КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації інформаційних технологій

Кафедра інформаційних технологій

(назва випускової кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

Комп'ютерна гра в жанрі Rogue-like на основі ігрового рушія Unity. Ч.2. Геймдизайн, аудіовізуальна частина і інтерфейс

Голик Євгеній Сергійович

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., доцент Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «**Комп'ютерна гра в жанрі Rogue-like на основі ігрового рушія Unity. Ч.2. Геймдизайн, аудіовізуальна частина і інтерфейс**»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідеї, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Голик Євгеній Сергійович
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-22с
Керівник Рябчун Ю.В.
(прізвище та ініціали)
Доктор філософії
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Доля О.В.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет:	Автоматизації інформаційних технологій
Випускова кафедра:	Інформаційних технологій
Освітній ступінь:	Бакалавр
Спеціальність:	Комп’ютерні науки
Освітня програма:	Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТ

Тетяна ГОНЧАРЕНКО

_____

„___”_____2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

	Голику Євгенію Сергійовичу

1. Тема роботи **Комп’ютерна гра в жанрі Rogue-like на основі ігрового рушія Unity. Ч.2. Геймдизайн, аудіовізуальна частина і інтерфейс.**

Затверджена наказом ректора КНУБА №235/23/25 від «14» лютого 2025 року

2. Керівник роботи	Рябчун Юлія Володимирівна, PhD

3. Строк подання Здобувачем роботи до захисту_____

4. Зміст пояснювальної записки за розділами:

Р.1	Аналіз предметної області та постановка задачі.
Р.2	Планування і проєктування
Р.3	Результати створення активів проєкту
Р.4	Онлайн дистриб’юція і маркетинг

5. Графічний матеріал за розділами:

Р.1 6 рисунків

Р.2 12 рисунків, 1 таблиця

Р.3 29 рисунків

Р.4 3 рисунки, 4 таблиці

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1. Аналіз предметної області та постановка задачі.	20.11.2024
Розділ 2	09.03.2025
Розділ 3	21.04.2025
Розділ 4	10.05.2025
Остаточне оформлення роботи	16.05.2025
Направлення роботи для перевірки на плагіат	25.05.2025
Попередній захист роботи на випусковій кафедрі	26.05.2025
Направлення роботи на рецензування	26.05.2025

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1	Рябчун Ю.В.,доц.каф.ІТ	20.11.24	
Розділ 2	Рябчун Ю.В.,доц.каф.ІТ	09.03.25	
Розділ 3	Рябчун Ю.В.,доц.каф.ІТ	21.04.25	
Розділ 4	Рябчун Ю.В.,доц.каф.ІТ	10.05.25	

8. Дата видачі завдання листопад 2025 р.

Зав. кафедри			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Керівники			Рябчун Ю.В.
	(підпис)		(прізвище та ініціали)
Здобувач			Голик Є.С.
	(підпис)		(прізвище та ініціали)

АНОТАЦІЯ

Голик Є.С. Комп'ютерна гра в жанрі Rogue-like на основі ігрового рушія Unity. Ч.2. Геймдизайн, аудіовізуальна частина і інтерфейс.

Атестаційна випускна робота бакалавра за спеціальністю 122 «Комп’ютерні науки», освітня програма «Інформаційні управляючі системи та технології». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Дана робота присвячена розробці гри в жанрі Rogue-like за допомогою ігрового двигуна Unity. Основна увага зосереджена на створенні динамічного ігрового процесу для розваги та утримання уваги гравця з допомогою різних механік, ворогів і випадкових сценаріїв.

У роботі розглядаються етапи проектування, реалізації та тестування гри. В рамках проекту створено прототип гри, що демонструє адаптивний ігролад і динамічну взаємодію гравця з середовищем. Результати можуть бути використані для подальшої побудови повноцінної гри або для адаптації у інших проектах.

Розробка має потенціал для подальшого розвитку та може бути цікавою для гравців і інших незалежних розробників-ентузіастів, що шукають нові ідеї для створення ігор.

Робота викладена на 75 сторінках.

Ключові слова: гра, жанр, Rogue-like, Unity, НПС (неігровий персонаж), механіки, ігровий процес, геймплей, спрайт, аудіо, інтерфейс.

SUMMARY

Holyk Y.S. A computer game in the Rogue-like genre based on the Unity game engine. Ч.2. Game design, audiovisual part and interface.

Bachelor's thesis in the specialty 122 “Computer Science”, educational program “Information Management Systems and Technologies.” - Kyiv National University of Construction and Architecture.

This work is devoted to the development of a game in the Rogue-like genre using the Unity game engine. The main focus is on creating a dynamic gameplay to entertain and keep the player's attention through various mechanics, enemies, and random scenarios.

The paper discusses the stages of game design, implementation and testing. As part of the project, a prototype game was created that demonstrates adaptive gameplay and dynamic interaction between the player and the environment. The results can be used for further development of a full-fledged game or for adaptation in other projects.

The development has the potential for further development and may be of interest to players and other independent enthusiast developers looking for new ideas for creating games.

The work is presented on 75 pages.

Keywords: game, genre, Rogue-like, Unity, NPC (non-player character), mechanics, gameplay, gameplay, sprite, audio, interface.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Unity – платформа для розробки.

Інді (англ. Indie, Independent) – незалежні розробники.

Інтелектуальна власність (IP, англ. Intellectual Property) – результат творчої діяльності, що має правову охорону.

НПС (англ. Non Player Character) – неігровий персонаж.

Спрайт – растрове зображення, яке рухається незалежно від фону.

Штучний інтелект (ШІ, англ. Artificial Intelligence) – програмні алгоритми, що імітують поведінку людини або іншого розумного агента.

Геймдизайн – розробка ігрового процесу, проектування персонажів, сюжетів та завдань.

Rogue-like – жанр відеоігор із процедурною генерацією рівнів, покроковою механікою та постійною смертю персонажа.

Rogue-lite – піджанр rogue-like із спрощеними механіками та елементами метапрогресу.

Ігролад, геймплей (англ. Gameplay) – взаємодія гравця з грою, сукупність правил та механік.

Кількість кадрів за секунду (FPS, англ. Frames Per Second) – показник плавності зображення у грі.

Користувацький інтерфейс (UI, англ. User Interface) – елементи взаємодії користувача з грою.

Інформаційна панель (HUD, англ. Heads-Up Display) – відображення ігрової інформації на екрані (здоров’я, ресурси, карта тощо).

Процедурна генерація – автоматичне створення контенту (рівнів, об’єктів) за допомогою алгоритмів.

Постійна смерть (Perma-death) – механіка гри, при якій смерть персонажа означає втрату всього прогресу.

Неігровий персонаж (NPC, англ. Non-Player Character) – персонаж, контрольований грою.

Штучний інтелект (AI, англ. Artificial Intelligence) – алгоритми для керування поведінкою персонажів.

Інтерфейс програмування додатків (API, англ. Application Programming Interface) – набір інструментів для розробки програмного забезпечення.

Набір засобів розробки (SDK, англ. Software Development Kit) – комплект інструментів для створення програм на певній платформі.

Користувацький інтерфейс і досвід користувача (UI/UX) – дизайн та зручність взаємодії користувача з програмою чи грою.

ВСТУП

Розваги існують з найдавніших часів історії людства і однією з найпоширеніших є різного виду ігри. Від кидальних кісток з помітками, до досвідів у віртуальній та змішаних реальностях – неймовірні метаморфози пережили ігри з розвитком технологій. В 1980-х роках двома програмістами Майклом Тоєм та Ґленном Вічманом була розроблена гра “Rogue” для операційної системи Unix. Гравець виконує роль авантюриста в підземеллі з незліченною кількістю монстрів і скарбів. Мета полягає в тому, щоб пробратися на нижній рівень, отримати амулет Єндора й піднятись на поверхню. Кожен рівень складався з із сітки з трьох кімнат на три кімнати з періодичними тупиковими коридорами. На відміну від більшості пригодницьких ігор того часу, макет підземелля та розміщення об’єктів у ньому генеруються випадковим чином, що і відрізняло її від конкурентів.

Саме Rogue вважається першою у світі грі в жанрі rogue-like. Завдяки своїм інноваційним механікам, Rogue започаткувала цілий жанр ігор, який згодом отримав назву rogue-like. Основними характеристиками таких ігор стали процедурна генерація рівнів, постійна смерть персонажа (permadeath), покроковий геймплей і висока складність, що стимулює гравця до стратегічного мислення. З плином часу ці принципи трансформувалися, породивши піджанр — rogue-lite, який зберігає деякі ключові елементи оригінального жанру, але спрощує або змінює інші, роблячи гру більш доступною широкому загалу.

Rogue-lite ігри зазвичай включають поступовий прогрес гравця навіть після поразки, використовуючи механіки метапрогресії: покращення навичок, розблокування нових предметів чи можливостей між проходженнями. Завдяки цьому гравець відчуває розвиток і мотивацію продовжувати, навіть у випадку частих невдач. Одним із найуспішніших прикладів сучасного rogue-lite є Hades, в якому поєднуються елементи динамічного бою, сюжетної побудови та покрокового прогресу, що вивело жанр на новий рівень популярності.

