies to create an interactive player experience. Special emphasis is placed on creating games with minimal computer resource requirements, which is a crucial aspect for broad player accessibility.

The thesis highlights the potential for using intelligent models to improve gameplay, personalize the player experience, and enhance the adaptability of games to different platforms and devices. The conclusions and recommendations of this work may benefit further research in game development and help refine the processes of platformer creation, making them more accessible and appealing to players.

Keywords: information technology, computer game, platformer, game development, optimization, intelligent models.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Ігри в жанрі платформер продовжують відігравати визначну роль у сучасній ігровій індустрії, демонструючи стабільне зростання популярності та комерційного успіху. Яскраві приклади останніх років, такі як "Hollow Knight", "Celeste" та "Ori and the Will of the Wisps", не лише здобули критичне визнання, але й досягли значних комерційних результатів, що підтверджує стійкий попит на якісні платформери. Особлива цінність жанру полягає в його універсальності - платформери приваблюють широкий спектр гравців завдяки інтуїтивно зрозумілим механікам та поступовому підвищенню складності, що робить їх доступними для новачків і одночасно викликаючими для досвідчених гравців.

В умовах стрімкого розвитку технологій та зростання ринку мобільних ігор, платформери демонструють виняткову адаптивність до нових платформ та способів керування. Сучасні дослідження в галузі геймдизайну підкреслюють особливу роль платформерів у розвитку когнітивних здібностей, просторового мислення та координації. Більше того, зростаюча тенденція до використання ігор в освітніх цілях відкриває нові перспективи для жанру, де навчальні елементи можуть органічно поєднуватися з захоплюючим геймплеєм.

Особливої актуальності набуває дослідження інноваційних підходів до розробки платформерів у контексті зростаючого попиту на інді-ігри та унікальний користувацький досвід. Сучасні гравці все частіше шукають проєкти, які пропонують оригінальні механіки та свіжі ідеї, зберігаючи при цьому знайому основу жанру.

Розробка гри "Weenfaster" у цьому контексті представляє собою актуальне дослідження можливостей інновацій у жанрі платформерів. Проєкт спрямований на створення унікального ігрового досвіду, який поєднує класичні елементи жанру з сучасними технологічними можливостями та дизайнерськими

рішеннями. Особлива увага приділяється впровадженню механік, що сприяють розвитку реакції та стратегічного мислення гравців, що відповідає сучасним тенденціям до створення ігор з освітньою цінністю. В умовах зростаючої конкуренції на ринку відеоігор, такий підхід до розробки може забезпечити проєкту унікальну ринкову позицію та привернути увагу як традиційних прихильників жанру, так і нової аудиторії.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерні науки Вінницького національного технічного університету 122 КІ "Розробка прикладних інтелектуальних інформаційних технологій та систем" та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання досліджень. Мета магістерської кваліфікаційної роботи є підвищення ефективності роботи ігрового рушія "Weenfaster" у жанрі платформер.

Для досягнення поставленої мети потрібно виконати наступні завдання:

- Аналіз сучасних технологій створення ігор у жанрі платформерів;
- Розробка архітектури ігрового рушія «Weenfaster» з розподілом відповідальності між модулями та компонентами;
- Створення UML-діаграм для відображення структури інформаційної технології створення гри "Weenfaster" у жанрі платформер.
- Програмна реалізація інформаційної технології створення гри "Weenfaster" у жанрі платформер;
- Проведення тестування створеної гри відповідно до сформованих сценаріїв тестування.
- Аналіз ефективності гри «Weenfaster» за параметрами продуктивності, графіки, адаптивності та інноваційності.

- Здійснення економічних розрахунків доцільності створення нової розробки.

Об’єктом дослідження є процес створення гри "Weenfaster" у жанрі платформер.

Предмет дослідження – програмні засоби створення гри "Weenfaster" у жанрі платформер.

У магістерській роботі застосовані різні методи дослідження. До них належать методи аналіз наукових джерел та літератури, об'єктно–орієнтоване програмування, статистичний аналіз, графічний дизайн, проектування ігор, проектування на основі існуючих ігрових рушіїв.

Наукова новизна. Наукова новизна полягає у подальшому удосконаленні інформаційної технології гри "Weenfaster" у жанрі платформер, що відрізняється від відомих удомконаленою інформаційною моделлю, яка забезпечує підвищення ефективності роботи ігрового рушія "Weenfaster" у жанрі платформер.

Практичне значення одержаних результатів. Практичне значення полягає в тому, що:

- розроблено алгоритм функціонування інформаційної технології створення гри "Weenfaster" у жанрі платформер;

- здійснено програмну реалізацію інформаційної технології створення гри "Weenfaster" у жанрі платформер.

Достовірність отриманих результатів. Достовірність отриманих результатів забезпечується ретельним тестуванням ігрових механік та оптимізованих алгоритмів на різних платформах і пристроях із різними технічними характеристиками. Під час розробки "Weenfaster" були використані сучасні методи кодування та оптимізації ресурсів, які дозволили досягти стабільної роботи гри навіть на застарілому обладнанні.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати були апробовані на науковотехнічній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [1].

Публікації. За результатами досліджень опубліковано тези доповіді [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ СТВОРЕННЯ ІГОР У ЖАНРІ ПЛАТФОРМЕР

1.1 Аналіз ігрових рушіїв у жанрі платформер

Історія жанру платформерів бере свій початок у ранніх 1980-х роках. Все почалося з простої гри Space Panic, випущеної в 1980 році, яка заклала перші цеглини в фундамент майбутнього жанру. Проте справжнім проривом стала легендарна Donkey Kong від Nintendo, що з'явилася в 1981 році. Саме ця гра вперше представила концепцію стрибків між платформами, яка стала визначальною характеристикою всього жанру.

Наступним значущим етапом розвитку стала поява Super Mario Bros. у 1985 році. Дана гра здійснила справжню революцію, представивши світу концепцію бічного скролінгу та багаторівневого дизайну. Вона встановила багато стандартів, які використовуються в платформерах і до сьогодні. Гравці вперше отримали можливість досліджувати великі рівні, шукати секрети та бонуси, взаємодіяти з різноманітними ворогами.

Механіка платформерів будується навколо руху персонажа в двовимірному або тривимірному просторі. Фундаментальним елементом є точний контроль над персонажем. Розробники приділяють особливу увагу відчуттю ваги персонажа, його прискоренню та гальмуванню. Кожен рух має бути точним та передбачуваним, щоб гравець міг впевнено планувати свої дії.

Стрибки в платформерах заслуговують окремої уваги, адже вони є серцем геймплею. Хороший платформер повинен мати ідеально налаштовану фізику стрибків. Висота, дальність, контроль у повітрі – все це ретельно балансується для створення приємного ігрового процесу. Важливим є відчуття контролю над

персонажем під час стрибка, можливість коригувати траєкторію та точність приземлення.

Дизайн рівнів у платформерах є складним мистецтвом, що вимагає глибокого розуміння ігрової механіки та психології гравця. Кожен рівень починається з простого введення, яке знайомить гравця з новими елементами. Поступово складність зростає, представляючи все більш складні випробування. При цьому важливо зберігати баланс між викликом та задоволенням від гри.

Візуальний дизайн платформерів має величезне значення. Графічне оформлення повинно чітко комунікувати з гравцем, показуючи, які елементи є платформами, що можна, а що не можна чіпати. Колірна схема, контраст, форми – все це працює разом, щоб створити зрозумілий та привабливий ігровий простір. Анімації персонажа мають бути плавними та інформативними, допомагаючи гравцю розуміти стан гри.

Звуковий дизайн у платформерах виконує не менш важливу роль. Кожна дія супроводжується відповідним звуковим ефектом, який надає грі додаткової глибини та забезпечує важливий зворотний зв'язок. Стрибки, приземлення, збір предметів, зіткнення з ворогами – все це має свій унікальний звуковий супровід. Музика також відіграє важливу роль, задаючи темп гри та створюючи потрібну атмосферу.

Прогрес в платформерах будується на поступовому ускладненні випробувань. Початкові рівні зазвичай прості та дозволяють гравцю освоїтися з керуванням. З кожним новим рівнем додаються нові елементи та механіки, які вимагають все більшої майстерності. При цьому важливо, щоб складність зростала плавно, не створюючи різких стрибків, які можуть відштовхнути гравця.

Сучасні платформери часто включають елементи інших жанрів. Це може бути система прокачки персонажа, елементи головоломок, бойові сцени або навіть елементи рольових ігор. Такий міжжанровий підхід дозволяє створювати

більш різноманітний та глибокий ігровий досвід, зберігаючи при цьому core-механіки платформера.

Технологічний прогрес дозволив платформерам еволюціонувати від простих двовимірних ігор до складних тривимірних світів. Сучасні платформери можуть похвалитися реалістичною фізикою, складними системами частинок, динамічним освітленням та іншими технічними досягненнями. Проте важливо, щоб технологічні інновації не заважали основному геймплею, а доповнювали його.

Особливу увагу в сучасних платформерах приділяють системі досягнень та колекціонування. Додаткові випробування, секретні зони, колекційні предмети – все це збільшує реіграбельність та мотивує гравців досліджувати кожен куточок рівня. При цьому важливо, щоб додатковий контент був цікавим та винагороджував гравця за витрачені зусилля.

Розробка платформерів вимагає глибокого розуміння ігрової механіки, фізики руху, психології гравця та принципів геймдизайну. Кожен елемент гри має працювати злагоджено з іншими, створюючи цілісний та захоплюючий ігровий досвід. Успішний платформер – це результат ретельного балансування всіх цих елементів.

Майбутнє жанру платформерів виглядає багатообіцяючим. Нові технології, такі як віртуальна реальність, відкривають нові можливості для інновацій. Проте основні принципи жанру – точне керування, продуманий дизайн рівнів та поступова прогресія складності – залишаються незмінними. Платформери продовжують еволюціонувати, зберігаючи при цьому свою унікальну ідентичність та привабливість для гравців.

Сучасний ринок відеоігор пропонує величезне різноманіття платформерів – від простих мобільних ігор до складних консольних проектів. Кожен може знайти щось для себе, незалежно від рівня майстерності та уподобань. Жанр продовжує розвиватися, експериментувати з новими ідеями та механіками,

залишаючись при цьому одним з найпопулярніших та найулюбленіших серед гравців усього світу.

1.2 Проблеми проектування сучасних платформерів

Проектування сучасних платформерів є складним процесом, що включає різноманітні технічні, художні та концептуальні аспекти. Одна з основних проблем проектування — це баланс між інноваціями і традиційними елементами жанру. З одного боку, гравці очікують певних знайомих характеристик, як-от чітке управління, плавний рух персонажів і захоплюючий дизайн рівнів. З іншого боку, вони прагнуть нових підходів, унікальних механік, незвичних візуальних стилів та ігрових можливостей. Тому розробники повинні обережно експериментувати, щоб створити гру, яка не лише відповідає сучасним вимогам, а й є унікальною та захопливою.

Іншою важливою проблемою є оптимізація продуктивності та вимог до апаратного забезпечення. Сучасні платформи, зокрема мобільні пристрої, обмежені в ресурсах, що вимагає від розробників особливої уваги до енергоефективності та оптимізації. Це передбачає зменшення використання пам'яті, оптимізацію роботи графічних ресурсів, зменшення складності фізики об'єктів тощо. Створення гри, яка працюватиме швидко і стабільно на різних пристроях — складне завдання, адже це може потребувати компромісів між графічною якістю та швидкістю.

Наступним викликом є проектування рівнів, яке вимагає уваги до деталей, логічної послідовності й розмаїття ігрових сценаріїв. Платформери зазвичай включають багато рівнів, кожен з яких повинен бути унікальним, цікавим і поступово збільшувати складність. Важливо забезпечити такий рівень викликів, щоб гравці залишалися зацікавленими, але не відчували розчарування через надмірну складність. Балансування складності потребує ретельного тестування і

зворотного зв'язку з гравцями, що іноді може бути трудомістким і ресурсозатратним процесом.

Візуальна складова є не менш важливою частиною проектування. Сучасні гравці очікують високоякісної графіки, яка доповнює атмосферу ігрового світу. Розробники часто стикаються з проблемою вибору між реалістичною графікою та більш стилізованим підходом, що дозволяє створити унікальний візуальний стиль. Крім того, розробка графічного контенту для платформера, зокрема персонажів, ворогів та оточення, вимагає злагодженої роботи художників, які мають врахувати технічні вимоги до ресурсів.

Ще одним важливим аспектом є звук і музика, які грають значну роль у створенні атмосфери та настрою гри. Багато платформерів відомі своєю музикою, яка не лише підсилює геймплей, а й дозволяє гравцям емоційно зануритися у гру. Звуковий дизайн повинен враховувати всі аспекти ігрового процесу — від фонового звуку до звуків, що супроводжують дії персонажів, таких як стрибки або удари. Звукові ефекти мають бути чіткими, не відволікати від основного геймплею та доповнювати загальний візуальний стиль.

Також розробка сучасних платформерів потребує розширених можливостей управління для гравців. У залежності від платформи, де запускається гра, варто враховувати різні способи введення — клавіатуру, мишу, контролер або сенсорний екран. Усе це має бути налаштовано так, щоб гравець відчував контроль над персонажем без зайвих труднощів. Управління має бути інтуїтивно зрозумілим і комфортним, інакше це може негативно вплинути на враження від гри.

Тестування і забезпечення якості є значною частиною проектування, оскільки платформери, через свою природу, часто містять багато механік, які потребують ретельної перевірки. Потрібно усунути будь-які баги, що можуть вплинути на геймплей, зокрема проблеми з колізією об'єктів, фізикою, поведінкою ворогів або керуванням персонажем. Тестування також повинно

охоплювати різні аспекти складності та забезпечення балансу, щоб гравець отримував задоволення від гри.

Вирішення основних проблем проектування сучасних платформерів вимагає використання сучасних технологій, оптимізаційних методів та творчого підходу. Один із ключових підходів для досягнення балансу між інноваціями та традиціями полягає в інтеграції класичних ігрових механік з новими елементами, які можуть збагатити досвід гравця. Наприклад, можна зберегти знайомі елементи, такі як стрибки та збирання бонусів, але додати нові механіки або інтерфейс для змінного рівня складності, що дозволить охопити ширшу аудиторію. Такі інновації варто впроваджувати поступово, щоб уникнути надмірної зміни основного досвіду, що може відштовхнути шанувальників жанру.

Щодо оптимізації продуктивності та вимог до апаратного забезпечення, одним з ефективних шляхів є використання алгоритмів оптимізації ресурсів і кодування, що знижують навантаження на систему. Наприклад, для зниження споживання пам'яті можна використовувати текстури низької роздільної здатності, які відображаються на пристроях із меншими екранами, або динамічно підвантажувати ресурси лише у потрібний момент. Також корисно використовувати спеціальні методи зменшення складності фізичних обчислень, що знижує вимоги до процесора. Це забезпечує стабільну роботу гри на різних пристроях без компромісів якості геймплею.

Проектування рівнів можна оптимізувати завдяки використанню процедурної генерації. Це дозволяє автоматично створювати унікальні рівні, які відповідають певним критеріям складності, при цьому зберігаючи логічну послідовність та розмаїття. Процедурно згенеровані рівні можуть створювати непередбачувані сценарії, які зацікавлюють гравців, а також дозволяють розробникам зекономити час і ресурси на ручне створення кожного рівня. Крім

того, можна впровадити механізми для регулювання складності рівнів залежно від успішності гравця, щоб підтримувати оптимальний рівень виклику.

Щоб забезпечити високу якість графіки без шкоди для продуктивності, розробники можуть використовувати техніку "спрайт-шейдингу" або стилізовану 2D-графіку, що не вимагає значних ресурсів, але дозволяє створити унікальний візуальний стиль. Крім того, розробка персонажів і об’єктів з мінімальною кількістю полігонів дозволяє досягти високої продуктивності, зберігаючи деталізацію. Це допоможе створити якісну графіку, яка буде одночасно привабливою і ресурсозберігаючою, що є важливим для платформерів.

Для забезпечення якісного звукового супроводу розробники можуть використовувати бібліотеки з уже оптимізованими звуковими ефектами та музикою, що забезпечують високу якість звучання з мінімальним навантаженням на ресурси. Важливо також застосовувати звукові ефекти лише в потрібні моменти, знижуючи надмірне навантаження на систему. Крім того, використання циклічних звукових тем зберігає атмосферність і запобігає надмірному використанню пам’яті та місця зберігання.

Покращення управління можна досягти шляхом адаптації до різних пристроїв і введення регульованих параметрів, щоб кожен гравець міг налаштувати управління під себе. Використання зручних елементів управління, які налаштовуються, дозволить розробникам створити інтуїтивно зрозумілий інтерфейс. Наприклад, для мобільних платформ варто передбачити сенсорне управління з можливістю індивідуального налаштування жестів, що підвищить зручність користування і дозволить охопити більше користувачів.