З розвитком ігрових технологій, рушіїв (як-от Unity, Unreal Engine) та широкого розповсюдження інді-розробки, жанр rogue-lite став особливо

привабливим для незалежних розробників. Його модульна структура, можливість повторного проходження (replayability) та глибока ігрова механіка дозволяють створювати захопливі проєкти навіть з обмеженими ресурсами.

Метою даної роботи є розробка прототипу комп'ютерної гри в жанрі Rogue-like, за допомогою Unity, що забезпечує захоплюючий ігровий досвід за рахунок використання динамічного ігрового процесу, різноманітних механік і характерних жанрових особливостей .

Завдання дослідження:

1. Аналіз жанрових особливостей ігор типу Rogue-like і виділення ключових елементів ігроладу.
2. Дослідження ринку наявних рішень.
3. Дослідити можливості Unity.
4. Розробити концепт гри.
5. Реалізувати основні ігрові механіки.
6. Провести тестування прототипу.

Об'єкт дослідження – процес розробки комп'ютерної гри в жанрі Rogue-like із застосуванням сучасних інструментів розробки.

Предмет дослідження – методи та інструменти реалізації ігроладу на платформі Unity.

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Ігрова індустрія набула небачених масштабів за останні два десятиріччя і за розмірами бюджетів зрівнялась з комерційним сегментом кіноіндустрії [1].

Проте з розвитком технологій і інтернету, розкіш мати інформацію та можливості для створення власної гри прийшла до пересічних людей. Це призвело до стрімкого розвитку пласті інді ігор [2, 3].

Інді-ігри, створені незалежними розробниками без великих видавництв, стали потужною силою для творчості, інклюзивності та інновацій. Вони створили більш інклюзивне та інноваційне ігрове середовище, зосередившись на мистецтві, розповіді та спільноті. Також причиною стала сильна комерціалізація і корпоратизація серед великих видавців. Акцент почав ставитись на комерційну успішність гри, перехід на конвеєризацію відомих IP, прибирання спірних або складних для пересічного потенційного покупця тем і придушення нових оригінальних ідей задля просування вже готових шаблонів і мінімізації ризику. Це і є основні причини успішності незалежної сцени ігрообства за останні роки. Інді-ігри зробили ігрову індустрію доступнішою. На відміну від дорогих AAA-ігор, орієнтованих на широку аудиторію, інді-ігри в першу чергу зосереджені на творчості та оригінальності [4, 5].

1.1 Постановка та аналіз проблеми

Одним з найбільш створюваних жанрі інді ігор є Rogue-like. Такою популярністю він користується через його гнучкість до жанровості, реіграбельності і фокусу на геймплеї [6]. Невибагливість до графічної частини дозволяє створювати ігри навіть в обмежених технічних обставинах. Варто зауважити, що це не означає поганий вигляд гри, багато представників жанру мають унікальну візуальну складову.

Проте створення гри в цьому жанрі ставить перед розробниками проблеми:

1. Випадковість оточення: розробка алгоритмів створення унікальних сценаріїв в процесі гри.

2. Унікальні механіки: впровадження унікальних ігроладних рішень для вирізнення в жанровій купі.
3. Реіграбельність: гра повинна залишатись цікавою після поразки або перемоги, мати можливість тримати увагу гравця.
4. Приємний зовнішній вигляд: гра може мати мінімальний вигляд, проте всі її частини мають виглядати лаконічно.

Аналіз проблеми: хоча сучасні реалії і роблять доступними розробку ігор, та це призводить до великої кількості низькоякісного продукту та збільшення рівня конкуренції. Rogue-like ігри потребують креативних рішень і їх реалізації для привернення уваги вибагливої аудиторії.

Розв'язання цих проблем включатиме:

- розробку алгоритмів створення випадкових сценаріїв;
- дизайн ігрових механік і ворогів;
- впровадження аудіовізуальної складової;

Вирішення цих задач дозволить створити унікальний продукт.

Отже, завданням в кваліфікаційній роботі буде розробити однокористувацьку гру в жанрі Rogue-like, яка забезпечує випадкові кімнати, що збільшуються з кожним рівнем та елементи виживання. Гравець має пройти через серію підземель із ворогами, збираючи унікальні предмети, розвиваючи персонажа і приймаючи рішення, що вплинуть на подальший прогрес.

1.2 Огляд літератури та готових існуючих готових рішень

Жанр roguelike вирізняється процедурною генерацією рівнів, постійною смертю персонажа (permadeath), високою складністю та елементами випадковості, що забезпечують унікальний досвід кожного проходження. Unity є оптимальним вибором для реалізації такої гри завдяки своїй гнучкості, широкому інструментарію та підтримці 2D і 3D графіки. Unity дозволяє ефективно реалізовувати процедурну генерацію, систему збережень, бойові механіки, а також має розвинену екосистему Asset Store, що прискорює розробку roguelike проєктів.

Розробка комп'ютерних ігор у жанрі rogue-like на основі ігрового рушія Unity є популярним напрямком у сучасній індустрії ігор. Цей жанр відзначається процедурною генерацією рівнів, високою складністю та постійною смертю персонажа, що робить кожну гру унікальною.

Платформа Unity Learn пропонує безкоштовні проекти та курси, які допомагають опанувати розробку ігор, включаючи жанр roguelike. Ці матеріали охоплюють різні аспекти розробки, від базових концепцій до просунутих технік [1].

При розробці roguelike-ігор важливо враховувати оптимізацію графіки. В [1] детально описує, як зменшити кількість викликів рендерингу та покращити продуктивність гри.

Проаналізувавши літературу [7 - 11], яка пропонує готові рішення для проблем, що можуть виникнути під час розробки гри, відзначили:

- *Офіційна документація Unity*: детальний посібник з використання Unity, який охоплює всі аспекти розробки ігор [7].
- *"Roguelike Tutorial - Complete Roguelike Tutorial"*: покроковий посібник англійською мовою зі створення roguelike-гри [8].
- *"Procedural Dungeon Generation Algorithm":* стаття, яка описує алгоритми процедурної генерації підземель для roguelike-ігор [9].
- *"r/RoguelikeDev":* форумна тема, де розробники діляться досвідом та порадами щодо створення roguelike-ігор [10].
- *"How to start making pixelart":* стаття, що описує основні кроки створення піксельної графіки [11].

1.3 Аналіз готових рішень

Для підвищення якості проекту, можна проаналізувати інші подібні продукти на ринку. Для цього співставимо дані з різних джерел і проаналізуємо 3 найбільш повторювані найкращі результати [12 - 15].

Після співставлення маємо такі проекти:

- The binding of Isaac
- Hades
- Spelunky 2

The binding of Isaac – комп'ютерна гра (рис.1.1), розроблена Едмундом Макміланом і випущена в 2011 році, перевипущена і розширена до версії Rebirth в 2014 сувмісно з видавцем Nicalis [16].

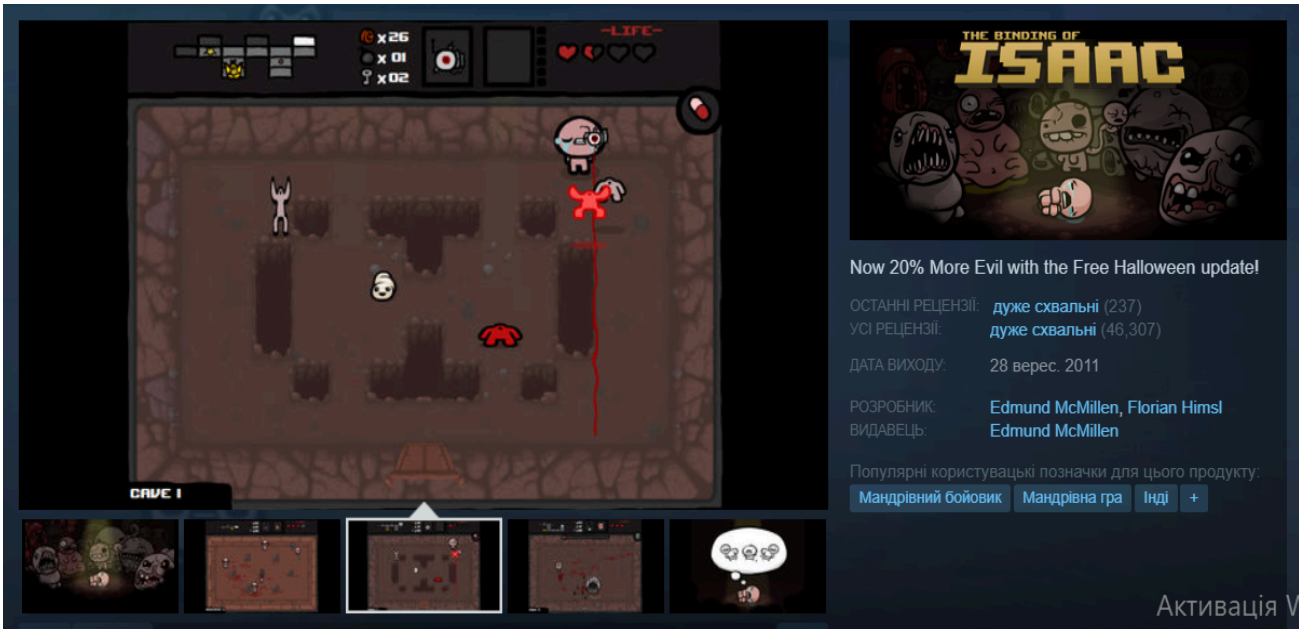


Рисунок 1.1. Фрагмент гри The binding of Isaac

Сеттинг: гравець бере на себе роль хлопчика на ім'я Ісаак, якого хоче вбити власна мати, і намагається знайти вихід зі свого підвалу. Гра просочена релігійними покликаннями, темами насилля і жорстокості. Сюжет подається під час завантажень і проходженням на різні кінцівки, для яких мають бути виконані певні умови.

Ігролад: гра поділена на рівні-поверхи, які в свій час розділені на випадково перемішані кімнати, що були створені вручну, але за рахунок їх великої кількості, їх повторюваність починаєш помічати далеко не одразу. Кімнати наповнені ворогами та перепонами, обидва мають шанс на випадіння ресурсу або предмету під час знищення. Коли гравець заходить в небезпечну кімнату, вихід перекривається, поки пастка не буде розброєна або вороги вбиті. Після виконання

умови, гравець отримує нагороду. Для переходу на новий поверх необхідно знайті і в бити боса рівня. Кожен поверх містить в собі одну гарантовану кімнату зі скрабом, де можна знайти корисний предмет. Також поверха мають інші опціональні поверхи, секретні кімнати, магазини, бібліотеки, кімнати з угодами. Персонаж має здоров'я, у вигляді різних сердечок з різними властивостями, один активний предмет, пасивні предмети і один брелок. Є можливість розблокування нових персонажів через виконання певних умов.

Переваги:

- Чудова реіграбельність: предмети утворюють унікальні синергії, що можуть кардинально змінити навіть схоже проходження.
- Подача сюжету: подання сюжету маленькими клаптиками спонукає гравця проходити гру далі, вигадувати теорії, думати.
- Контент: гра має неймовірну кількість наповнення.

Недоліки:

- Зовнішний вигляд: гра має специфічний зовнішній вигляд, який може бути огидний деяким користувачам.
- Монотонність: гра може починати набридати після чисельних невдач під час спроб досягнення деяких кінцівок.

Hades [17] – гра, розроблена і випущена американською студією Supergiant Games (рис.1.2). Була випущена 2018 році в моделі раннього доступу, повний реліз відбувся в 2020 році.

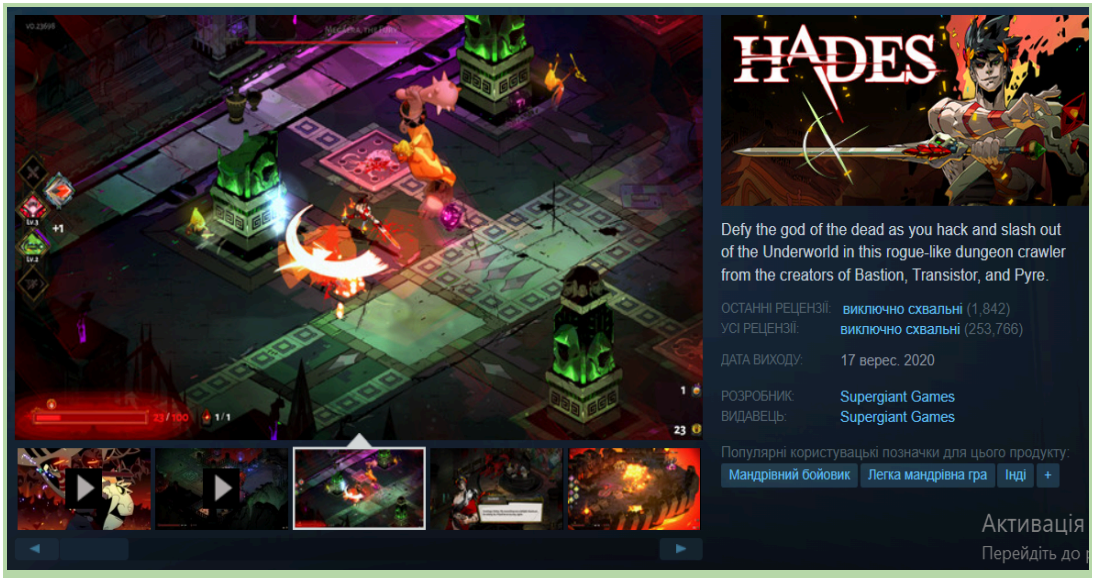


Рисунок 1.2. Фрагмент гри Hades

Сеттинг: сюжет основано на давньогрецькій міфології. Головний герой Загрей, син Аїда, намагається втекти з царства мертвих і дістатися Олімпу. На його шляху йому допомагають боги, які посилають Загрею дари.

Ігролад: гра так само базується на системі поверхів та кімнат, однак в Hades вони більші. Дари богів виступають в ролі різних виді покращень, додаючи грацю шкоду по площі, кругові атаки, тощо. Гравець має змогу вибирати різну зброю, що суттєво впливає на стиль гри. Камера зафіксована на гравці і Загрей завжди знаходиться в центрі екрану. Між різними поверхами знаходяться боси. В цьому випадку вони кожен раз однакові, що дозволяє гравцеві легше адаптуватись до їх атак і робить проходження легшими і швидшими. Після смерті, герой потрапляє в палац Аїда, де гравець може покращити зброю або характеристики персонажа за ресурси, що здобуваються під час забігів. Також в палаці доступні діалоги з другорядними персонажами, та можливість поліпшити з ними відносини, подарувавши подарунок, що теж знаходиться під час гри.

Переваги:

- Зовнішній вигляд: гра виглядає чудово, має плавні анімації.
- Різноманітність: наповнення контентом, відмінним від битв, дозволяє зробити інколи так необхідну паузу.

Недоліки:

- Побудова світу: гра має на диво мало греків, тема грецького міфу абсолютно не досліджена в цій грі. Грецький міф являє собою просто гарну обгортку [18].

Spelunky 2 – гра-платформер (рис.1.3), що вийшла у 2020 році [19]. Розробником є студії Mossmouth та Blitworks. Гра є сиквелом Spelunky.

Сеттинг: гравець бере контроль над Аною, дочкою головного героя першої гри, яка прилетіла на Місяць, щоб знайти свої батьків. Героїня подорожує смертельними печерами, повними пасток і ворогів.

Ігролад: гравець пересувається продерно згенерованими печерами. Гра представляє собою класичну формулу Rogue-like, перенесену в двовимірну вертикальну площину, наповнену секретами, завідома створеними сценаріями та пастками. Як і в Hades, в грі присутня хаб локація, де можна зустріти інших персонажів. Гра бере формулу першої частини, виправляє недоліки і додає нового.

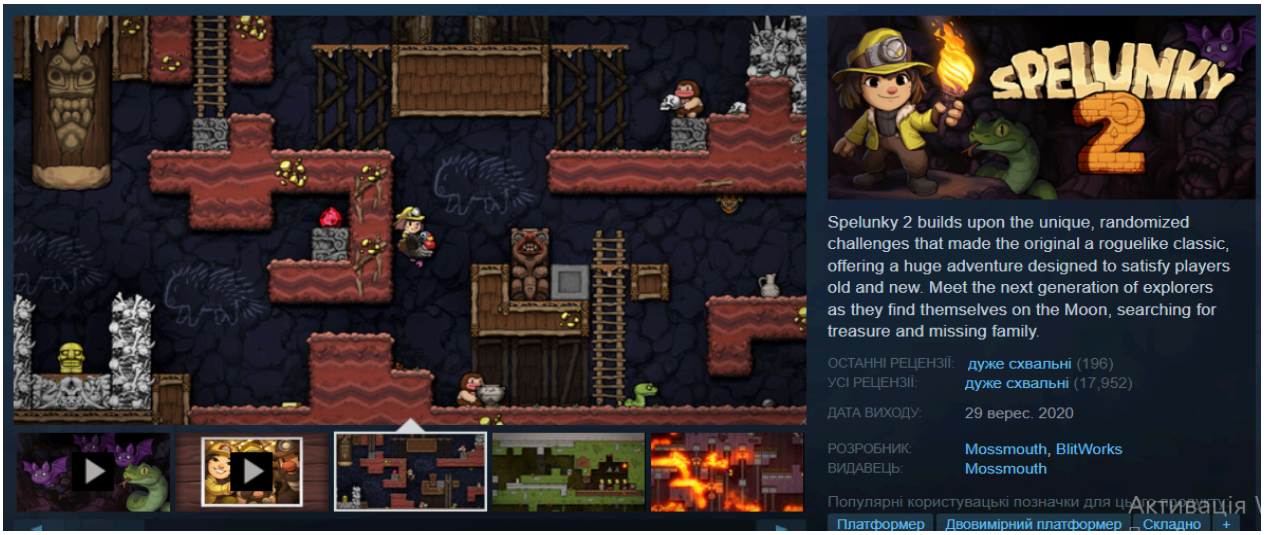


Рисунок 1.3.Фрагмент гри Spelunky 2

Переваги:

- Приємний зовнішній вигляд: гра підійде будь-якій аудиторії.
- Керування: платформінг врі відчувається чудово.
- Різноманітність: гра не дає нудьгувати, кожна спроба відчувається сильно відмінною від попередньої.