На завершення, тестування та забезпечення якості можна поліпшити шляхом використання автоматизованих систем для пошуку багів, що дозволить швидше знаходити та усувати проблеми. Зокрема, можна застосовувати спеціальні інструменти для автоматизованого тестування колізії об'єктів, продуктивності та інших аспектів. Крім того, корисно залучати спільноту бета-

тестерів для збору зворотного зв'язку, що дозволить вдосконалити ігровий досвід на основі реальних відгуків і швидше виявити помилки, які не завжди помітні під час автоматизованого тестування.

1.3 Порівняльний аналіз ігор-аналогів у жанрі платформер

Оглянемо деякі відомі ігри-платформери та проаналізуємо їх сильні і слабкі сторони.

Один з найвідоміших платформерів усіх часів, Super Mario Bros., розроблений компанією Nintendo, вперше з'явився у 1985 році і заклав основи жанру платформерів. Гравець керує Маріо (або, у багатокористувацькому режимі, його братом Луїджі), які подорожують через серію рівнів у Королівстві Грибів, щоб врятувати принцесу Піч. Гра включає низку класичних елементів: збір монет для отримання додаткових життів, подолання ворогів, таких як грибці Goomba і черепахи Koopa, а також пошук схованих предметів і бонусів, таких як гриб, який збільшує Маріо. Super Mario Bros. відома своєю чіткою механікою керування, збалансованою складністю та різноманітними рівнями, від підземель до повітряних замків.

Sonic the Hedgehog — класична серія платформерів від Sega, яка вперше з'явилася у 1991 році. В її центрі — їжачок Сонік, який має надзвичайну швидкість і бореться з доктором Роботніком (також відомим як доктор Еґґман). Гра побудована навколо високошвидкісного геймплею, де гравець керує Соніком, прискорюється через рівні, стрибає через перешкоди, збирає золоті кільця, які захищають його від ушкоджень, і бореться з різноманітними механічними ворогами. Sonic the Hedgehog характеризується унікальними рівнями, в яких можна обирати різні шляхи, що дозволяє використовувати швидкість Соніка. Кільця грають ключову роль у виживанні персонажа — якщо Сонік має кільця, він виживає після удару, але втрачає їх.

Rayman — серія яскравих платформерів від Ubisoft, що вперше з'явилася в 1995 році. Гравець керує Реймaном — героєм без рук і ніг, який має магічні здібності та унікальні навички, такі як політ, бійка і використання ударів. Метою гри є подорож через красиві фантастичні світи, де Рейман збирає лумси (маленькі літаючі істоти) і рятує своїх друзів від ворогів, таких як злий маг Містер Дарк. Гра відома своєю графічною стилістикою, різноманітними персонажами та гумором. Rayman має як простіші, так і складні рівні з головоломками, де гравцеві потрібно використовувати різні здібності Реймана для подолання перешкод.

Donkey Kong Country — серія платформерів від Nintendo, вперше випущена у 1994 році. Гравець керує Донкі Конгом та його товаришами, які подорожують джунглями, печерами та іншими локаціями, щоб збирати банани і боротися з ворогами, такими як крокодилоподібні істоти — кримлінги. Основна сюжетна лінія обертається навколо порятунку викрадених бананів або боротьби за збереження своєї домівки. Donkey Kong Country відрізняється красивою графікою (завдяки використанню попередньо рендерованих 3D-моделей), високою складністю ігрових рівнів і кооперативним режимом, де два гравці можуть допомагати один одному. Кожен персонаж має унікальні навички, що допомагають долати рівні і розв’язувати головоломки.

Ori and the Blind Forest — платформер у стилі метроїдванія, випущений у 2015 році компанією Moon Studios. Гра зосереджена на Орі — маленькому духу, який шукає спосіб врятувати ліс Нібель, що занепадає. Гравець керує Орі, використовуючи його спритність, здатність лазити по стінах і літати на короткі дистанції, щоб досліджувати великий відкритий світ, заповнений головоломками і ворогами. Орі розвиває свої здібності в міру проходження гри, що дозволяє повертатися до попередніх рівнів для відкриття нових локацій і секретів. Ori and the Blind Forest отримала визнання за емоційну історію, високу якість графіки та музику, які створюють неповторну атмосферу.

Наведемо порівняльний аналіз даних ігор у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз відомих ігор-платформерів

Гра	**Переваги**	**Недоліки**
Super Mario Bros.	Класична механіка платформера; просте, інтуїтивне управління; доступність для широкої аудиторії; різноманітні рівні і перешкоди.	Деякі гравці можуть вважати гру застарілою через просту графіку і лінійний геймплей.
Sonic the Hedgehog	Високошвидкісний геймплей, що робить гру динамічною; яскрава графіка; різноманітні шляхи проходження рівнів.	Висока швидкість може ускладнювати контроль; деякі рівні не дають достатньо часу для реакції на перешкоди, що робить гру складнішою.
Rayman	Унікальний ігровий стиль та гумор; приваблива графіка; різноманітні здібності Реймана, що дозволяють проходити гру в різний спосіб.	Може бути складною для нових гравців через нестандартну механіку; деякі рівні можуть бути занадто складними і вимагають точного виконання дій.
Donkey Kong Country	Гарна графіка для свого часу; можливість кооперативної гри; цікавий та різноманітний дизайн рівнів.	Високий рівень складності може бути проблемою для новачків; гра може бути важкою через обмежену кількість життів і рідкісні контрольні точки на рівнях.

Продовженяя таблиці 1.1

Гра	Переваги	Недоліки
Ori and the Blind Forest	Висока якість графіки і музики; емоційна історія, яка залучає гравця; унікальні здібності Орі для проходження складних головоломок.	Деякі головоломки можуть бути складними для новачків; гра має обмежену кількість підказок, що може ускладнити проходження певних рівнів.

1.4 Висновки до розділу 1

У розділі 1 було розглянуто загальні аспекти жанру платформерів, їхні особливості та популярні приклади, які мали значний вплив на розвиток ігрової індустрії. Платформери займають особливе місце у світі відеоігор завдяки простоті основної механіки, яка поєднується зі значним потенціалом для творчих експериментів та інновацій. Класичні ігри, такі як *Super Mario Bros.*, *Sonic the Hedgehog*, *Rayman*, *Donkey Kong Country* і *Ori and the Blind Forest*, слугують чудовими прикладами для аналізу, оскільки кожна з них зробила свій внесок в жанр, запропонувавши унікальні механіки, сюжетні елементи та стилістичні рішення. Це різноманіття показує, що платформери здатні адаптуватися до вимог часу, інтегруючи як традиційні, так і нові елементи для забезпечення захопливого досвіду.

Розглянуті ігри ілюструють ключові проблеми та переваги платформерів, що стосуються керування, продуктивності, дизайну рівнів, графіки та звукового супроводу. Також було акцентовано на значенні інноваційних підходів для подолання цих проблем. Загалом, платформери продовжують залишатися актуальними завдяки можливості створення як складних, так і більш доступних

ігор, що залучають широку аудиторію. Усі ці особливості визначають високий потенціал жанру для подальших досліджень та удосконалень, які можуть сприяти розробці нових цікавих ігор із оригінальним підходом до класичних механік.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ГРИ WEENFASTER У ЖАНРІ ПЛАТФОРМЕР

2.1 Розробка UML-діаграм інформаційної технології гри «Weenfaster»

UML (Unified Modeling Language) – це стандартний засіб візуального моделювання, що широко застосовується в розробці програмного забезпечення, зокрема ігор. Він забезпечує розробникам можливість створити графічні моделі системи, що полегшують розуміння архітектури та взаємодії між її елементами. Проектування гри за допомогою UML є ефективним засобом структурування та оптимізації процесу розробки, що допомагає створити чітке уявлення про функціональні можливості гри, її взаємозв'язки та потенційні проблеми ще до початку активного кодування.

UML дозволяє розробникам створити модель гри у вигляді набору діаграм, кожна з яких описує певний аспект функціонування гри. Класова діаграма є одним із найважливіших типів UML-діаграм при проектуванні гри. Вона представляє класи, які будуть використовуватися в грі, а також їхні атрибути та методи. Для гри у жанрі платформер важливо виділити основні класи, такі як Player, Enemy, Level, Obstacle, Item тощо. Клас Player може містити атрибути, що описують поточний стан гравця (наприклад, рівень здоров’я, кількість зібраних монет), а також методи для здійснення основних дій, таких як стрибки, переміщення, атаки. Класи Enemy та Obstacle відповідають за ворогів та перешкоди, з якими гравець може взаємодіяти або уникати.

Ще одна важлива діаграма UML – діаграма послідовності, яка допомагає візуалізувати порядок виконання дій та взаємодію між об’єктами у часі. Наприклад, у платформері можна створити діаграму послідовності, яка показує, як об’єкти Player, Enemy та Environment взаємодіють під час процесу бою. Ця діаграма демонструє, як ігровий персонаж може атакувати ворога, отримувати

від нього пошкодження або взаємодіяти з об’єктами середовища, такими як платформи або пастки. Послідовність виконання подібних дій є критично важливою для забезпечення правильної логіки гри.

Діаграма активностей UML також може бути корисною у проектуванні гри. Цей тип діаграми дозволяє показати потік виконання певного процесу, що особливо корисно для моделювання ігрових механік, які залежать від складних умов або багатоступеневих процесів. Наприклад, діаграма активностей може ілюструвати алгоритм переходу між рівнями: коли гравець завершує один рівень, гра перевіряє виконані завдання, оновлює статус гравця, зберігає прогрес та переміщує персонажа на наступний рівень.

Діаграма станів UML є ще одним важливим інструментом для проектування ігор, який особливо корисний для відображення станів, у яких може перебувати об’єкт під час гри. У випадку платформера, гравець може мати кілька станів: Stand (стояння), Jump (стрибок), Run (біг), Attack (атака), Damage (отримання ушкоджень) та інші. Діаграма станів дозволяє чітко показати, у яких умовах об’єкт може переходити з одного стану в інший, що важливо для створення логічно зв'язаної системи дій персонажа.

Діаграми компонентів UML дають змогу створити модель фізичної структури системи. Вони показують розподіл програмних модулів та зв’язки між ними, що допомагає визначити, як різні частини гри взаємодіють на фізичному рівні. У випадку гри можна створити діаграму компонентів, яка відображає модулі керування персонажем, керування рівнями, фізики, графіки, звукових ефектів та інші, демонструючи їхню взаємодію між собою та з апаратними компонентами системи.

Діаграма послідовності гри «Weenfaster» наведена на рис. 2.1.

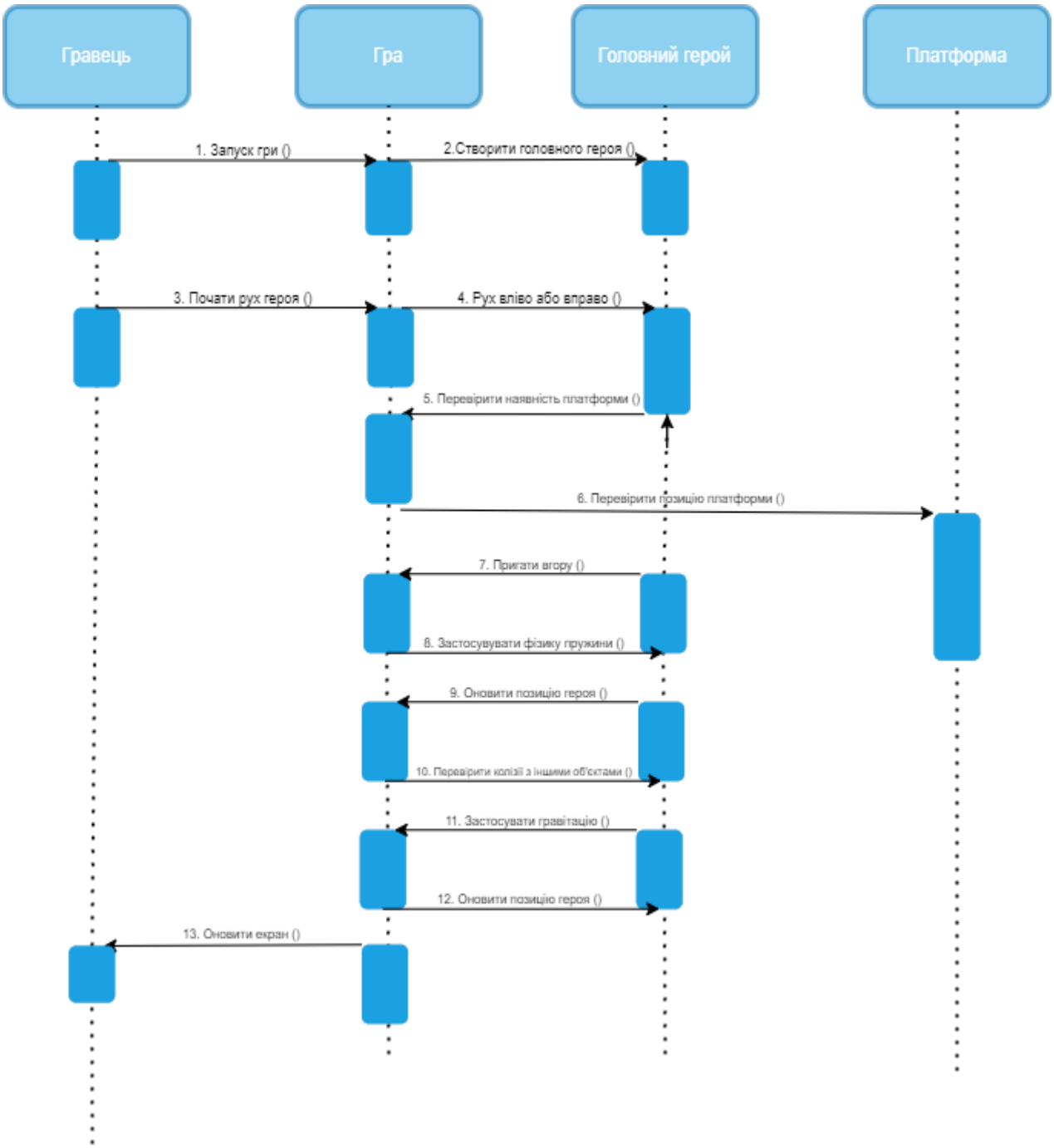


Рисунок 2.1 – Діаграма послідовності для гри «Weenfaster»

На діаграмі послідовності зображено послідовність дій, які відбуваються під час гри у платформер. Вона ілюструє взаємодію між різними об’єктами,

такими як *Гравець*, *Гра*, *Головний герой* і *Платформа*, показуючи, як ці об’єкти обмінюються повідомленнями та виконують дії для досягнення ігрових цілей.

Процес починається з того, що *Гравець* запускає гру (крок 1), і система створює об’єкт *Головний герой* (крок 2), тобто персонажа, якого гравець буде контролювати. Далі, гравець починає рух героя (крок 3), що активує рух персонажа ліворуч або праворуч (крок 4), залежно від команди, яку обирає гравець.

Гра перевіряє, чи є в межах героя платформи (крок 5), і за потреби перевіряє їхні розміри та розташування (крок 6), щоб визначити можливість взаємодії. Це важливо для того, щоб герой міг пересуватися на платформі або між платформами, не порушуючи правил гри.

Якщо є платформа, з якою герой може взаємодіяти, система виконує перевірку взаємодії та визначає, чи може герой здійснити стрибок (крок 7). При цьому гра враховує фізику стрибка, яку потім застосовує для правильного розрахунку руху (крок 8).

Після цього оновлюється позиція головного героя (крок 9), враховуючи його рух. На даному етапі система також перевіряє можливі колізії з іншими об'єктами (крок 10), такими як вороги, предмети або пастки, які можуть перебувати на шляху героя. Це забезпечує реалізм ігрового процесу та взаємодію героя з ігровим середовищем.

Далі застосовується гравітація (крок 11), яка впливає на позицію героя у вертикальній площині. Це дозволяє забезпечити коректне падіння героя, коли він не стоїть на платформі, і додає реалістичності в його рухи. Після застосування гравітації знову оновлюється позиція героя (крок 12), що дозволяє врахувати всі зміни в його стані та забезпечити динаміку гри.

Завершальним етапом є оновлення екрану (крок 13), яке показує гравцеві актуальний стан гри, враховуючи всі дії, які відбулися під час попередніх кроків.

Тепер створимо діаграму класів системи (рис. 2.2).