Недоліки:

- Складність: Spelunky далеко не проста гра.
- Сиквел: гра є сиквелом, хоч і не зобов'язує грати в попередника.

Для дослідження графічного і аудіального стилю, характерного іграм минулої епохи, було обрано Final Fantasy 1 (рис 1.4).[20]

Рисунок 1.4. Фрагмент гри Final Fantasy

Final Fantasy (яп. ファイナルファンタジー, фаїнару фантадзі укр. Остання фантазія) — відеогра в жанрі JRPG. Була випущена компанією Square (нині Square Enix) 1987 року та поклала початок провідному бренду Square — серії Final Fantasy. Уперше була розроблена на платформу Nintendo Family computer. Final Fantasy I стала однією з найвпливовіших ранніх JRPG і зіграла одну з провідних ролей у формуванні й популяризації жанру.

В даному випадку, треба звернути увагу на візуальний стиль. Гравець бачить світ зверху. Використовуючи різні візуальні прийоми, художники створюють відчуття ізометрії, хоча гра працює лише у двох вимірах. Враховуючи обмеження

картриджу, анімації мають, в кращому випадку, 2 кадри. На рисунку 1.5 зображено повні анімації всіх персонажів.

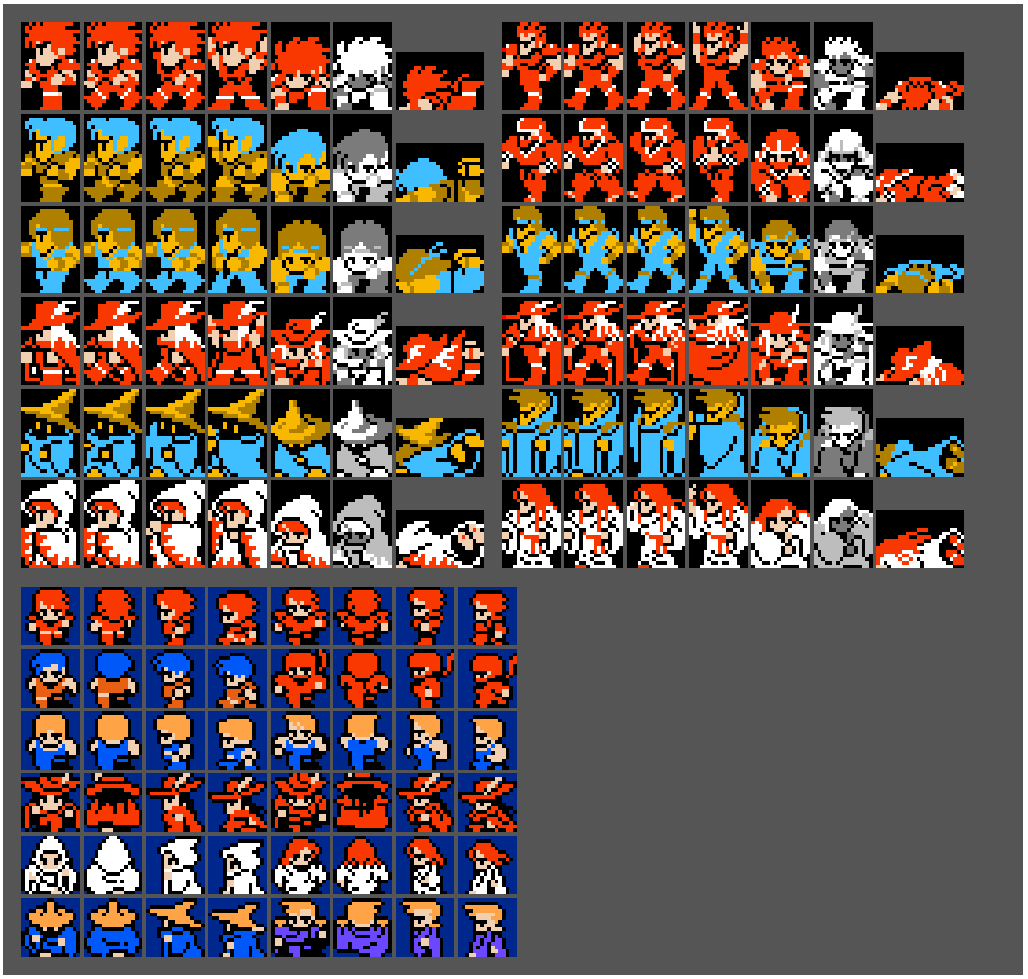


Рисунок 1.5. Таблиця анімацій персонажів

Музичний супровід був створений з урахуванням обмежень консолей. Приставка мала підтримку 4 каналів з різними формами хвиль: 2 квадратних, 1 трикутна і 1 шумова. Всі музичні композиції короткі і починаються знову, після закінчення.

1.4 Дерево цілей

Дерево цілей – це наочне зображення підпорядкованості та взаємозв’язку цілей, що демонструє розподіл генеральної мети на підцілі та окремі завдання [20].

Мета роботи – розробка прототипу комп’ютерної гри в жанрі Rogue-like. Вона передбачає:

- Високий рівень технічної реалізації та стабільності.
- Забезпечує реіграбельність, завдяки процедурному генеруванню рівнів і різноманітності сценаріїв.
- Відповідає сучасним вимогам продуктивності.

Зважаючи на вказані вище тези, отримуємо дерево цілей на рис. 1.6.

Рисунок.1.6 Дерево цілей роботи

Для кращого розуміння розглянемо його детальніше:

1. Базова архітектура – база гри. Вона охоплює основні елементи проекту, взаємодії і механіки.

1.1. Сценарій гри – сценарій охоплює весь процес від моменту запуску гри, до її завершення.

1.2. До ігрових об’єктів відносяться:

1.2.1. Гравець.

1.2.2. Неігрові персонажі.

1.2.3. Предмети (зброя, ресурси т.п.).

1.2.4. Рівні.

2. Реалізація механік – створення правил, принципів і систем, які визначають взаємодію гравця з ігровим середовищем, персонажами, об’єктами. Вони формують основну частину ігроладу.

2.1. Генерація рівнів – важлива частина випадковості жанру. В цьому випадку генерація буде процедурна.

2.2. Бойова система – механіка, що визначає взаємодію гравця з ворогами. Варіації можливостей атаки, захисту, отримання шкоди.

3. Тестування й оптимізація – доведення гри до її стабільної та коректної роботи.

3.1. Виправлення помилок – знаходження і виправлення неправильної роботи механічної частини гри.

3.2. Оптимізація продуктивності – можливе зменшення навантаження гри на систему. Очищення коду і ресурсів від непотребу, компресія.

1.5 Вимоги та особливості проектування системи

Проектування - це етап життєвого циклу розроблення програмних систем, наступний після інженерії вимог [21]. Завданням цього етапу є перетворення побажань замовників системи, які ми подали як моделі вимог, у проектні рішення, що забезпечують здійснення згаданих побажань у формі відповідної системи програмування.

Вимоги до системи:

1. Функціональні вимоги
a. Система збереження прогресу.
b. Механіка досягнень.
2. Інтерфейс
a. Ергономічність.
b. Можливість налаштувань різних елементів гри.
3. Розширюваність
a. Гнучке проектування для можливість легкого впровадження оновлень.
4. Адаптивність
a. Оптимізація для моніторів з різним розширенням.
5. Продуктивність

а. Враховуючи візуальний стиль, можна зробити висновок, що суттєвого навантаження на систему не буде.

Розділ 2. ПЛАНУВАННЯ І ПРОЄКТУВАННЯ

Продукт являє собою комп’ютерну гру в жанрі Rogue-like з особливостями характерними жанру і випадковою генерацією рівнів. Шляхом дослідження і прийняття рішень був створений ескізний проект.

2.1 Розробка концепції гри: сеттинг, сюжет, стиль

Події відбуваються в вигаданому фентезійному світі.

Королівство жило мирно, поки не з’явився таємничий острів і не окутав землі туманом. Мирні жителі почали помирати від постійних набігів монстрів. Король посилає експедицію, та вона потрапляє у шторм і потерпає невдачі. Гравець бере на себе роль вцілілого солдату, якому треба розгадати загадку таємничого острову і врятувати королівство. Після фінального боса, герой дізнається, що острів з’являється кожні 300 років, і увесь цей час бився з своїми солдатами і минулою експедицією, а всі монстри – навіювання ядовитого туману острова, який харчується загубленими душами. Після чого, втрачає здоровий глузд і сам стає головним босом. Таким чином повторні проходження гри пояснюється логічно.

Гра стилізована під ретро-ігри часів третього покоління домашніх консолей[27]. Кольорова гама була обрана монохромна, в сірих кольорах. Музичний супровід також розроблений і стилізований під представників даної епохи. Консолі того часу мали п’ять музикальних каналів з різними хвилями: дві квадратні, одна трикутна, шумова і канал для семплів. Композиції для гри були написані використовуючи дані обмеження.

2.2 Ескізний проєкт

Ескізний проєкт – початковий етап створення проекту, на якому визначаються основні характеристики продукту [22]. Результатами ескізного проекту буде:

1. Загальний опис.

2. Опис вимог.
3. Макет інтерфейсу.
4. Макет ігрової сцени.

2.2 Загальний опис і опис вимог

Загальні дані:

- Робоча назва проєкту: Гра
- Сеттинг: темне фентезі
- Жанр: action RPG Rogue-like
- Основні механіки: система ресурсів, предметів, процедурна генерація, жаб'ячий зір

Технічні вимоги:

- Поведінка камери: гравець в центрі
- Ігровий рушій: Unity
- Мова програмування: C#
- Графіка: двовимірний піксельарт
- Формат звуку: .wav .ogg
- Формат растрових зображень: .png
- Розширення гри: 640x360
- Розширення спрайтів: 32x32
- Кожна дія має мати звуковий ефект
- Необхідні фахівці: програмісти мова C#, художники, аудіорежиссери, тестувальники
- Оцінка часу: 3 місяці

Вимоги до інтерфейсу меню:

1. Кнопка "Продовжити гру"
2. Кнопка "Почати гру"
3. Кнопка "Налаштування"
   3.1. Кнопка "Звук"

3.1.1. Налаштування гучності музики

3.1.2. Налаштування гучності ефектів

3.1.3. Налаштування гучності загальної

3.2. Кнопка “Графіка”

3.2.1. Налаштування розширення

3.2.2. Налаштування яскравості

3.3. Кнопка “Мова”

3.3.1. Вибір мови

3.4. Кнопка “Назад”

4. Вихід

На основі вимог до інтерфейсу меню будуємо діаграму прецедентів [23] зображену на рисунку 2.1.

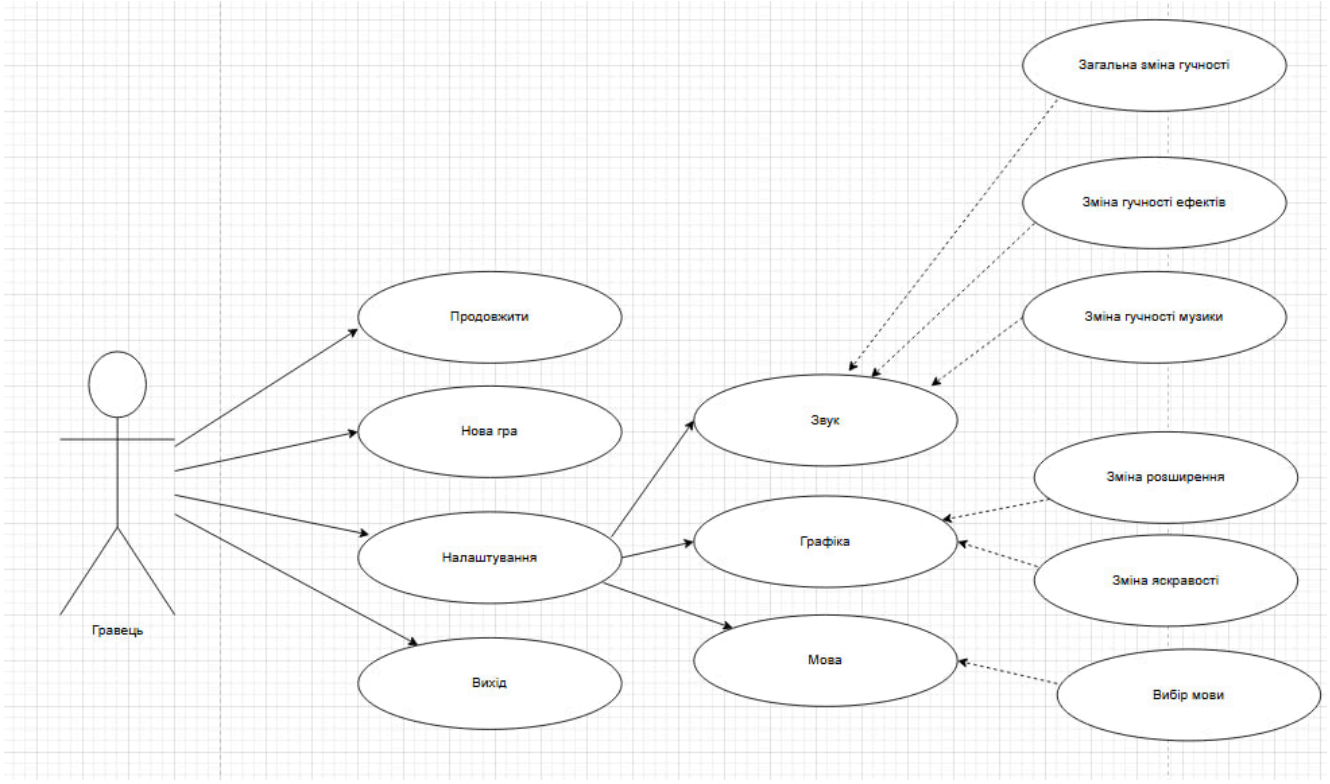


Рис 2.1. Діаграма використання головного меню

Розглянемо більш детально кожен варіант в таблиці 2.1.

Таблиця 2.1. Специфікації прецедентів

Складова	Опис	Передумови	Основний потік подій	Постумови
Продовжити	Запускає попередньо збережену гру	Є збережена гра	Система запускає збережену гру	Не визначені
Нова гра	Створює порожнє збереження і запускає	Не визначені	Система запукає нову гру	Не визначені
Налаштування	Користувач потрапляє в меню налаштувань	Натиснення кнопки налаштування	Користувача переносить в меню налаштувань	Не визначені
Звук	Користувач потрапляє в меню налаштування звуку	Натиснення кнопки звук	Користувача переносить меню налаштувань звуком	Не визначені
Зміна гучності ефектів	Користувач змінює гучність ефектів	Рух повзунка гучності	Змінюється гучність ефектів	Не визначені
Зміна гучності музики	Користувач змінює гучність музики	Рух повзунка гучності	Змінюється гучність музики	Не визначені
Зміна загальної гучності	Користувач змінює загальну гучніть гри	Рух повзунка гучності	Змінюється гучність ефектів	Не визначені
Графіка	Користувач потрапляє в меню налаштувань графіки	Натиснення кнопки графіка	Користувача переносить в меню налаштувань графікою	Не визначені
Зміна розширення	Користувач змінює розмір вікна гри	Натиснення кнопки розширення	З’являється випадаючий список з варіантами розширення	Не визначені

Зміна яскравості	Користувач змінює яскравість	Рух повзунка яскравості	Зміна яскравості вікна	Не визначені
Мова	Користувач змінює мову гри	Натиснення кнопки мова	З’являється випадаючий список з мовами	Не визначені
Вихід	Користувач покидає програму	Натиснення кнопки вихід	Закриття програми	Не визначені

Вимоги до інтерфейсу під час гри:

1. Шкала здоров’я
2. Шкала енергії
3. Рівень
4. Щит
5. Сила атаки
6. Кнопка взаємодії

2.3 Макет меню і ігрової сцени

За допомогою графічних редакторів Aseprite та Adobe Photoshop створимо макет головного меню (рис.2.2).

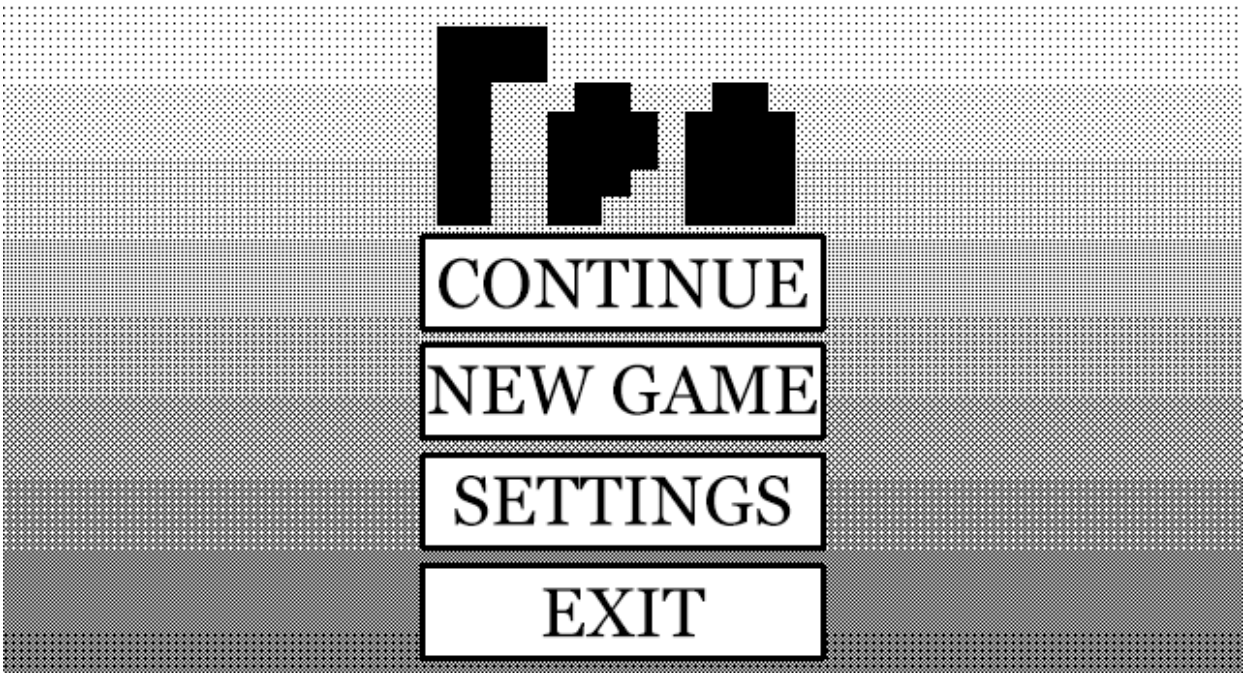


Рисунок 2.2 Макет інтерфейсу головного меню.