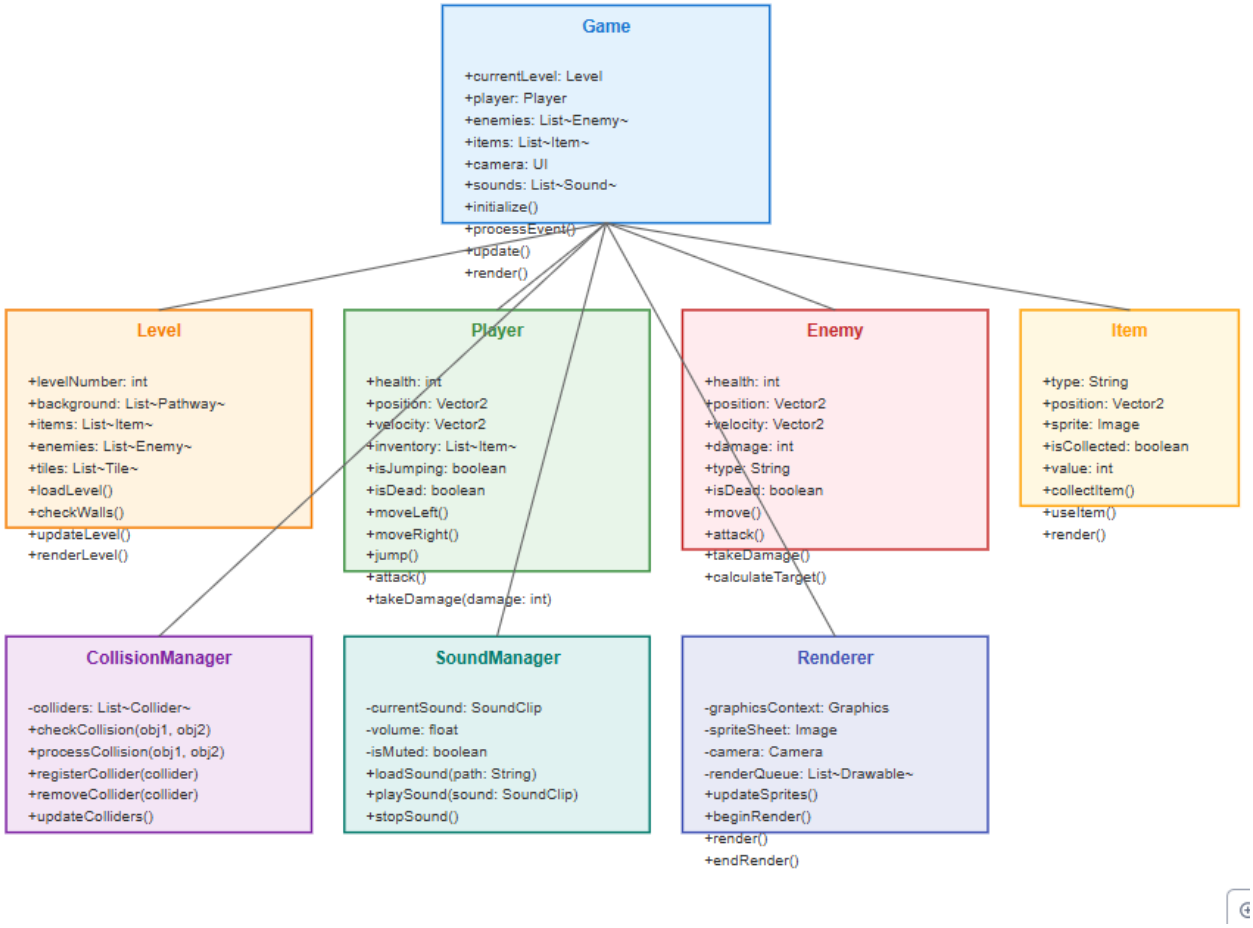


Рисунок 2.2 – Діаграма класів для гри «Weenfaster»

Розглянемо детально кожен клас гри Weenfaster, його призначення, властивості, методи та взаємозв'язки з іншими класами.

КЛАС GAME

Це центральний клас гри, який керує всією ігровою логікою та станом.

Властивості:

currentLevel зберігає поточний рівень гри, дозволяючи контролювати його стан та оновлення.

player містить об'єкт гравця, через який здійснюється основне керування персонажем.

enemies зберігає список всіх ворогів на поточному рівні.

items містить колекцію всіх предметів у грі.

score відслідковує рахунок гравця протягом гри.

isRunning визначає, чи гра активна чи призупинена.

Методи класу Game забезпечують повний контроль над ігровим процесом:

startGame запускає гру, ініціалізуючи всі необхідні компоненти та ресурси.

pauseGame призупиняє гру, зберігаючи її поточний стан.

update оновлює стан всіх ігрових об'єктів кожен кадр.

render відповідає за відображення всіх візуальних елементів.

endGame завершує гру, звільняючи ресурси та зберігаючи результати.

КЛАС LEVEL

Відповідає за структуру та логіку окремого рівня гри.

Властивості:

levelNumber унікальний ідентифікатор рівня. platforms містить всі платформи на рівні.

enemies зберігає ворогів, специфічних для даного рівня.

items включає всі предмети, розміщені на рівні.

background зберігає фонове зображення рівня.

Методи рівня забезпечують його функціонування:

loadLevel завантажує всі необхідні ресурси та об'єкти рівня.

unloadLevel вивантажує ресурси при переході на інший рівень.

updateLevel оновлює стан всіх об'єктів на рівні.

renderLevel відображає всі елементи рівня на екрані.

КЛАС PLAYER

Керує персонажем гравця та його взаємодією з ігровим світом.

Властивості: health відслідковує здоров'я персонажа.

position визначає розташування гравця у просторі.

velocity контролює швидкість та напрямок руху.

sprite містить візуальне представлення персонажа. inventory зберігає зібрані предмети.

Методи персонажа визначають його можливості:

moveLeft переміщує персонажа вліво.

moveRight переміщує персонажа вправо.

jump реалізує механіку стрибка.

attack дозволяє атакувати ворогів.

takeDamage обробляє отримання ушкоджень.

collectItem додає предмети до інвентарю.

КЛАС ENEMY

Відповідає за поведінку ворожих персонажів.

Властивості: health визначає кількість здоров'я ворога.

position зберігає координати ворога.

velocity контролює рух ворога.

sprite містить зображення ворога.

damage визначає силу атаки ворога.

Методи ворога визначають його поведінку:

move керує переміщенням ворога.

attack реалізує атаку на гравця.

takeDamage обробляє отримання ушкоджень.

die визначає поведінку при знищенні ворога.

КЛАС PLATFORM

Реалізує платформи, по яких переміщується гравець.

Властивості: position визначає розташування платформи.

size встановлює розміри платформи.

sprite містить візуальне представлення платформи.

Метод render відповідає за відображення платформи на екрані.

КЛАС ITEM Керує ігровими предметами та бонусами.

Властивості:

type визначає тип предмета.

position зберігає координати предмета.

sprite містить зображення предмета.

Метод collect обробляє підбирання предмета гравцем.

КЛАС COLLISIONMANAGER

Відповідає за обробку зіткнень між об'єктами.

Властивості: colliders містить список всіх об'єктів, що можуть стикатися.

Методи менеджера колізій:

checkCollision перевіряє наявність зіткнення між об'єктами.

resolveCollision обробляє наслідки зіткнення.

КЛАС INPUTHANDLER

Обробляє введення від користувача.

Властивості: currentInput зберігає поточний стан введення.

Методи обробки введення:

getInput отримує дані від пристроїв введення.

processInput обробляє отримані команди.

КЛАС AUDIOMANAGER

Керує звуковим супроводом гри.

Властивості:

backgroundMusic зберігає поточну фонову музику.

soundEffects містить колекцію звукових ефектів.

Методи аудіо менеджера:

playMusic відтворює музику.

stopMusic зупиняє відтворення музики.

playSound відтворює звукові ефекти.

КЛАС RENDERER Відповідає за візуалізацію гри.

Властивості:

graphicsContext забезпечує доступ до графічного контексту.

Методи рендерера:

drawSprite малює спрайти на екрані.

clearScreen очищує екран.

updateScreen оновлює відображення.

ВЗАЄМОЗВ'ЯЗКИ МІЖ КЛАСАМИ

Game є центральним класом, який координує роботу всіх інших компонентів. Він безпосередньо взаємодіє з Level, Player, Enemy, Item, CollisionManager, InputHandler, AudioManager та Renderer.

Level управляє колекціями Platform, Enemy та Item, забезпечуючи їх правильне розташування та взаємодію.

Player взаємодіє з Enemy через систему атак та отримання ушкоджень, з Item через механіку збору предметів, та з Platform через переміщення по них.

CollisionManager забезпечує коректну фізичну взаємодію між всіма об'єктами гри, перевіряючи та обробляючи зіткнення.

InputHandler передає команди від гравця до об'єкта Player, забезпечуючи керування персонажем.

AudioManager взаємодіє з усіма класами, які потребують звукового супроводу своїх дій.

Renderer відповідає за відображення всіх візуальних елементів гри, працюючи з властивостями sprite кожного класу.

Така архітектура забезпечує модульність та масштабованість гри, дозволяючи легко додавати нові функції та модифікувати існуючі компоненти без порушення загальної структури проекту.

2.2 Проектування архітектури інформаційної технології «Weenfaster»

Розглянемо детальну архітектуру інформаційної технології гри Weenfaster (рис. 2.3).

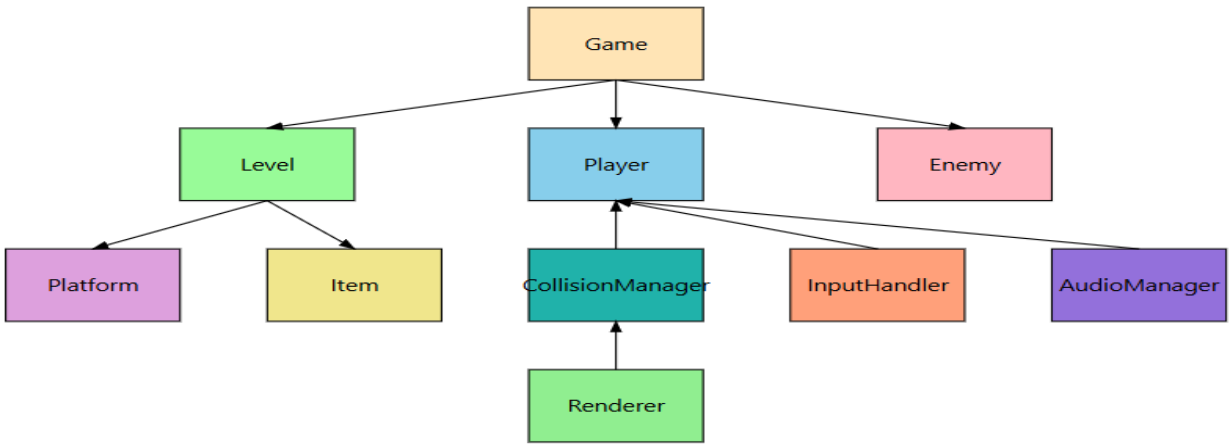


Рисунок 2.3 – Архітектура інформаційної технології гри Weenfaster

1. ЦЕНТРАЛЬНИЙ КОМПОНЕНТ - КЛАС GAME

Це ядро всієї системи, яке виконує роль координатора між усіма компонентами гри.

Основні властивості:

- currentLevel: зберігає інформацію про поточний рівень
- player: містить об'єкт гравця
- enemies: колекція активних ворогів
- items: список доступних предметів
- score: система підрахунку очок
- isRunning: статус активності гри

Ключові методи:

- startGame(): ініціалізація гри
- pauseGame(): призупинення ігрового процесу

- update(): оновлення стану всіх об'єктів
- render(): відображення графічних елементів
- endGame(): завершення гри

2. КОМПОНЕНТ РІВНІВ - КЛАС LEVEL

Відповідає за структуру та логіку кожного окремого рівня.

Властивості:

- levelNumber: ідентифікатор рівня
- platforms: набір платформ
- enemies: вороги на рівні
- items: предмети для збору
- background: фонове оформлення

Методи:

- loadLevel(): завантаження ресурсів рівня
- unloadLevel(): вивантаження ресурсів
- updateLevel(): оновлення стану об'єктів
- renderLevel(): відображення елементів

3. КОМПОНЕНТ ГРАВЦЯ - КЛАС PLAYER

Керує головним персонажем гри.

Властивості:

- health: рівень здоров'я
- position: координати розташування
- velocity: швидкість та напрямок
- sprite: візуальне представлення
- inventory: інвентар предметів

Методи:

- moveLeft(): рух вліво
- moveRight(): рух вправо
- jump(): виконання стрибка

- attack(): здійснення атаки
- takeDamage(): обробка пошкоджень
- collectItem(): збір предметів

4. КОМПОНЕНТ ВОРОГІВ - КЛАС ENEMY

Керує поведінкою ворожих персонажів.

Властивості:

- health: рівень здоров'я
- position: розташування
- velocity: параметри руху
- sprite: зображення
- damage: сила атаки

Методи:

- move(): переміщення
- attack(): атака гравця
- takeDamage(): отримання пошкоджень
- die(): обробка знищення

5. КОМПОНЕНТ ПЛАТФОРМ - КЛАС PLATFORM

Забезпечує створення та функціонування платформ.

Властивості:

- position: розташування
- size: розміри
- sprite: візуальне представлення

Основний метод:

- render(): відображення платформи

6. КОМПОНЕНТ ПРЕДМЕТІВ - КЛАС ITEM

Керує ігровими бонусами та предметами.

Властивості:

- type: тип предмета

- position: розташування
- sprite: зображення

Метод:

- collect(): обробка підбирання

7. МЕНЕДЖЕР КОЛІЗІЙ - КЛАС COLLISIONMANAGER

Відповідає за обробку зіткнень.

Властивості:

- colliders: список об'єктів для перевірки

Методи:

- checkCollision(): перевірка зіткнень
- resolveCollision(): обробка наслідків

8. ОБРОБНИК ВВЕДЕННЯ - КЛАС INPUTHANDLER

Керує взаємодією з користувачем.

Властивості:

- currentInput: поточний стан введення

Методи:

- getInput(): отримання команд
- processInput(): обробка введення

9. АУДІО МЕНЕДЖЕР - КЛАС AUDIOMANAGER

Керує звуковим супроводом.

Властивості:

- backgroundMusic: фонова музика
- soundEffects: звукові ефекти

Методи:

- playMusic(): відтворення музики
- stopMusic(): зупинка відтворення
- playSound(): програвання ефектів

10. СИСТЕМА РЕНДЕРИНГУ - КЛАС RENDERER

Забезпечує візуалізацію гри.

Властивості:

- graphicsContext: графічний контекст

Методи:

- drawSprite(): малювання спрайтів
- clearScreen(): очищення екрану
- updateScreen(): оновлення відображення

ВЗАЄМОЗВ'ЯЗКИ МІЖ КОМПОНЕНТАМИ:

1. Вертикальні зв'язки:

- Game контролює всі інші компоненти
- Level керує Platform та Item
- Player взаємодіє з Enemy та Item

2. Горизонтальні зв'язки:

- CollisionManager взаємодіє з усіма ігровими об'єктами
- InputHandler передає команди до Player
- AudioManager обслуговує всі компоненти
- Renderer відображає всі візуальні елементи

Архітектура інформаційної технології гри Weenfaster має низку суттєвих переваг, які забезпечують ефективну розробку та функціонування гри. Перш за все, слід відзначити високу модульність системи, яка дозволяє розробляти та модифікувати окремі компоненти незалежно один від одного. Кожен модуль має чітко визначені обов'язки та інтерфейси взаємодії, що значно спрощує процес розробки та подальшої підтримки проекту. Така модульна структура також забезпечує можливість легкого масштабування системи, додавання нових функцій та модифікації існуючих без необхідності значних змін в інших частинах коду.

Важливою перевагою є висока гнучкість архітектури, яка дозволяє адаптувати гру під різні платформи та вимоги. Система побудована таким чином,

що окремі компоненти можуть бути замінені або модифіковані без впливу на загальну функціональність. Це особливо важливо при необхідності оптимізації гри під різні пристрої або при впровадженні нових технологій.

Архітектура забезпечує ефективне управління ресурсами системи. Завдяки чіткому розподілу відповідальності між компонентами, оптимізовано використання пам'яті та обчислювальних ресурсів. Система керування ресурсами дозволяє завантажувати та вивантажувати об'єкти за необхідності, що покращує продуктивність гри та зменшує навантаження на систему.

Особливу увагу варто приділити зручності розробки та тестування. Модульна структура дозволяє різним командам розробників працювати над окремими компонентами паралельно, не створюючи конфліктів. Кожен модуль може бути протестований окремо, що значно спрощує процес виявлення та виправлення помилок. Також архітектура передбачає можливість легкого розширення функціоналу та додавання нових ігрових механік без необхідності значної перебудови існуючого коду.

Система демонструє високу надійність та стабільність роботи завдяки чіткому розподілу відповідальності між компонентами та продуманій системі обробки помилок. Кожен компонент має власні механізми обробки виключних ситуацій, що запобігає поширенню помилок на всю систему. Це забезпечує стабільну роботу гри навіть при виникненні проблем в окремих модулях.

Архітектура також забезпечує високу продуктивність завдяки оптимізованій взаємодії між компонентами. Система побудована таким чином, що мінімізує накладні витрати на комунікацію між модулями, забезпечуючи швидку обробку ігрових подій та плавний геймплей. Використання ефективних алгоритмів та оптимізованих структур даних дозволяє досягти високої продуктивності навіть при обробці складних ігрових ситуацій.

Важливою перевагою є масштабованість архітектури, яка дозволяє легко розширювати функціонал гри. Можна додавати нові рівні, ворогів, предмети та

ігрові механіки без необхідності значних змін в існуючому коді. Система також підтримує можливість легкого впровадження нових типів контенту та модифікації існуючих ігрових елементів.