Також розробимо макети для екрану налаштувань, кожну з вкладок (рис. 2.3 -2.6).

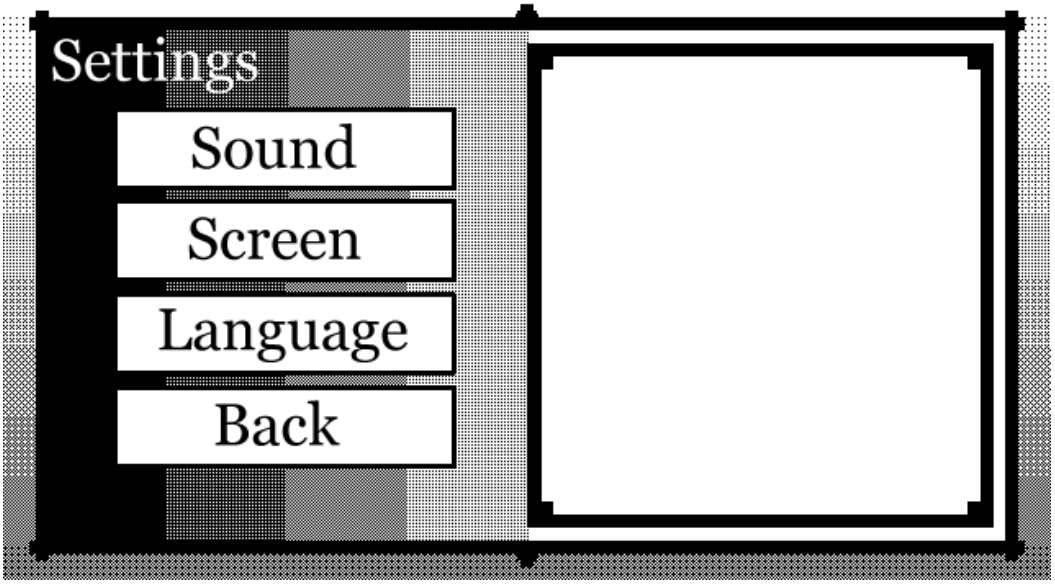


Рисунок 2.3 Вікно налаштувань

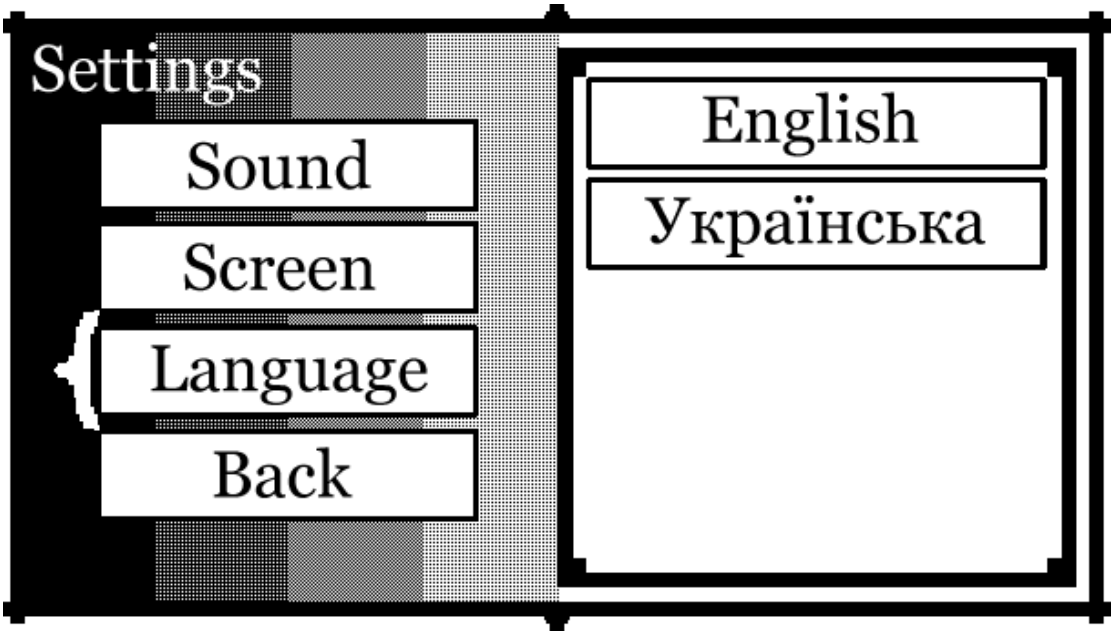


Рисунок 2.4 Вікно налаштувань мови

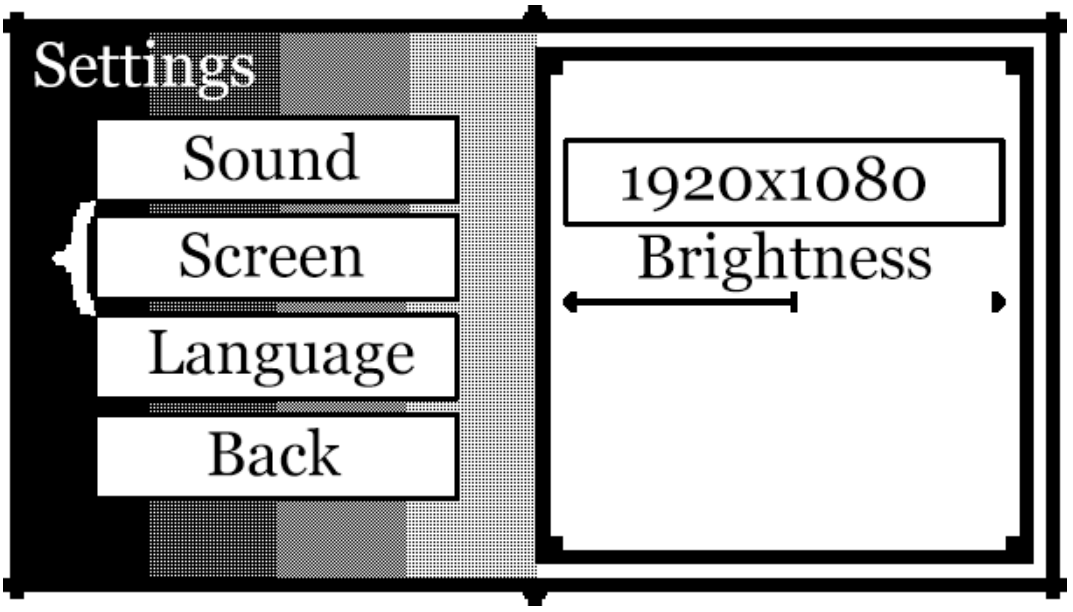


Рисунок 2.5 Вікно налаштувань екрану

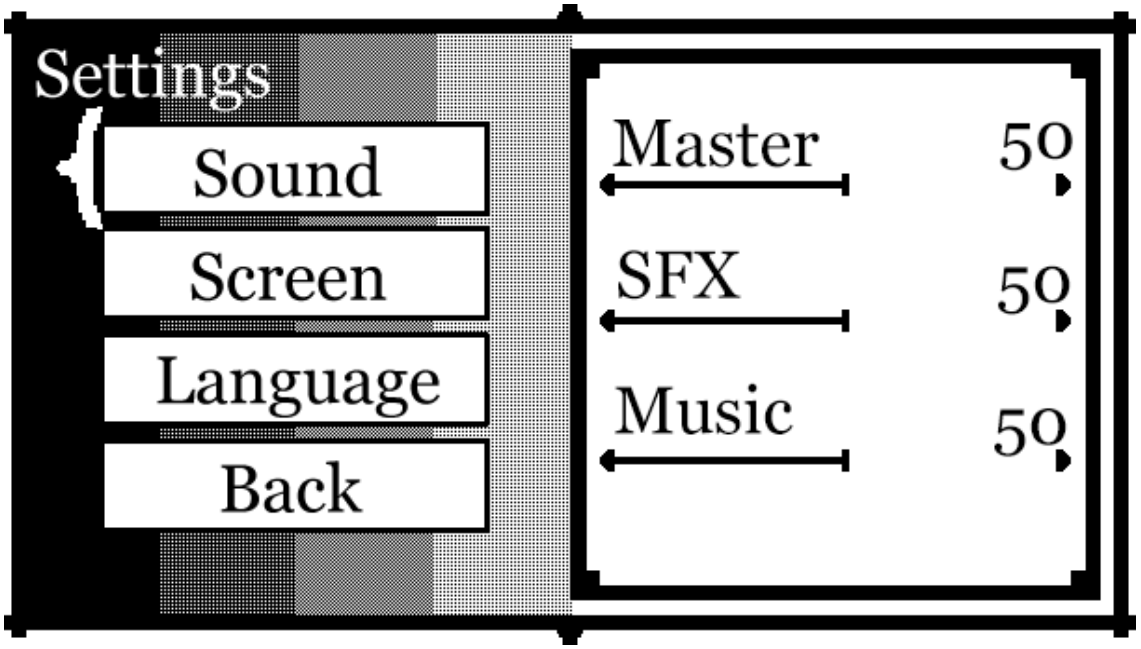


Рисунок 2.6. Вікно налаштувань звуку

Відповідно до вимог будуємо макет інтерфейсу під час гри. Для глибшого розуміння інтерфейс розміщений на макеті вікна гри (рис. 2.7).