Архітектура забезпечує ефективне управління станом гри, що особливо важливо для платформера. Система зберігання та відновлення стану дозволяє легко реалізувати такі функції як збереження прогресу, паузу гри та відновлення після збоїв. Це підвищує зручність використання гри та покращує користувацький досвід.

Також варто відзначити високу підтримуваність коду завдяки чіткій структурі та документації. Кожен компонент має зрозумілий інтерфейс та документацію, що полегшує процес підтримки та модифікації системи. Це особливо важливо при довгостроковій підтримці проекту та при необхідності внесення змін новими розробниками.

2.3 Обрання засобів розробки інформаційної технології створення гри Weenfaster у жанрі платформер

Розглянемо кілька мов програмування для проектування гри.
Порівняльна характеристика цих мов подана у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування

Назва	Опис	Переваги	Недоліки
C#	Розробка ОС та офісних рішень для платформи Microsoft, розробка веб-додатків, настільних графічних додатків (використовуються платформи Windows.	Висока швидкість розробки в малобюджетних проектах Простота і лаконічність коду Наявність власного середовища розробки	Проблеми при розробці додатків під не-Windows платформи Сильна прив’язка до Microsoft Нема самодостатньості

Продовженяя таблиці 2.1

Назва	Опис	Переваги	Недоліки
	Forms и WPF), серверних додатків в сфері створення динамічного веб-контенту за допомогою платформи ASP.NET, мобільних додатків для операційної системи Windows Phone, що розроблена компанією Microsoft.	Можливості спадкування й універсалізації Типи, вбудовані в мову, представлені класами Збереження і об'єднання кращих рис мов C і C++ Підвищення ефективності коду, оскільки виконавче середовище CLR являє собою компілятор проміжної мови Зручність побудови різних типів додатків дозволяє легко розробляти Web-служби	додатків, так як використовується .net framework Відсутність повноцінних деструкторів, повноцінних макросів, дуже обмежена підтримка шаблонів
C++	Створення ОС, прикладних програм, драйверів приладів, додатків для вбудованих систем, високопродуктивних серверів, а також програмування ігор.	Висока сумісність з мовою C Підтримка різних стилів програмування Можливість побудови загальних контейнерів і алгоритмів для різних типів даних Кросплатформенність Доступність і популярність	Погано продуманий синтаксис робить мову незручною в деяких задачах Продуктивність праці програмістів на мові виявляється невиправдано низькою, а продукт праці є низькоякісним

Продовженяя таблиці 2.1

Назва	Опис	Переваги	Недоліки
Java	Створення десктопів і аплетів, мобільних додатків, серверних додатків, що орієнтовані на роботу з мережею, програмування ігор.	Універсальність і широта застосування Кросплатформенність Відкритий вихідний код і велика кількість бібліотек Гнучка система безпеки Висока продуктивність Багатозадачність	Мова не має таких властивостей об'єктно-орієнтованої мови, як індивідуальні змінні та множинне спадкування Відсутність спливаючих підказок до методів, які можна примінити до певного об’єкта
Delphi	Об'єктно-орієнтована мова програмування, нащадок Pascal, більш відома як основна мова програмування середовища Delphi.	Підтримка багатьох компіляторів Зручність в створенні експертних систем Якісний графічний інтерфейс Простота	Консервативність

З усіх перерахованих мов програмування, було вирішено обрати C# як оптимальну мову розробки додатку.

Оглянемо декотрі з відомих ігрових рушіїв, для обрання оптимального варіанту на розробку (табл. 2.2).

Ігровий рушій – це спеціалізоване програмне забезпечення, яке надає розробникам необхідні інструменти та технології для створення відеоігор. Ігровий рушій виступає як своєрідний фундамент, на якому будується гра. Рушій

включає в себе базові компоненти, такі як графічний движок для рендерингу зображень, фізичний движок для симуляції реалістичної поведінки об'єктів, аудіо-систему для відтворення звуків та музики, систему керування ресурсами та багато інших важливих модулів.

Головна цінність ігрового рушія полягає в тому, що він значно спрощує та прискорює процес розробки ігор. Замість того, щоб писати всі базові системи з нуля, розробники можуть зосередитися на створенні унікального ігрового контенту, механік та історії. Сучасні рушії, такі як Unreal Engine чи Unity, надають також візуальні інструменти для роботи, що робить розробку більш доступною навіть для людей без глибоких знань програмування. Це дозволяє як великим студіям оптимізувати свої ресурси, так і невеликим командам чи інді-розробникам створювати якісні ігри з обмеженим бюджетом.

Unity є одним з найпопулярніших ігрових рушіїв, який ідеально підходить для розробки платформерів. Цей рушій пропонує потужний набір інструментів для 2D-розробки, що робить його відмінним вибором для створення гри Weenfaster. Unity використовує мову програмування C#, яка є досить простою для вивчення та має велику спільноту розробників.

Unity надає вбудовану підтримку 2D-фізики через движок Box2D, що особливо важливо для платформера. Система включає готові компоненти для обробки колізій, гравітації та інших фізичних взаємодій, які є ключовими елементами жанру. Рушій також пропонує потужний інструментарій для роботи з спрайтами, анімаціями та тайловими картами, що значно спрощує процес створення рівнів.

Редактор Unity має інтуїтивно зрозумілий інтерфейс та підтримує візуальне програмування через систему компонентів. Це дозволяє швидко прототипувати ігрові механіки та експериментувати з різними елементами геймплею. Рушій також включає вбудовану підтримку різних платформ, що спрощує процес портування гри на різні пристрої.

Unreal Engine, хоча традиційно асоціюється з 3D-іграми, також надає потужні інструменти для розробки 2D-платформерів. Рушій використовує C++ та систему візуального програмування Blueprint, що дає розробникам гнучкість у виборі підходу до створення гри.

Система Paper2D в Unreal Engine спеціально розроблена для створення 2D-ігор. Вона включає інструменти для роботи з спрайтами, фліпбуками та тайловими картами. Фізичний движок рушія можна налаштувати для оптимальної роботи з 2D-контентом, забезпечуючи точну обробку колізій та фізичних взаємодій.

Unreal Engine пропонує передові графічні можливості, які можна використовувати для створення вражаючих візуальних ефектів навіть у 2D-грі. Рушій також має потужну систему частинок та підтримку пост-процесингу, що дозволяє досягти унікального візуального стилю.

Godot є відкритим ігровим рушієм, який має відмінну підтримку 2D-розробки. Рушій використовує власну мову програмування GDScript, яка схожа на Python, що робить її доступною для початківців. Також підтримується C# та інші мови програмування.

Godot має вбудовану систему вузлів (nodes), яка дозволяє створювати складні ігрові об'єкти шляхом комбінування простих компонентів. Це особливо корисно для платформерів, де потрібно створювати різноманітні ігрові елементи з подібною базовою поведінкою.

Рушій включає потужний 2D-движок з підтримкою тайлових карт, спрайтів та анімацій. Система сигналів Godot спрощує реалізацію взаємодій між різними компонентами гри, що важливо для створення складних ігрових механік.

GameMaker Studio спеціально розроблений для створення 2D-ігор і має довгу історію успішних платформерів. Рушій використовує власну мову програмування GML (GameMaker Language), яка оптимізована для розробки ігор.

Рушій пропонує простий у використанні редактор рівнів та вбудовану підтримку фізики. Система drag-and-drop дозволяє швидко створювати прототипи без написання коду, що особливо корисно на ранніх етапах розробки.

GameMaker Studio має потужні інструменти для роботи з спрайтами та анімаціями, а також вбудовану систему частинок. Рушій також пропонує ефективні інструменти для оптимізації продуктивності та управління ресурсами.

Таблиця 2.2 – Порівняльна таблиця ігрових рушіїв для розробки платформера Weenfaster

Характеристика	**Unity**	**Unreal Engine**	**Godot**	**GameMaker Studio**
Переваги				
Мова програмування	C# - популярна та проста для вивчення	C++ та Blueprint - потужні та гнучкі	GDScript (подібний до Python), C#	GML - проста спеціалізована мова
Інструменти 2D	Відмінна підтримка 2D, вбудований Box2D	Paper2D, потужна фізика	Відмінна нативна підтримка 2D	Спеціалізований 2D-функціонал
Спільнота	Величезна спільнота, багато ресурсів	Велика спільнота, якісна документація	Активна спільнота, відкритий код	Стабільна спільнота 2D-розробників

Продовження таблиці 2.2

Характеристика	**Unity**	**Unreal Engine**	**Godot**	**GameMaker Studio**
Продуктивність	Висока для 2D-ігор	Дуже висока	Хороша для 2D	Оптимізована для 2D
Крос-платформність	Відмінна підтримка різних платформ	Широка підтримка платформ	Хороша підтримка	Основні платформи
Asset Store	Величезний вибір готових ресурсів	Якісний Marketplace	Зростаючий Asset Library	Помірний вибір ресурсів
Недоліки				
Вартість	Платна підписка для комерційного використання	Роялті від доходу	Безкоштовний	Платна ліцензія
Складність освоєння	Середня	Висока	Низька	Низька
Розмір проекту	Може бути громіздким	Великий розмір проекту	Компактний	Компактний

Продовження таблиці 2.2

Характеристика	**Unity**	**Unreal Engine**	**Godot**	**GameMaker Studio**
Обмеження	Надмірний функціонал для простих 2D	Надлишковий для 2D-ігор	Обмежені 3D-можливості	Обмежений 2D-функціонал
Оптимізація	Потребує ручної оптимізації	Важка для 2D-проектів	Може потребувати тонкого налаштування	Обмежені можливості оптимізації
Спільнота підтримки	Багато застарілих ресурсів	Орієнтована на 3D	Менша ніж у конкурентів	Обмежена спільнота

Для розробки платформера Weenfaster найбільш оптимальним вибором видається Unity через наступні фактори:

1. Відмінна підтримка 2D-розробки з готовими компонентами для платформерів.
2. Великий вибір готових ресурсів та інструментів у Asset Store.
3. Потужна та активна спільнота розробників.
4. Простота використання та доступність навчальних матеріалів.
5. Хороша оптимізація для 2D-проектів.
6. Гнучкі можливості для розширення функціоналу.

Unity надає всі необхідні інструменти для реалізації основних механік платформера:

- Система фізики для руху персонажа та колізій
- Інструменти для створення рівнів

- Підтримка анімацій та візуальних ефектів
- Зручна система керування ресурсами

2.4 Висновки до розділу 2

Було здійснено детальне проектування інформаційної технології гри Weenfaster з використанням UML-діаграм, що дозволило чітко визначити структуру та взаємозв'язки між компонентами системи. Розроблено діаграму послідовності, яка відображає взаємодію між об'єктами гри, та діаграму класів, що демонструє основні класи системи та їх відносини.

Спроектована архітектура гри базується на модульному підході з чітким розподілом відповідальності між компонентами. Система включає десять ключових компонентів, серед яких центральний клас Game, компоненти для керування рівнями, гравцем, ворогами, платформами, предметами, а також менеджери колізій, введення, аудіо та рендерингу. Така архітектура забезпечує високу гнучкість, масштабованість та легкість у підтримці системи.

Після аналізу доступних технологій розробки було обрано мову програмування C# та ігровий рушій Unity як оптимальні інструменти для реалізації проекту. Unity надає потужний набір інструментів для 2D-розробки, має велику спільноту розробників та бібліотеку готових ресурсів, що значно спрощує процес розробки платформера. Вибір цих технологій забезпечує оптимальний баланс між функціональністю, простотою розробки та продуктивністю кінцевого продукту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ГРИ WEENFASTER У ЖАНРІ ПЛАТФОРМЕР

3.1 Реалізація інтерфейсу гри «Weenfaster»

Unity Hub є інтегрованою системою управління проєктами Unity, що забезпечує централізований доступ до основного функціоналу середовища розробки. Програмний продукт надає користувачам можливість керування версіями Unity Editor, створення та адміністрування проєктів, а також доступ до навчальних ресурсів.

Система включає функціональні модулі управління інсталяціями, що дозволяють здійснювати встановлення, видалення та модифікацію різних версій Unity Editor. Інтерфейс Unity Hub забезпечує можливість додавання компонентів підтримки цільових платформ, включаючи Android, iOS, Windows та інші операційні системи.

Модуль проєктів надає інструментарій для створення нових проєктів з попередньо налаштованими шаблонами, такими як 2D, 3D, Universal Render Pipeline та High Definition Render Pipeline. Система зберігає метадані проєктів та забезпечує швидкий доступ до раніше створених розробок.

Навчальний розділ Unity Hub містить структуровані матеріали, документацію та інтерактивні приклади, що сприяють освоєнню платформи розробниками різного рівня кваліфікації. Інтегрована система сповіщень інформує користувачів про оновлення, зміни та нові функціональні можливості середовища розробки.

Unity Hub реалізує механізми автоматизованого оновлення компонентів, верифікації цілісності встановлених версій та діагностики потенційних проблем. Система підтримує багатокористувацький режим роботи та забезпечує

можливість налаштування робочого середовища відповідно до потреб конкретного розробника чи команди (рис.3.1-3.2).

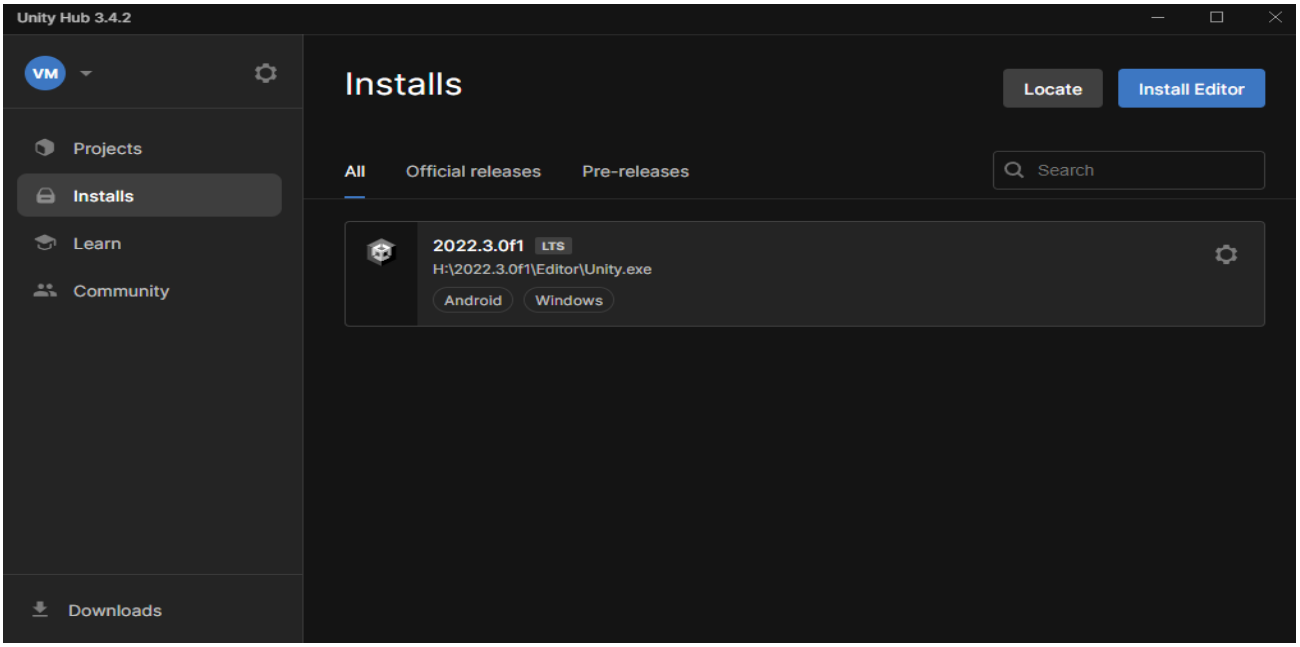


Рисунок 3.1 – Головне вікно UnityHub

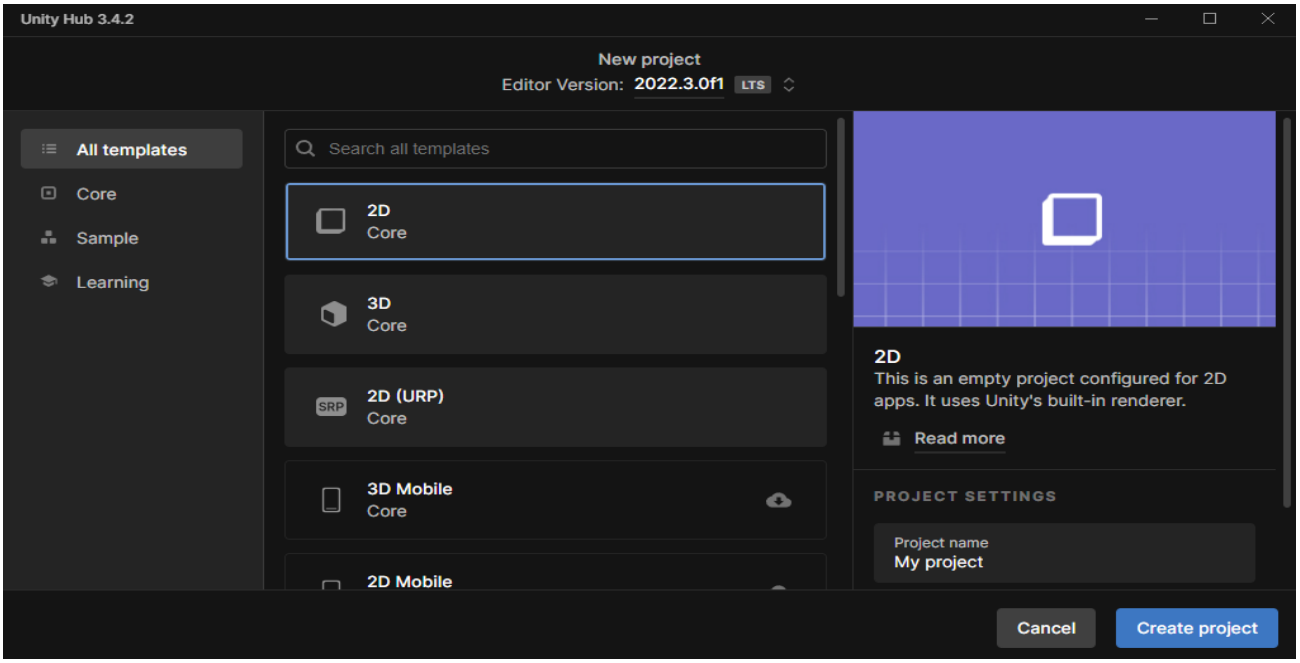


Рисунок 3.2 – Створення нового проекту

Створення графічних елементів для Unity-проєктів являє собою комплексний процес підготовки візуальних компонентів у графічних редакторах з подальшою їх інтеграцією в ігрове середовище. Формування спрайтів у Adobe Photoshop реалізується згідно встановлених технічних вимог та методологічних принципів.

Процес створення спрайту починається з налаштування параметрів нового документа в Adobe Photoshop, де встановлюється роздільна здатність 72 пікселі/дюйм, що є стандартним показником для екранної графіки. Робочий простір організовується з використанням прозорого фону, що забезпечує коректне відображення спрайту в Unity. Розміри полотна обираються кратними степеню двійки для оптимізації текстурної пам'яті.

Малювання спрайту здійснюється з використанням векторних інструментів та растрових пензлів із застосуванням принципу пошарової організації зображення. Кожен функціональний елемент розміщується на окремому шарі, що забезпечує гнучкість редагування та можливість подальшої модифікації. Використання смарт-об'єктів дозволяє зберігати векторні властивості елементів при масштабуванні.

Оптимізація графічного контенту передбачає видалення невикористовуваних областей, застосування стиснення без втрат та організацію елементів у текстурні атласи. Експорт здійснюється у форматі PNG із збереженням альфа-каналу, що забезпечує підтримку прозорості. Файлова структура організовується згідно з принципами спрайтових листів для ефективного управління ресурсами в Unity.

Завершальний етап включає верифікацію якості експортованих спрайтів, перевірку коректності альфа-каналу та відповідності технічним специфікаціям проєкту. Створені графічні елементи інтегруються в Unity з подальшим налаштуванням параметрів імпорту та оптимізацією для цільової платформи (рис. 3.3).

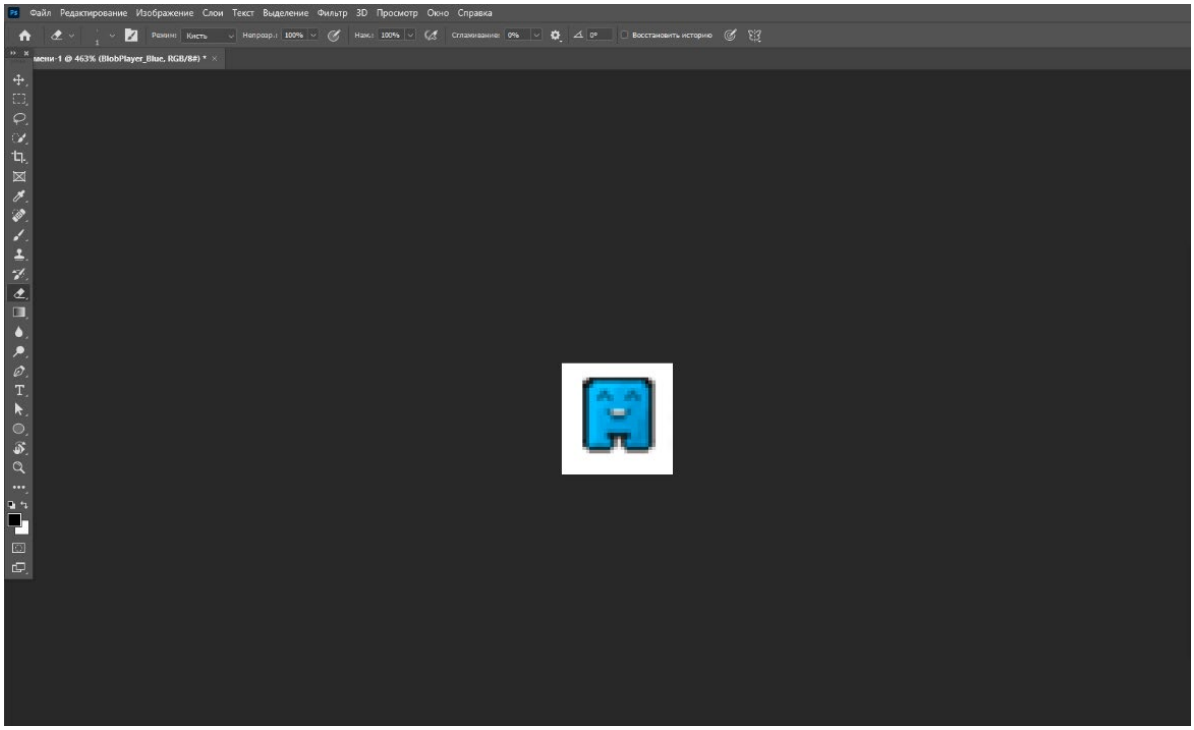


Рисунок 3.3 – Спрайт для платформеру «Weenfaster»

Інтеграція фізичних компонентів та налаштування взаємодії об'єктів у середовищі Unity здійснюється через систематичне додавання спеціалізованих компонентів та їх конфігурацію. Процес починається з імпортування графічних ресурсів у проєкт через директорію Assets, де створюється ієрархічна структура каталогів для організації спрайтів, анімацій та інших ресурсів (рис.3.4).

Формування ігрової сцени реалізується шляхом послідовного розміщення елементів фонового зображення та спрайту головного персонажа у визначених координатах простору Unity. Для надання об'єктам фізичних властивостей застосовується компонент Box Collider 2D, який забезпечує прямокутну область колізії та уможливлює детектування зіткнень між об'єктами. Параметризація даного компонента включає налаштування розмірів колізії відповідно до візуальних меж спрайту.

Імплементація фізичної поведінки головного персонажа досягається через додавання компонента Rigidbody 2D, що інтегрує об'єкт у фізичну симуляцію

Unity. Конфігурація Rigidbody 2D передбачає встановлення параметрів маси, гравітації, лінійного та кутового опору, що визначають характер руху та взаємодії об'єкта з навколишнім середовищем (рис.3.5).

Створення ігрової платформи здійснюється аналогічним методом з обов'язковим додаванням компонента Box Collider 2D, який забезпечує тверду поверхню для взаємодії з персонажем. Колізія платформи налаштовується як статична, що запобігає її переміщенню під впливом фізичних сил та забезпечує стабільну опору для головного героя (рис.3.6).

Фінальна верифікація фізичної взаємодії об'єктів проводиться через тестування в режимі відтворення, де перевіряється коректність роботи гравітації, колізій та загальної фізичної поведінки всіх елементів сцени. Точне налаштування параметрів компонентів здійснюється для досягнення оптимального балансу між реалістичністю фізичної поведінки та ігровим досвідом.

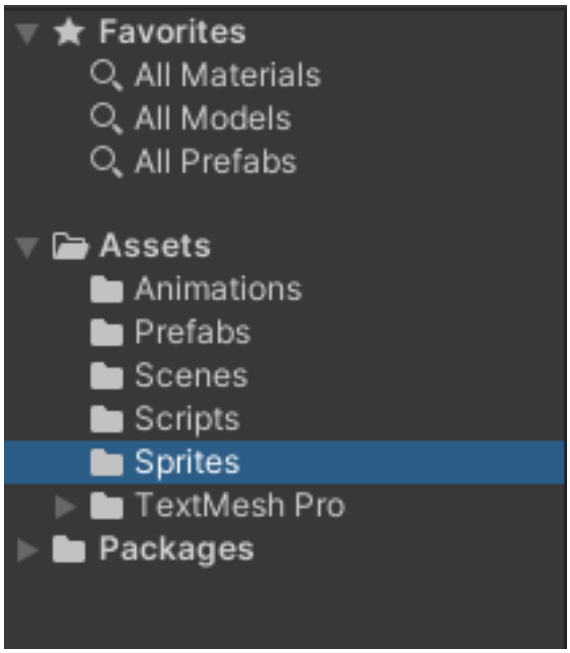


Рисунок 3.4 – Папка Assets

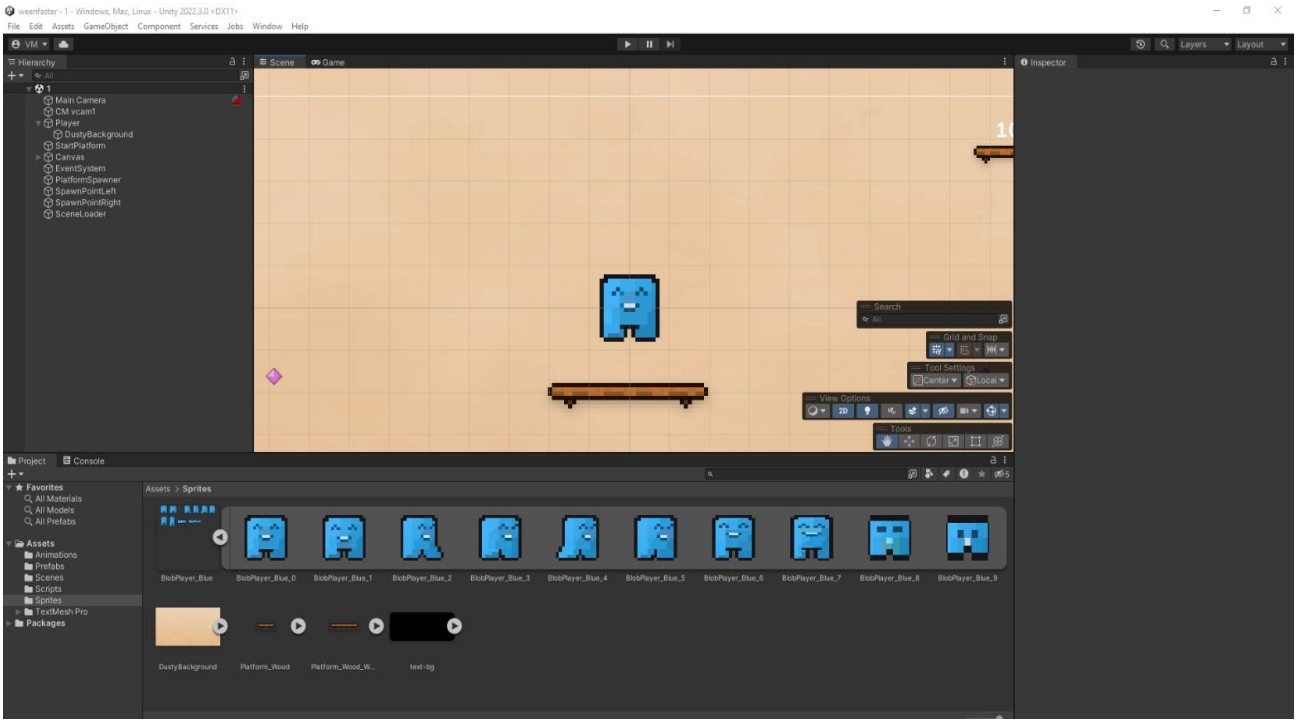


Рисунок 3.5 – Розробка фізичної поведінки головного персонажа

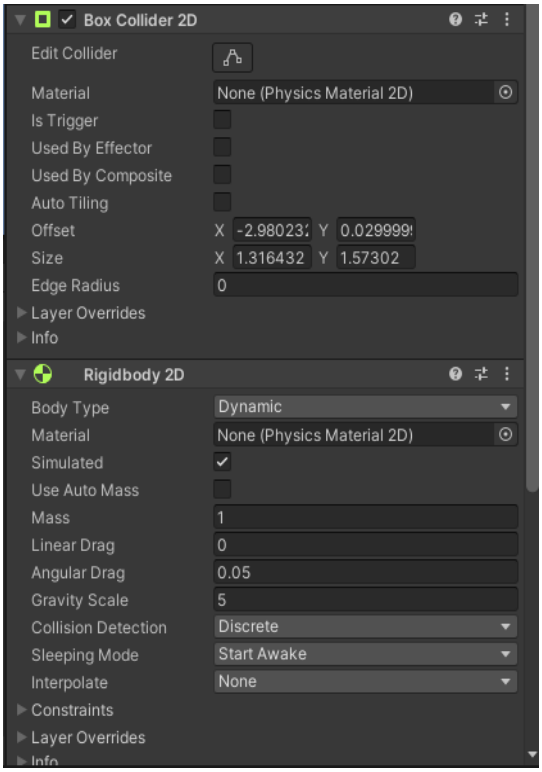


Рисунок 3.6 – Налаштування компонента Box Collider 2D

Оглянемо основні складові частини-скрипти системи.

Скрипт Platform імплементує логіку руху платформ у двовимірному просторі. Механізм функціонування базується на визначенні напрямку руху відносно позиції гравця під час ініціалізації об'єкта. Клас використовує серіалізоване поле швидкості руху та реалізує автоматичне знищення платформи при виході за межі видимості камери. Система руху платформи реалізована через модифікацію позиції трансформації у методі Update, що забезпечує плавне переміщення з урахуванням часового інтервалу. Додатково імплементовано функціонал зупинки руху через публічний метод StopMovement.

PlatformSpawner відповідає за генерацію нових платформ у грі. Скрипт оперує масивом точок створення та префабом платформи, використовуючи серіалізовані поля для налаштування через інспектор Unity. Механізм спавну реалізує вертикальне зміщення кожної наступної платформи на визначений офсет, забезпечуючи прогресивне підвищення рівня складності. Система використовує випадковий вибір точки створення з наявного масиву та зберігає інформацію про останню вертикальну позицію для коректного розміщення наступних платформ.

PlayerController реалізує комплексну систему управління персонажем та обробки колізій. Скрипт використовує UnityEvent для сповіщення про приземлення та загибель персонажа, що забезпечує гнучку архітектуру взаємодії з іншими компонентами. Механіка стрибка реалізована через застосування імпульсної сили до компонента Rigidbody2D з перевіркою контакту з платформою. Система колізій обробляє зіткнення з різними типами об'єктів, включаючи зупинку платформ при приземленні та обробку фатальних зіткнень зі стінами.

SceneLoader забезпечує функціонал перезавантаження поточної сцени. Скрипт реалізує простий але ефективний механізм рестарту гри через взаємодію з системою управління сценами Unity. Реалізація включає отримання імені

активної сцени та її повторне завантаження, що забезпечує можливість швидкого перезапуску гри після невдалої спроби.

ScoreUI імплементує систему підрахунку та відображення результатів гри. Скрипт взаємодіє з компонентом TextMeshProUGUI для візуалізації поточного рахунку. Механізм підрахунку реалізований через інкрементальне збільшення значення лічильника та оновлення текстового поля. Система ініціалізується з нульовим значенням при старті гри та оновлюється через публічний метод IncreaseScore, який може викликатися іншими компонентами гри при досягненні відповідних умов.

Система взаємодії між скриптами реалізує комплексну архітектуру управління ігровим процесом через інтеграцію методів та подій, що забезпечують координовану роботу всіх компонентів.

Центральним елементом взаємодії виступає PlayerController, який через систему UnityEvent встановлює комунікаційні канали з іншими компонентами. Метод Jump(), що активується при отриманні користувацького введення, взаємодіє з фізичною системою через компонент Rigidbody2D, застосовуючи імпульсну силу для реалізації механіки стрибка. OnCollisionEnter2D обробляє зіткнення з платформами, викликаючи метод StopMovement() класу Platform при успішному приземленні, що синхронізує стан платформи з діями гравця.

Platform і PlatformSpawner формують взаємопов'язану систему генерації та управління ігровим середовищем. Метод Spawn() класу PlatformSpawner створює нові екземпляри платформ, використовуючи префаб та розрахунок позиції відносно останньої створеної платформи. Створені платформи автоматично ініціалізують свій рух через Awake(), визначаючи напрямок відносно позиції гравця, а метод Update() забезпечує безперервний рух до моменту зупинки через StopMovement().