Рисунок 2.7 Макет сцени гри з інтерфейсом

Верхня частина представляє собою шкалу здоров’я, статистику персонажа, кількість щита та рівень. Шкала енергії розміщена під самим персонажем для зручності. При підході до предмета з’являється його назва та кнопка взаємодії. Дружні неігрові персонажі мають іконку над головою: діалог, завдання або магазин. В правому нижньому кутку кнопка інвентарю.

2.4 Вибір засобів впровадження

Головним засобом розробки було обрано рушій Unity (рис 2.8).

Unity – кросплатформений ігровий рушій запущений в 2005 році, підходить для розробки дво- і тривимірних застосунків.

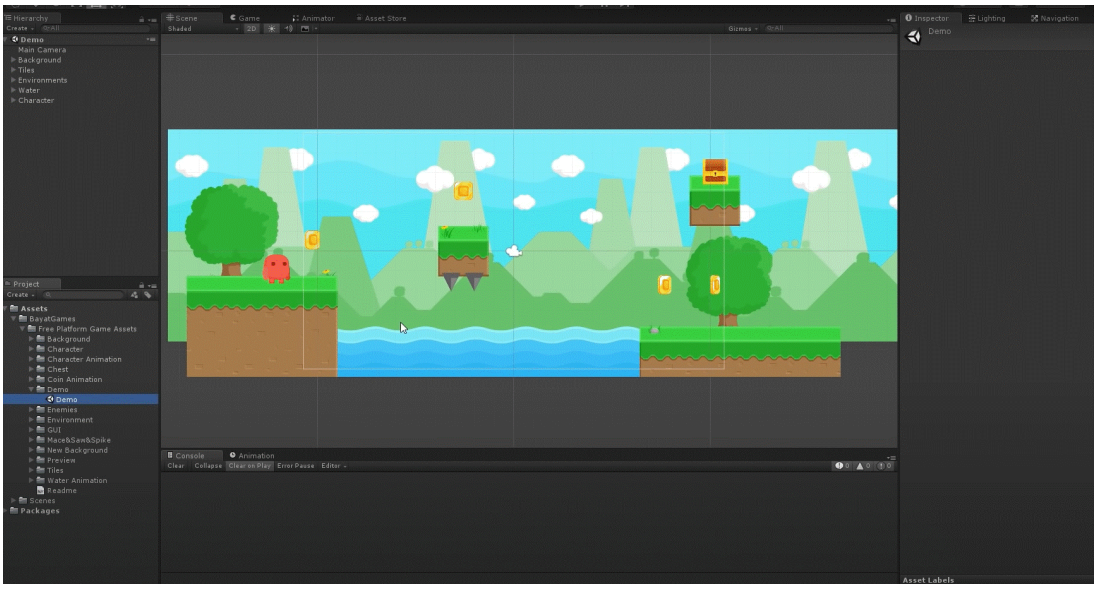


Рисунок 2.8. Робочий екран Unity engine

Причини обрати Unity для розробки гри в жанрі rogue-lite:

- Кросплатформеність
  - Unity дозволяє створювати ігри одразу для декількох платформ, таких як Windows, macOS, Linux, Android, iOS, WebGL, а також консолей. Це дає можливість розширити аудиторію гри без необхідності переписувати код під кожну платформу.
- Широка підтримка 2D та 3D графіки
  - Незалежно від того, чи планується гра з класичним 2D виглядом чи з ізометричною 3D графікою, Unity має готові інструменти та оптимізовані пайплайни для обох випадків.
- Розвинене середовище розробки
  - Unity має інтуїтивно зрозумілий інтерфейс редактора, що дозволяє швидко створювати сцени, налаштовувати об’єкти, анімації, колізії та багато іншого без необхідності занурюватися виключно в код.
- Підтримка C#

- Unity використовує мову C#, яка є досить легкою для вивчення, має чітку синтаксичну структуру, але водночас достатньо потужна для реалізації складних систем, таких як штучний інтелект, менеджери станів, збереження прогресу тощо.
  - Можливість масштабування проєкту
- Навіть якщо гра починається як невеликий інді-проєкт, Unity дозволяє легко масштабувати його, додаючи нові модулі, функціональності, багатокористувацький режим або онлайн-збереження, не змінюючи фундаментальної структури.

Для роботи з графікою вибрано Adobe Photoshop та Aseprite.

Aseprite – програма для роботи з піксельартом і растровою графікою випущена у 2001 році (рис.2.9) [24]. Саме вона буде використовуватися в якості основного графічного редактору.

Переваги Aseprite:

- Спеціалізований для піксель-арту — всі інструменти оптимізовані для створення спрайтів, тайлів та ретро-графіки.
- Зручна робота з анімацією — підтримка покадрової анімації, onion skinning, таймлайну та циклічних анімацій.
- Підтримка експорту у форматах, зручних для ігрових рушіїв — спрайт-листи, PNG, GIF, JSON.
- Інтуїтивно зрозумілий інтерфейс — легкий у використанні як для новачків, так і для досвідчених художників.
- Можливість використання скриптів та плагінів — для автоматизації та розширення функціональності.
- Разова ліцензія — купується один раз, без щомісячної підписки.

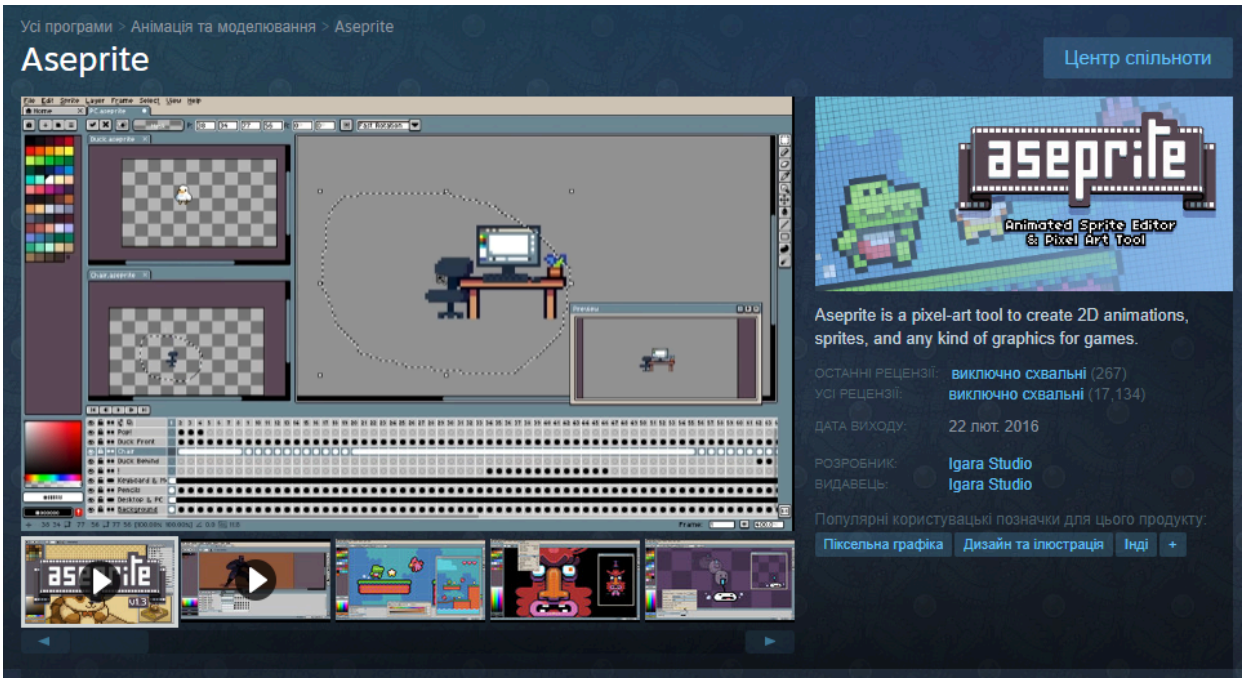


Рисунок 2.9 Aseprite сторінка в Steam

Adobe Photoshop – графічний редактор, розроблений фірмою Adobe. Програми були обрані через свій багатий функціонал і зручність в використанні (рис 2.10) [25].