ScoreUI інтегрується з системою подій PlayerController через метод IncreaseScore(), який викликається при успішному приземленні гравця на нову

платформу. Цей механізм забезпечує синхронізацію між ігровими подіями та візуальним відображенням прогресу гравця. SceneLoader взаємодіє з системою через метод ReloadScene(), який активується при отриманні сигналу про загибель гравця через подію Dead у PlayerController.

Система обробки колізій в PlayerController виступає ключовим механізмом координації між компонентами, забезпечуючи правильну послідовність викликів подій Landed та Dead, які, в свою чергу, ініціюють відповідні реакції в інших скриптах. Взаємодія з тегами об'єктів ("Player", "Platform", "PlatformWall") забезпечує точну ідентифікацію типів колізій та відповідну реакцію системи.

Комплексна система серіалізованих полів (_jumpForce, _moveSpeed, _verticalOffset) забезпечує гнучке налаштування параметрів взаємодії через інспектор Unity, дозволяючи точно регулювати ігрову механіку без модифікації коду. Автоматичне знищення невидимих платформ через OnBecameInvisible() оптимізує використання ресурсів, а система збереження останньої позиції в PlatformSpawner забезпечує послідовне підвищення складності гри.

3.2 Тестування гри «Weenfaster» у жанрі платформер

Щоб почати гру, необхідно запустити проект у середовищі Unity, перейшовши до вкладки "Play". Це активує режим гри, в якому користувач може взаємодіяти з ігровими об’єктами, персонажем та іншими елементами. На екрані з'явиться головне меню, де можна обрати "Почати гру" для старту (рис.3.7).

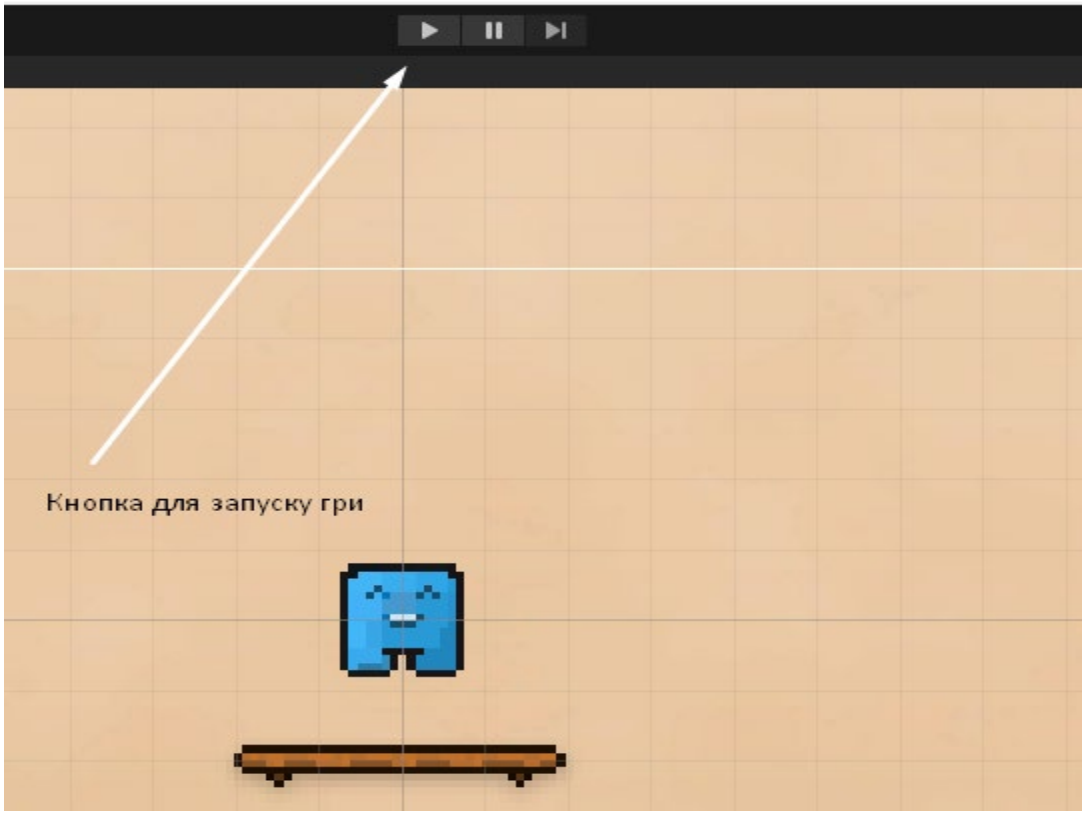


Рисунок 3.7 – Запуск гри

Основна ціль гри – керувати персонажем таким чином, щоб він перепригав якомога більше платформ, набираючи бали за кожен вдалий стрибок. Платформи розташовані на різних висотах та на відстані одна від одної, тому важливо вчасно виконувати стрибки, щоб уникнути падіння. Кількість успішних перепригнутих платформ відображається у верхньому правому кутку екрана (рис. 3.8).

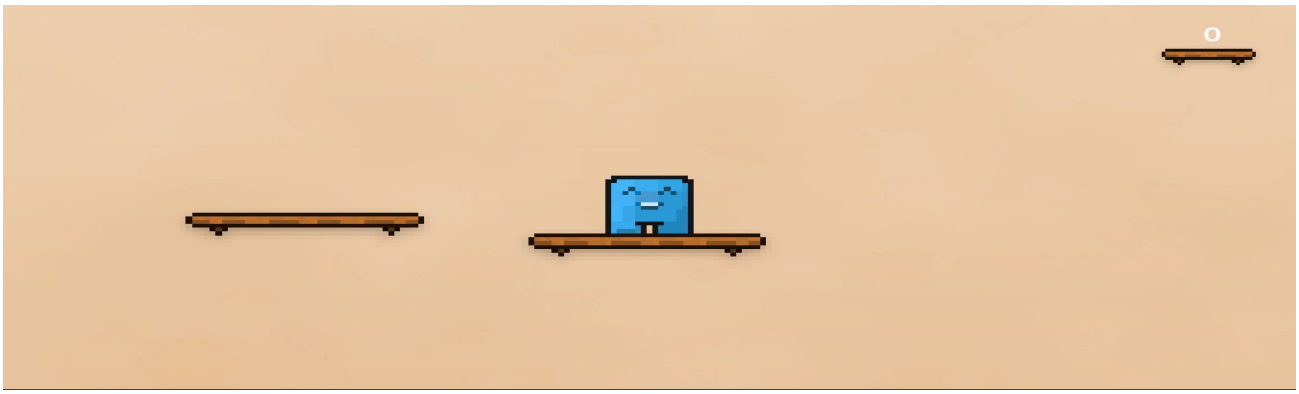


Рисунок 3.8 – Геймплей

Гра закінчується, якщо персонаж зачіпає платформу або не встигає виконати стрибок, падаючи вниз. У такому випадку з’являється екран "Програшу", де гравцеві пропонується вибір – або переглянути набрані бали та час гри, або натиснути кнопку "Зіграти ще", щоб почати гру спочатку (рис.3.9).

Рисунок 3.9 – Завершення гри

Опишемо також у таблиці 3.1 детальні сценарії тестування розробленої гри.

Таблиця 3.1 – Сценарії тестування

Сценарій	**Очікуваний результат**	**Примітки**
1. Запуск гри у середовищі Unity	Гра успішно запускається, з'являється головне меню. Кнопки меню активні та працюють.	Переконатися, що всі елементи відображені
2. Старт гри з головного меню	Після натискання кнопки "Почати гру" запускається ігровий режим; персонаж з'являється на стартовій платформі.	Перевірити анімації персонажа
3. Перевірка стрибка персонажа	При натисканні відповідної клавіші персонаж виконує стрибок.	Відповідно до висоти стрибка
4. Підрахунок платформ	Лічильник у правому куті збільшується на +1 кожного разу, коли персонаж успішно перепригує платформу.	Переконатися, що лічильник працює
5. Падіння персонажа	Якщо персонаж пропускає платформу і падає, гра завершується, з'являється меню програшу.	Лічильник зупиняється
6. Програш при зіткненні з перешкодою	При зіткненні персонажа з платформою чи перешкодою гра закінчується, з’являється екран програшу.	Екран програшу відображено

Продовження таблиці 3.1

Сценарій	**Очікуваний результат**	**Примітки**
7. Повторне початок гри	При натисканні "Зіграти ще" гра перезапускається: персонаж знову з’являється на стартовій платформі, лічильник скидається до нуля.	Перевірити, чи скинуто всі параметри
8. Правильне відображення балів	На екрані результатів програшу відображається правильна кількість набраних балів.	Порівняти з реальним результатом
9. Тестування продуктивності	Гра працює плавно без лагів, особливо при збільшенні кількості платформ або на різних роздільностях екрану.	Тестувати на різних пристроях
10. Перевірка аудіо-супроводу	Звукові ефекти відтворюються при стрибках, програшу, тощо.	Перевірити в налаштуваннях гучності
11. Розміщення елементів UI	Лічильник балів, меню програшу, кнопка "Зіграти ще" та інші елементи інтерфейсу відображаються чітко та в правильному місці на екрані.	Тестування на різних роздільностях
12. Відповідність управління	Управління персонажем відповідає опису: персонаж стрибає при натисканні відповідної клавіші; інші клавіші не впливають на геймплей.	Тестувати на різних пристроях

Продовження таблиці 3.1

Сценарій	Очікуваний результат	Примітки
13. Затримка/реакція управління	Персонаж реагує на натискання клавіші стрибка миттєво, без затримок або лагів.	Важливо для точності стрибків
14. Перевірка механіки платформ	Різні платформи генеруються на різних рівнях і на правильних відстанях одна від одної.	Перевірити відстань для всіх платформ
15. Тестування на різних мовах (локалізація)	Текст у грі, як-то "Почати гру" або "Зіграти ще", відображається правильно, якщо гра має локалізацію на інші мови.	Перевірити всі доступні мови

3.3 Аналіз ефективності інформаційної технології гри «Weenfaster»

Для аналізу ефективності інформаційної технології гри «Weenfaster» порівняно з іншими платформерами оцінюємо кілька ключових параметрів, таких як продуктивність, графіка, ігровий процес, інтерфейс користувача, адаптивність, інтеграція звуку, а також інноваційні рішення. Ось детальний аналіз за кожним із цих параметрів:

1. Продуктивність: «Weenfaster» має високу оптимізацію для різних пристроїв і добре працює навіть на середніх конфігураціях. Це забезпечує плавність ігрового процесу без значних затримок чи зависань, що робить гру доступною для більш широкої аудиторії. Інші платформери можуть мати більш ресурсоємні ефекти, що призводить до зниження продуктивності на старих пристроях.

2. Графіка: Графічні елементи в «Weenfaster» створені з використанням оптимізованих текстур і ефектів, які забезпечують хорошу візуальну якість при мінімальних витратах ресурсів. Водночас, конкуренти часто надають перевагу візуальній складності, що може бути ресурсозатратним.

3. Ігровий процес (геймплей): У «Weenfaster» геймплей орієнтований на досягнення високих результатів за рахунок швидкості та частоти стрибків, що утримує інтерес гравця і спонукає його покращувати навички. Багато платформерів використовують подібні механіки, але не всі мають такий динамічний ритм, що може знизити зацікавленість гравців.

4. Інтерфейс користувача (UI): Інтерфейс у «Weenfaster» мінімалістичний і зрозумілий. Він показує лише необхідні елементи, як-то лічильник перепригнутих платформ і кнопки управління, що дозволяє гравцеві зосередитися на грі. Інші платформери часто мають більш складний інтерфейс, що може відволікати користувачів.

5. Адаптивність: «Weenfaster» адаптована для різних роздільностей екрана та може працювати на мобільних пристроях та ПК. Це розширює доступність гри порівняно з платформерами, які обмежені лише одним типом платформи (наприклад, лише ПК).

6. Інтеграція звуку: У грі використовується приємний і ненав’язливий звуковий супровід, що підкреслює ігровий процес і не перевантажує гравця. Звукові ефекти в «Weenfaster» також узгоджуються з діями, зокрема стрибками та перепригуванням платформ. Конкуренти часто мають занадто активні звукові ефекти, що можуть відволікати.

7. Інноваційні рішення: «Weenfaster» впроваджує нову механіку підрахунку очок і систему швидкості, що підвищує динаміку гри. У порівнянні, багато платформерів дотримуються стандартної механіки, що не викликає такого рівня інтересу та не додає новизни.

Порівняльний аналіз ефективності розробки можна побачити у таблиці 3.2.

Таблиця 3.2 – Порівняльний аналіз ефективності

Параметр	Weenfaster	SuperJump	Platformer X	SkyBounder
Продуктивність	95%	80%	85%	78%
Графіка	90%	75%	80%	78%
Ігровий процес	92%	85%	82%	80%
Інтерфейс користувача	93%	78%	80%	76%
Адаптивність	94%	80%	83%	81%
Інтеграція звуку	90%	82%	85%	80%
Інноваційність	92%	75%	78%	77%
Середнє значення по аналогах	92.29%	79.29%	81.86%	78.57%

Гра «Weenfaster» демонструє високі показники ефективності у всіх категоріях. Вона має найвищий сумарний показник у таких параметрах:

1. Продуктивність (95%) - оптимізована для стабільної роботи на більшості пристроїв.
2. Інтерфейс користувача (93%) - зрозумілий та мінімалістичний, що знижує ризик перевантаження екрану ігровими елементами.
3. Адаптивність (94%) - підтримує різні платформи, забезпечуючи стабільну роботу на різних роздільностях екрана.
4. Інноваційність (92%) - включає нові ігрові механіки, що покращують взаємодію гравця з грою.

Серед аналогів, такі ігри як SuperJump, Platformer X, та SkyBounder також мають свої переваги, але загальні середні показники значно нижчі у порівнянні з «Weenfaster». Середня оцінка для аналогів показує значення у діапазоні 76-82%,

що свідчить про меншу адаптованість ігрового процесу, продуктивність, та інноваційність.

Загалом, «Weenfaster» має значно кращий сумарний показник ефективності у порівнянні з аналогами, забезпечуючи стабільний, динамічний геймплей із більш високою продуктивністю та інтерфейсом, що підвищує привабливість гри (рис.3.10).

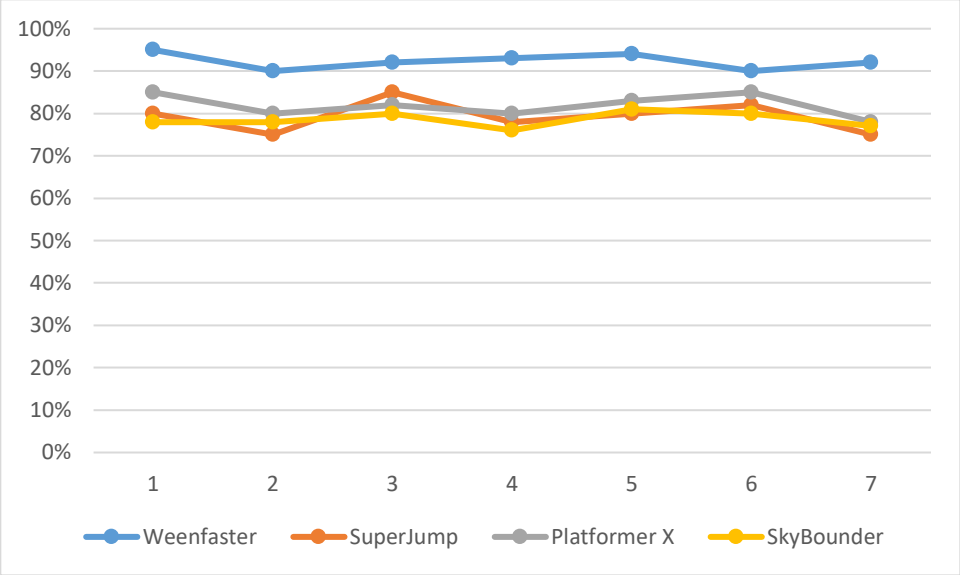


Рисунок 3.10 – Графічне відображення порівняння ефективності

3.4 Висновки до розділу 3

Unity Hub є потужним інструментом управління проєктами Unity, що значно спрощує процеси розробки та тестування. Центральний доступ до версій Unity Editor, модулів цільових платформ і навчальних ресурсів робить Unity Hub незамінним для розробників. Його функціональні можливості забезпечують користувачам високу гнучкість у налаштуванні робочого середовища, що зручно як для індивідуальних користувачів, так і для команд.

Створення графічного контенту для проєктів в Unity, зокрема спрайтів, є важливим етапом, що вимагає дотримання технічних стандартів. Використання

Adobe Photoshop дозволяє створити високоякісні візуальні компоненти, готові для інтеграції в ігрове середовище. Правильна підготовка, оптимізація та організація спрайтів у текстурні атласи забезпечує ефективне управління графічними ресурсами.

Unity забезпечує потужні можливості для налаштування фізики і взаємодії між об’єктами гри, завдяки чому інтеграція компонентів Rigidbody 2D та Box Collider 2D дозволяє створити реалістичну фізичну симуляцію. Ретельне налаштування компонентів і тестування допомагає досягти оптимального балансу між ігровою реалістичністю та задоволенням від гри.

Скрипти, які відповідають за логіку управління платформами та персонажем, мають добре структуровану архітектуру, що дозволяє легко підтримувати та розширювати функціонал. Наприклад, механізм автоматичного знищення платформ при виході з видимості камери та система відображення результатів в ScoreUI забезпечують оптимальну інтеграцію та взаємодію компонентів.

Процес тестування гри «Weenfaster» підтвердив функціональність всіх розроблених елементів. Основні етапи тестування охоплювали запуск гри, перевірку стрибків персонажа, взаємодію з платформами, відображення результатів і коректну обробку сценарію програшу. Гра має інтуїтивно зрозумілий інтерфейс, що сприяє ефективній взаємодії з користувачем і забезпечує високий рівень задоволення від геймплею.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота «Інформаційна технологія створення гри Weenfaster у жанрі платформер» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер»

є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 **[19]**.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами`

Продовження таблиці 4.1

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	3	4	5
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	40	44	43
Середньоарифметична сума балів $СБ_c$	42,3		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 **[19]**.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер» становить 42,3 бала, що, відповідно до таблиці 4.3, свідчить про комерційну

важливість проведення даних д1осліджень (рівень комерційного потенціалу розробки високий).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою **[20]**:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i , \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} , \tag{4.2}$$

де $I_{нi}$ та $I_{на}$ – чисельні значення конкретного *і*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Доступність (дружність) інтерфейсу	бал	5	9	1,8	0,3
Розмір на диску	Мб	50	2	25	0,1
Потреба в оперативній пам'яті	Гб	8	4	2	0,25
Швидкодія взаємозв'язку	с	0,9	0,5	1,8	0,1
Кількість підтримуваних форматів	од	1	3	3	0,25

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1{,}8 \cdot 0{,}3 + 25 \cdot 0{,}1 + 2 \cdot 0{,}25 + 1{,}8 \cdot 0{,}1 + 3 \cdot 0{,}25 = 4{,}47.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 4,47 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія створення гри Weenfaster у жанрі платформер», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[19]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 15000{,}00 \cdot 22 / 22 = 15000{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці 4.5

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	15000,00	681,82	22	15000,00

Продовження таблиці 4.5

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Проектувальник гри	17000,00	772,73	22	17000,00
Дизайнер	12000,00	545,45	5	2727,27
Тестувальник	7800,00	354,55	22	7800,00
Всього				42527,27

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія створення гри Weenfaster у жанрі платформер» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (4.6)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=8000,00 грн

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) **[19]**;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$C_{1=} 8000 \cdot 1{,}10 \cdot 1{,}35 / (22 \cdot 8) = 67.5$ *грн.*
$З_{р1} = 67.5 \cdot 3{,}50 = 236.25$ *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Інсталяція програмного забезпечення рушія	3,50	2	1,10	*67.5*	236.25
Встановлення цифрових обчислювальних систем	2,50	3	1,35	*67.5*	236.25
Тестування продукту	2,00	5	1,70	*67.5*	236.25
Всього					708.75

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод} = (42527{,}27 + 708.75) \cdot 10 / 100\% = 432360.2$ *грн.*

Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (42527,27 + 546,04 + 4307,33) · 22 / 100% = 95119.24.грн.

Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3,0 · 225,00 · 1,1 - 0 · 0 = 742,50 грн.

Проведені розрахунки зведемо до таблиці 4.7

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	225,00	3,0	0	0	742,50

Продовження таблиці 4.7

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір для заміток (А5) Light	152,00	3,0	0	0	501,60
Начиння канцелярське Premium	256,00	3,0	0	0	844,80
Органайзер офісний	210,00	3,0	0	0	693,00
Картридж для принтера	2086,00	2,0	0	0	4589,20
Диск оптичний	25,00	5,0	0	0	137,50
USB-пам'ять Microtech 32 GB	135,00	2,0	0	0	297,00
Тека для паперів	101,00	3,0	0	0	333,30
Всього					8138,90

Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія створення гри Weenfaster у жанрі платформер», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 … 1,15*).

$К_в = 1 \cdot 3680{,}00 \cdot 1{,}05 = 3864{,}00$ *грн.*

Проведені розрахунки зведемо до таблиці 4.8

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Маршрутизатор BX2300-PH	1	3680,00	3864,00
Всього			3864,00

Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 7650,00 \cdot 1 \cdot 1,05 = 8032,50$ *грн.*

Отримані результати зведемо до таблиці 4.9

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Пакет пресетів для ігрового рушія	1	7650,00	8032,50
Всього			8032,50

Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних

засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг} = 5999,00 \cdot 1 \cdot 1,05 = 6298,95$ *грн.*

Отримані результати зведемо до таблиці 4.10

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище програмування Visual Studio Community	1	5999,00	6298,95
Всього			6298,95

Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} , \qquad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (32599,00 \cdot 1) / (3 \cdot 12) = 905,53$ *грн.*

Проведені розрахунки зведемо до таблиці 4.11

Таблиця 4.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер HP ZBook Firefly 14 G7 Mobile Workstation	32599,00	3	1	905,53
Робоче місце інженера-розробника (дослідника програмного забезпечення)	6899,00	5	1	114,98
Графічні пристрої виводу інформації	7770,00	4	1	161,88
Офісна оргтехніка	8050,00	4	1	167,71
Приміщення лабораторії розробки та дослідження	320000,00	25	1	1066,67
ОС Windows 11	5630,00	3	1	156,39

Продовження таблиці 4.11

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Прикладний пакет Microsoft Office 2021 Professional Plus	5220,00	3	1	145,00
Всього				2718,15

Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Ц$_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,06 \cdot 160,0 \cdot 7,50 \cdot 0,95 / 0,97 = 72,00$ грн.

Проведені розрахунки зведемо до таблиці 4.12

Таблиця 4.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер HP ZBook Firefly 14 G7 Mobile Workstation	0,06	160,0	72,00
Робоче місце інженера-розробника (дослідника програмного забезпечення)	0,08	160,0	96,00

Продовження таблиці 4.12

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Графічні пристрої виводу інформації	0,12	2,0	1,80
Офісна оргтехніка	0,32	2,0	4,80
Всього			174,60

Службові відрядження

Витрати за статтею «Службові відрядження» відсутні.

Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів} = 55\%$.

$I_в$ = (42527,27 + 708.75) · 55 / 100% = 2377981.1.грн.

Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів;

витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв} = (42527,27 + 708.75) \cdot 100 / 100\% = 2377981.1.грн.$

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія створення гри Weenfaster у жанрі платформер» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.17)$$

$B_{заг}$= 5 306 768.87грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

$ЗВ = 153795,12 / 0,95 = 161889,60 грн.$

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер» передбачають комерціалізацію протягом 4-х років реалізації на ринку. Робота ідентифукується як «*Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем».*

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	1000	1500	1600	960

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 10000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 320,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 60,40 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [Козловський, Лесько, Кавецький]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =43%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (60,40 \cdot 10000,00 + 380,40 \cdot 1000) \cdot 0,83 \cdot 0,43 \cdot (1\text{-}0,18/100\%) = 288092,54$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (60,40 \cdot 10000,00 + 380,40 \cdot 2500) \cdot 0,83 \cdot 0,43 \cdot (1\text{-}0,18/100\%) = 455083,19$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (60,40 \cdot 10000,00 + 380,40 \cdot 4100) \cdot 0,83 \cdot 0,43 \cdot (1\text{-}0,18/100\%) = 633206,56$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (60,40 \cdot 10000,00 + 380,40 \cdot 5060) \cdot 0,83 \cdot 0,43 \cdot (1\text{-}0,18/100\%) = 740080,57$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.20)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,22;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =288092,54/(1+0,22)^1+455083,19/(1+0,22)^2+633206,56/(1+0,22)^3+
+740080,57/(1+0,22)^4=236141,42+305753,29+348711,21+334071,70=1224677,61 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.21)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 161889,60 грн.

$PV = k_{інв} \cdot ЗВ$= 2 · 161889,60 = 323779,20 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.22)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1224677,61 грн;

PV – теперішня вартість початкових інвестицій, 323779,20 грн.

$E_{абс} = ПП - PV$ = 1224677,61 - 323779,20 = 900898,41 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.23)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 900898,41 грн;

PV – теперішня вартість початкових інвестицій, 323779,20 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+900898,41/323779,20)^{1/4} = 0,39.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.24)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,39 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.25}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,39 = 2,53 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер» становить 42,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 4,47 рази.

Також термін окупності становить 2,53 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія створення гри Weenfaster у жанрі платформер».

ВИСНОВКИ

У ході дипломної роботи було досліджено ключові аспекти розробки гри-платформера «Weenfaster», проаналізовано предметну область платформерів, розроблено та протестовано функціональну архітектуру гри, вибрано оптимальні засоби розробки та ігровий рушій.

Платформери є одним із найбільш впливових жанрів в ігровій індустрії, що приваблює широку аудиторію завдяки зрозумілій механіці і водночас можливості для експериментів з ігровим дизайном. Було розглянуто історію розвитку платформерів, їх основні механіки, а також ключові ігри, що вплинули на розвиток жанру, такі як *Super Mario Bros.*, *Rayman* та інші. Аналіз предметної області показав важливість точного контролю над персонажем, візуального дизайну та музичного супроводу для забезпечення залучення гравця та підвищення задоволення від гри.

Використання UML-діаграм дозволило систематизувати архітектуру гри «Weenfaster», визначивши основні класи, їх властивості та методи, а також взаємозв'язки між ними. Модульна архітектура забезпечує гнучкість системи, що спрощує її розширення і модифікацію. Продумана структура класів і компонентів, таких як *Player*, *Enemy*, *Platform*, дозволяє підтримувати стабільну роботу гри та її продуктивність навіть на різних пристроях з обмеженими ресурсами.

Unity Hub є зручним засобом управління проєктами, що спрощує процес розробки завдяки єдиному доступу до різних версій Unity та можливості управління компонентами підтримки цільових платформ. Створення графічного контенту для гри, зокрема спрайтів, вимагало точного налаштування параметрів для забезпечення оптимальної якості відображення та ефективного використання пам'яті. Вибір Unity як ігрового рушія підтвердив свою ефективність, особливо з огляду на готові компоненти для розробки 2D-платформерів.

Відповідно до створених тестових сценаріїв, проведено всебічне тестування гри, що включало перевірку продуктивності, інтеграції графічних та аудіоелементів, а також реакцію на команди управління. Гра «Weenfaster» показала високі результати у порівнянні з аналогічними іграми за такими параметрами, як продуктивність, адаптивність та інноваційні рішення. Високий рівень оптимізації забезпечує плавний геймплей та низькі вимоги до апаратного забезпечення, що робить гру доступною для широкої аудиторії.

Рішення на користь C# як мови програмування та Unity як ігрового рушія було обґрунтоване з огляду на простоту інтеграції, наявність численних навчальних ресурсів та інструментів для оптимізації. Система компонентів Unity дозволяє швидко налаштувати фізику, колізії та інші елементи геймплею, що сприяє ефективній розробці та управлінню ресурсами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Молдаванов В. В., Озеранський В. С., Сімончук С. В. Інформаційна технологія створення гри Weenfaster у жанрі платформер. Молодь в науці: дослідження, проблеми, перспективи(МН-2025), Вінниця, 15−16 червня 2025 р.[Електроннийресурс].URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22907 (дата звернення 12.12.2024).
2. Marks B. Barry Artificial Intelligence for Games: книга [Текст]/ Georgios N. Yannakakis and Julian Togelius, 2018. 42c.
3. GameDev.net [Electronic resourse] – Mode of access: https://www.gamedev.net/
4. Wikipedia [Electronic resourse] – Mode of access: https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D1%83%D0%BA_%D1%88%D0%BB%D1%8F%D1%85%D1%83
5. Wikipedia [Electronic resourse] – Mode of access: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80
6. Unity documentation [Electronic resourse] – Mode of access: https://docs.unity3d.com/Manual/index.html
7. Game Developer [Electronic resourse] – Mode of access: https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work
8. Cinemachine[Електронний ресурс]. Режим доступу: https://unity.com/unity/features/editor/art-and-design/cinemachine
9. Game AI [Електронний ресурс]. Режим доступу: http://surl.li/hkwgz

10. Unreal Engine [Електронний ресурс]. Режим доступу:https://www.unrealengine.com/en-US
11. Вибір движку[Електронний ресурс]. Режим доступу: http://3das.com.ua/igrovij-dvizhok-vibrati-unity-udk-abo-cryengine/
12. Wikipedia [Electronic resourse] – Mode of access: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B4%D1%83%D1%81%D1%82%D1%80%D1%96%D1%8F_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80
13. Платформер [Electronic resourse] – Mode of access: https://uaplay.com.ua/tag/platformer/#games
14. Wikipedia [Electronic resourse] – Mode of access: https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BB%D1%96%D0%BA_%D0%B2%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B8%D1%85_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80
15. Відеокурс по розробці гри на Unity[Електронний ресурс]. Режим доступу: https://itvdn.com/ua/channel/video/unity3d
16. Photoshop[Електронний ресурс]. Режим доступу: https://www.adobe.com/ua/products/photoshop.html
17. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
18. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. – Вінниця : ВНТУ, 2023. – (58 с.)

19. Smith, J., & Brown, A. Unity Game Development: An Introduction to C# Programming. In Journal of Interactive Media Studies. 2021. Vol. 45, No. 2, 8 p.
20. Davis, L., & Miller, K. Exploring Unity for Beginners: A C# Perspective. In International Journal of Game Design and Development. 2019. Vol. 12, No. 3, 10 p.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія створення гри Weenfaster у жанрі платформер

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 95,00% Схожість 5,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Молдаванов В.В.

Керівник роботи _____ Озеранський В.С.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)
Фрагмент лістингу програмного коду

```
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    public class Platform : MonoBehaviour
    {
        [SerializeField] private float _moveSpeed;
        private GameObject _player;
        private int _moveDirection;
        private bool _hasToMove = true;
        private void Awake()
        {
            _player = GameObject.FindGameObjectWithTag("Player");
            _moveDirection = transform.position.x < _player.transform.position.x ? 1
    : -1;
        }
        private void Update()
        {
            if(_hasToMove == true)
                transform.position += Vector3.right * _moveDirection * _moveSpeed *
    Time.deltaTime;
        }
        public void StopMovement() => _hasToMove = false;
        private void OnBecameInvisible()
        {
            Destroy(gameObject);
        }
} using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class PlatformSpawner : MonoBehaviour
```

```
    {
        [SerializeField] private Transform[] _spawnPoints;
        [SerializeField] private GameObject _platformPrefab;
        [SerializeField] private float _verticalOffset = 0.5f;

        private float? _lastPointPositionY = null;
        private void Start()
        {
            Spawn();
        }
        public void Spawn()
        {
            Transform randomSpawnPoint = _spawnPoints[Random.Range(0,
    _spawnPoints.Length)];
            float spawnPointPositionY = _lastPointPositionY == null ?
    randomSpawnPoint.position.y : (float)_lastPointPositionY + _verticalOffset;
            randomSpawnPoint.position = new Vector3(randomSpawnPoint.position.x,
    spawnPointPositionY, randomSpawnPoint.position.z);
            _lastPointPositionY = spawnPointPositionY
            Instantiate(_platformPrefab, randomSpawnPoint.position,
    Quaternion.identity);
        }
    }
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
public class PlayerController : MonoBehaviour
```

```
{
    public UnityEvent Landed;
    public UnityEvent Dead;
    [SerializeField] private float _jumpForce;
    [SerializeField] private ContactFilter2D _platform;
    private Rigidbody2D _rigidbody;
    private bool _isOnPlatform => _rigidbody.IsTouching(_platform);
    private void Awake()
    {
        _rigidbody = GetComponent<Rigidbody2D>();
    }
    public void Jump()
    {
        if (_isOnPlatform == true)
            _rigidbody.AddForce(Vector2.up * _jumpForce, ForceMode2D.Impulse);
    }
    private void OnCollisionEnter2D(Collision2D collision)
    {
        GameObject collisionObject = collision.gameObject;
        if(collisionObject.transform.parent != null)
        {
            if (collisionObject.transform.parent.TryGetComponent(out Platform
platform))
                platform.StopMovement();
        }
        if (collisionObject.CompareTag("PlatformWall"))
            Dead?.Invoke();
```

```
        else if(collisionObject.CompareTag("Platform"))
        {
            collisionObject.tag = "Untagged";
            Landed?.Invoke();
        }
    }
} using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class SceneLoader : MonoBehaviour
{
    public void ReloadScene()
    {
        string currentSceneName = SceneManager.GetActiveScene().name;
        SceneManager.LoadScene(currentSceneName);
    }
}
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
public class ScoreUI : MonoBehaviour
{
    private TextMeshProUGUI _field;
    private int _score = 0;
```

```
    private void Awake()
    {
        _field = GetComponent<TextMeshProUGUI>();
    }
    private void Start()
    {
        _field.text = _score.ToString();
    }
    public void IncreaseScore()
    {
        _score += 1;
        _field.text = _score.ToString();
    }
}
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«Інформаційна технологія створення гри Weenfaster у жанрі платформер»

Виконав: студент 5-го курсу, групи 2КН-23м
спеціальності 122 – «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)

Молдаванов В.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН

Озеранський В. С.
(прізвище та ініціали)

«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

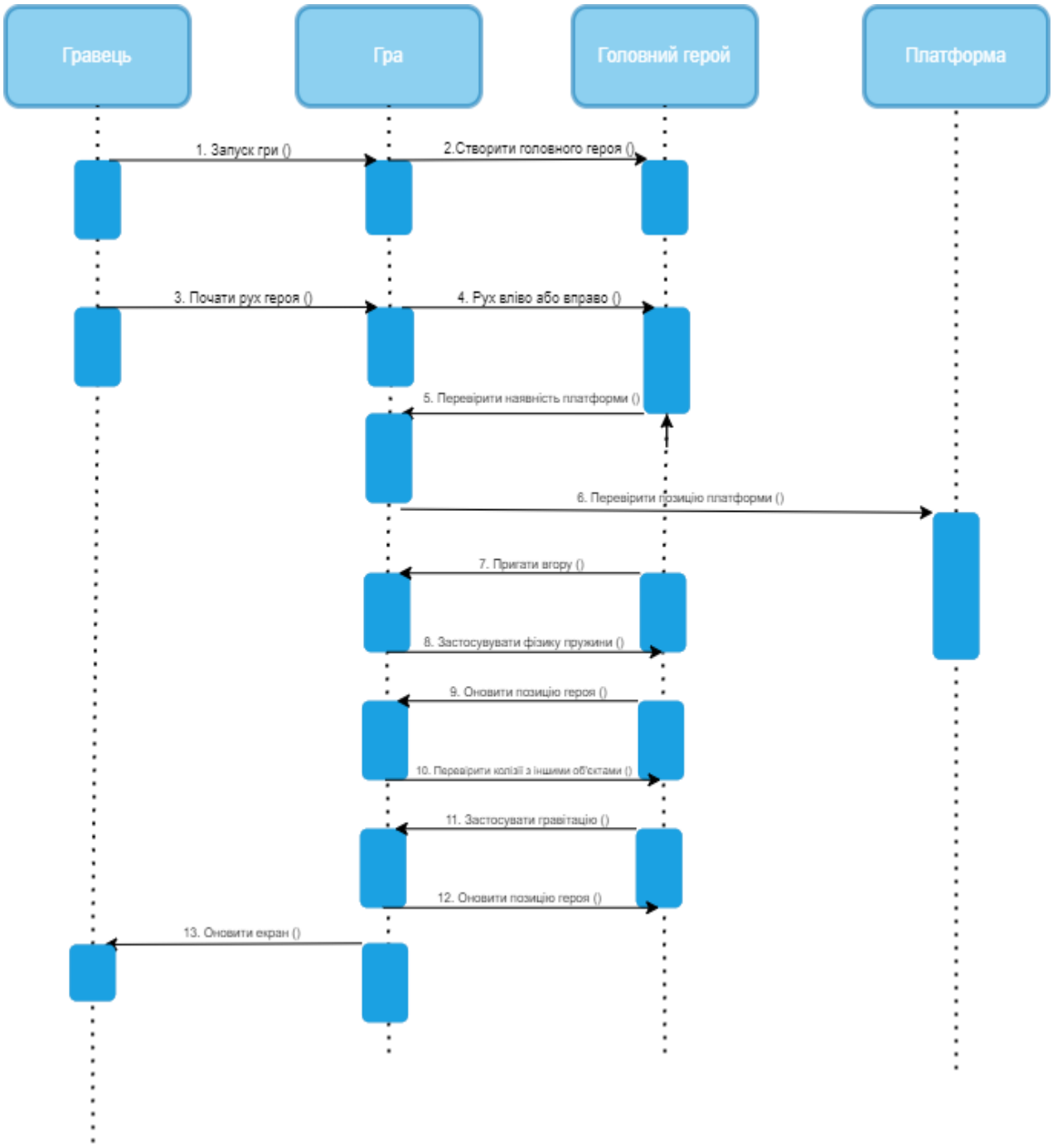


Рисунок В.1 – Діаграма послідовності

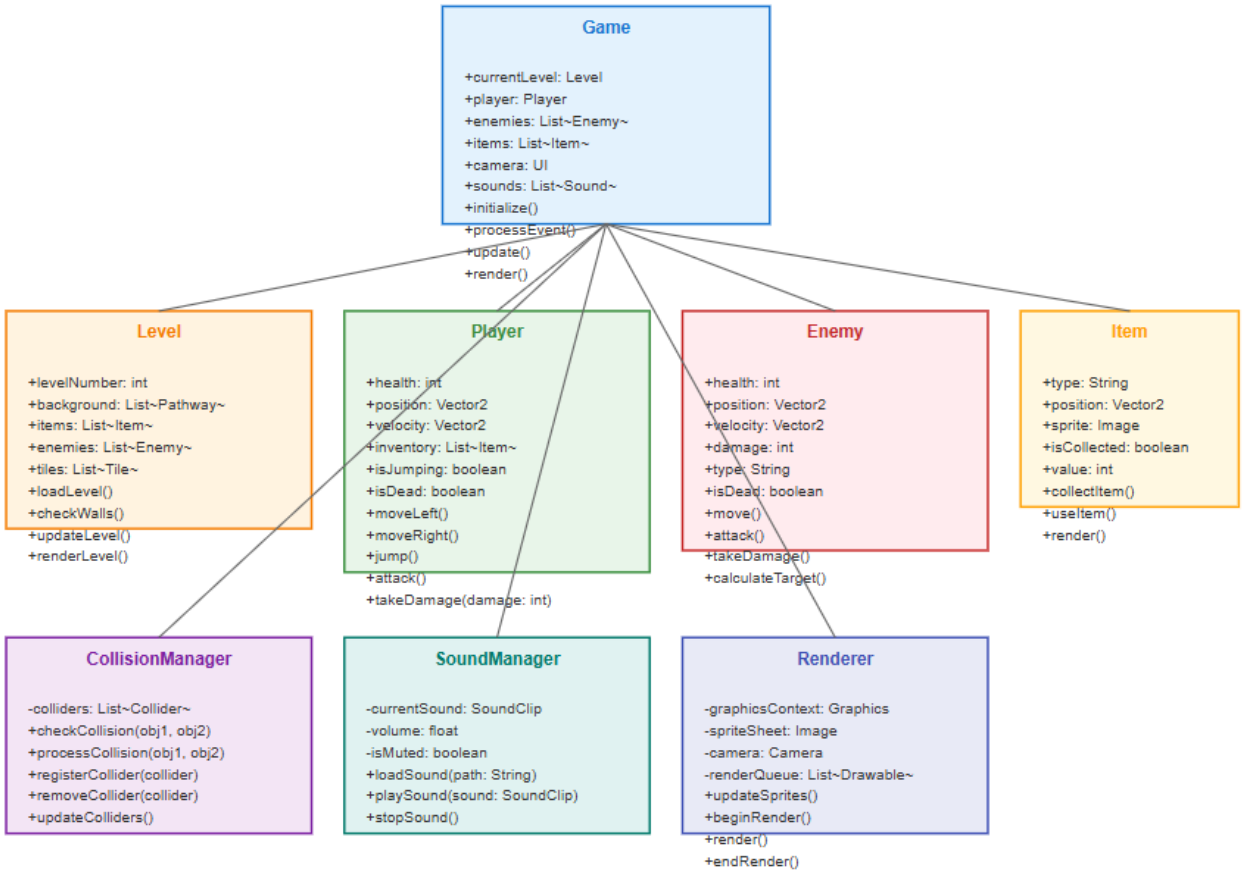


Рисунок В.2 – Діаграма класів

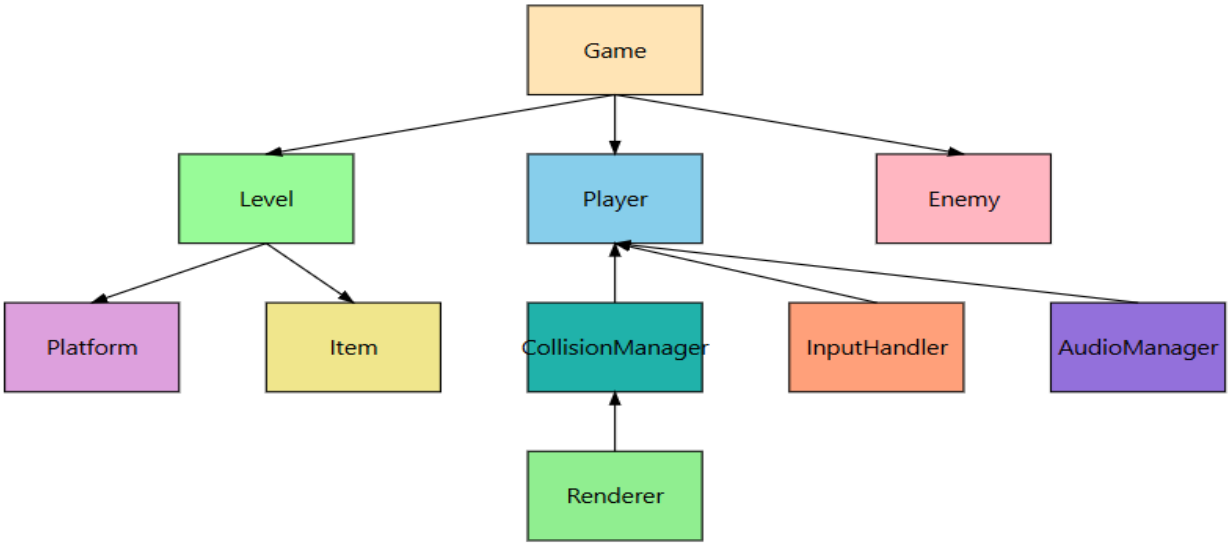


Рисунок В.3 – Архітектура системи

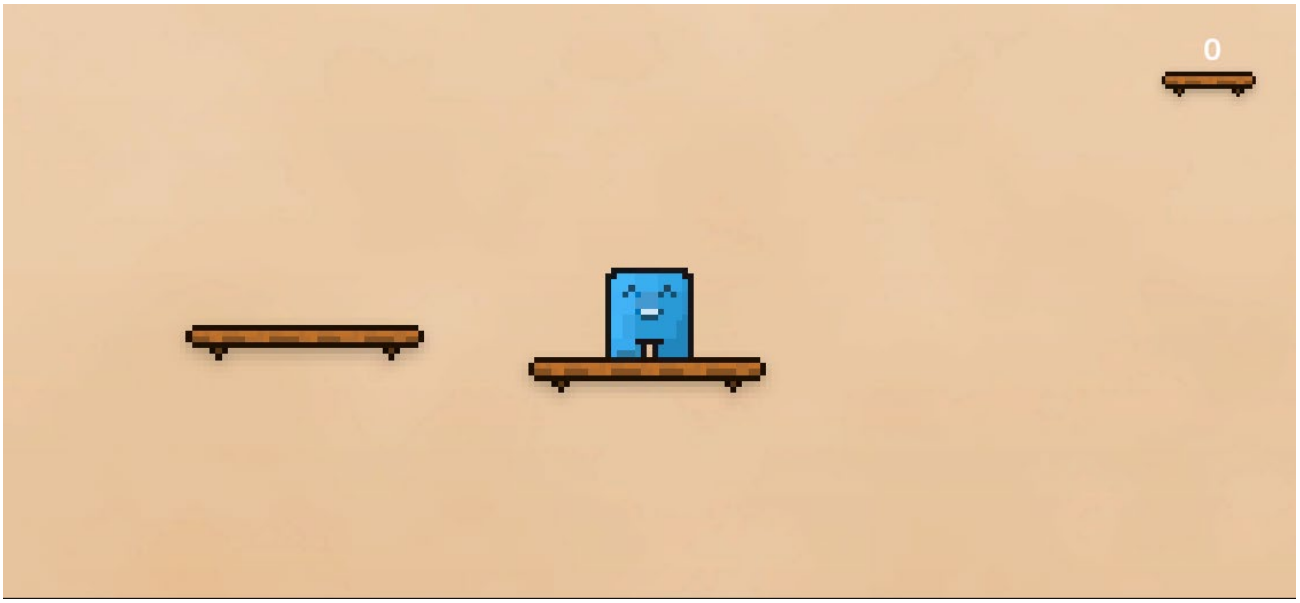


Рисунок В.4 – Ігровий процес

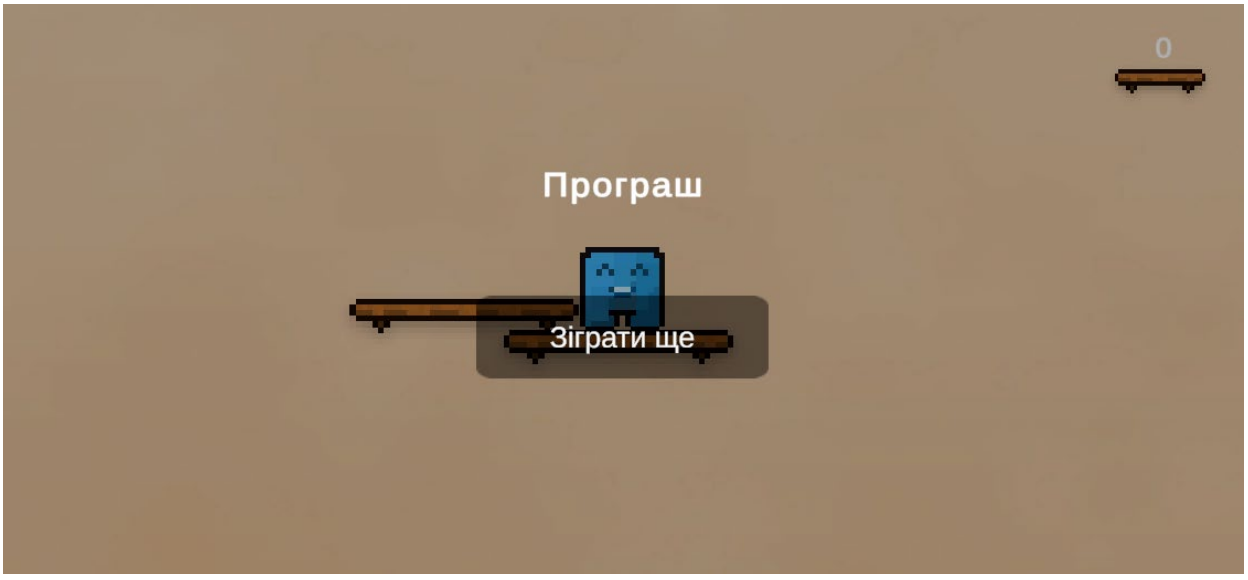
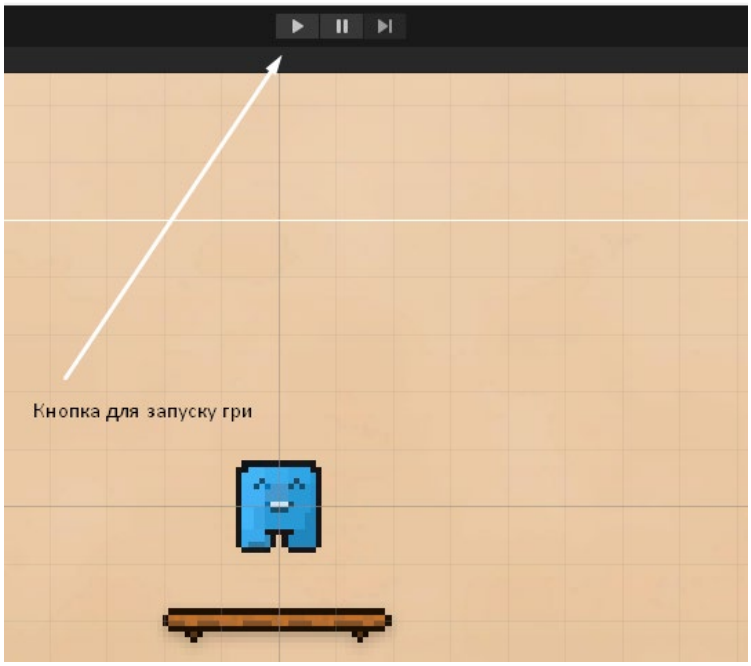


Рисунок В.5 – Завершення гри

Додаток Г (довідниковий)
Інструкція користувача

Для початку потрібно запустити гру це можна зробити в середовищі Unity.



Після чого ми отримаємо можливість до гри.

Метою гри є перепригнути найбільшу кількість платформ. Підрахунок перепригнутих платформ знаходиться з правого кутка. Коли персонаж зачіплює платформи або не встигає перепригнути гра закінчується. І з’являється меню в якому можна нажати на кнопку зіграти ще щоб спробувати ще.