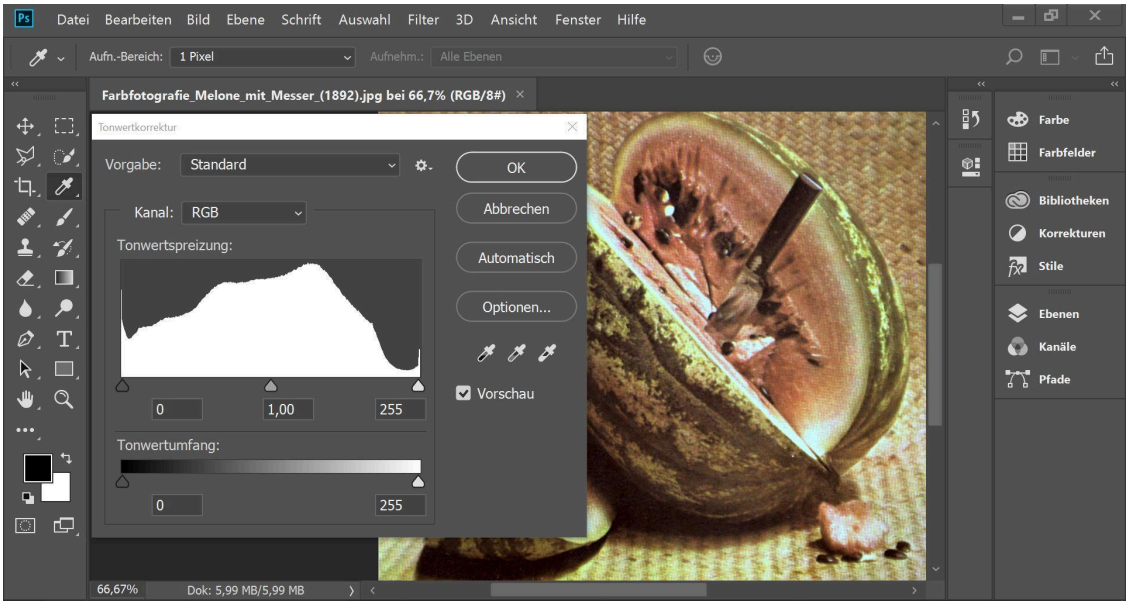


Рисунок 2.10. Робочий екран Adobe Photoshop

Для створення і редагування звукових ефектів використано Reaper [26]. Програма дозволяє записувати і редагувати звуки, музику, MIDI (рис.2.11).

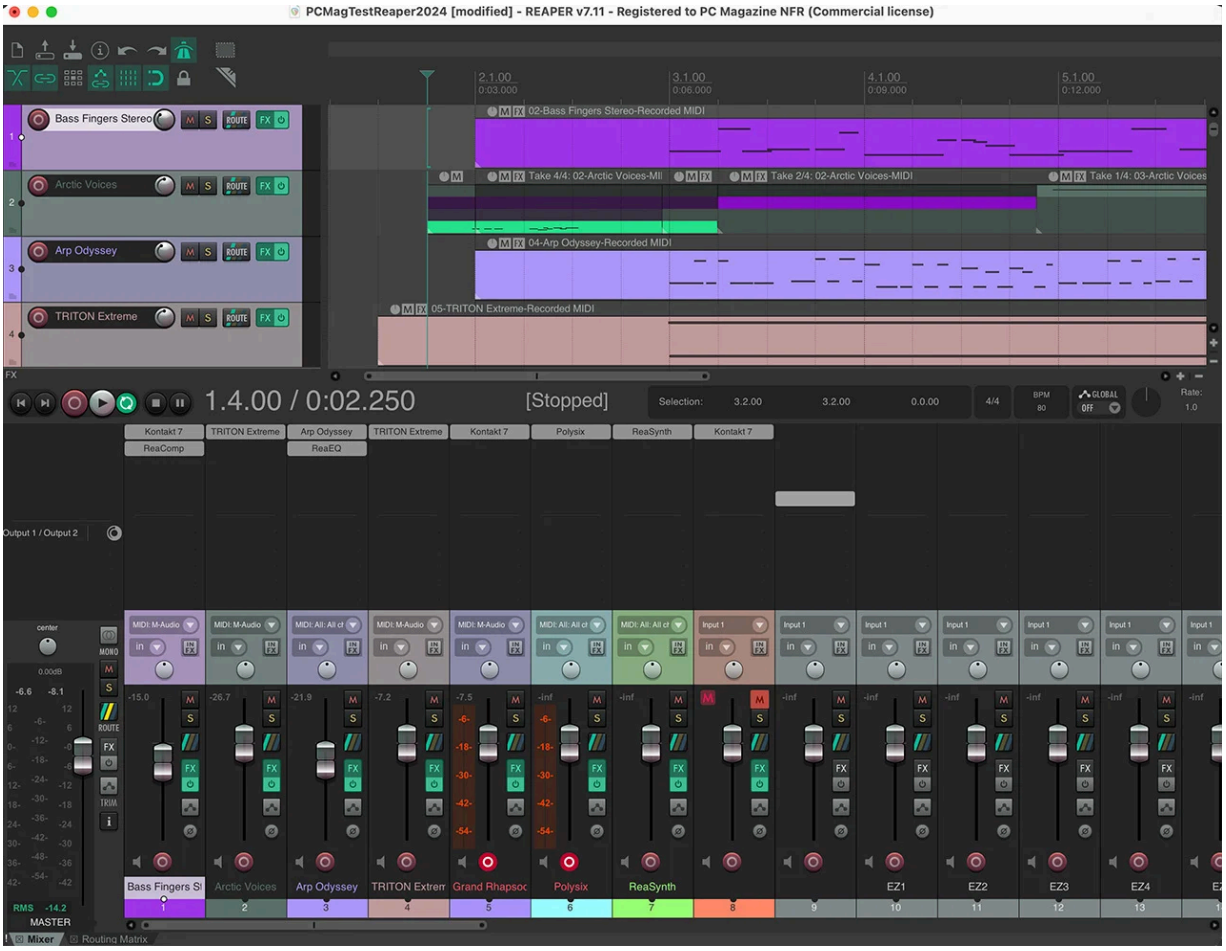


Рисунок 2.11 Інтерфейс програми Reaper

Для створення характерного звучання старих консолей обрано програму Famistudio.[34] FamiStudio — це простий музичний редактор для Nintendo Entertainment System або Famicom. Інтерфейс зображений на рисунку 2.12.

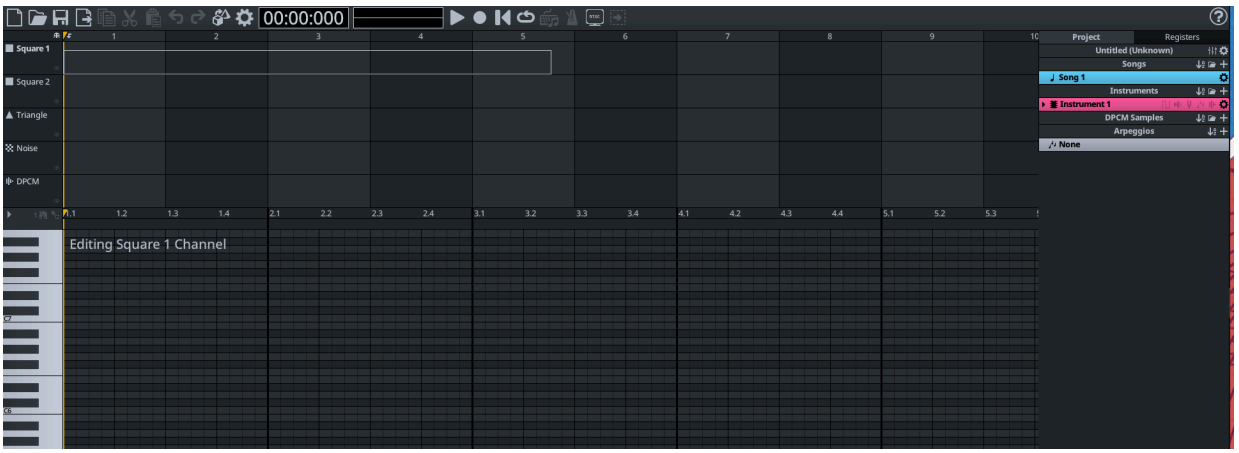


Рисунок 2.12. Інтерфейс програми Famistudio

Розділ 3. РЕЗУЛЬТАТИ СТВОРЕННЯ АКТИВІВ ПРОЄКТУ

Результатом проєкту є гра в стилі rogue-lite. Дана частина сфокусована на графічній і аудіальній частині проєкту. Було створено і реалізовано велику кількість анімацій, текстур, ефектів, музичного супровіду, частину активів на майбутні оновлення. Створений і впроваджений інтерфейс.

3.1. Музичний супровід

Композиції можна послухати по посиланню:
https://soundcloud.com/notholyk/sets/hra

OST1 Endangered Castle

Дана композиція грає в головному меню гри. Назва пояснює зав'язку сюжету – замок в небезпеці. Музика поєднує епічні загравання з середньовічними мотивами. На рисунку 3.1. зображений проект треку.

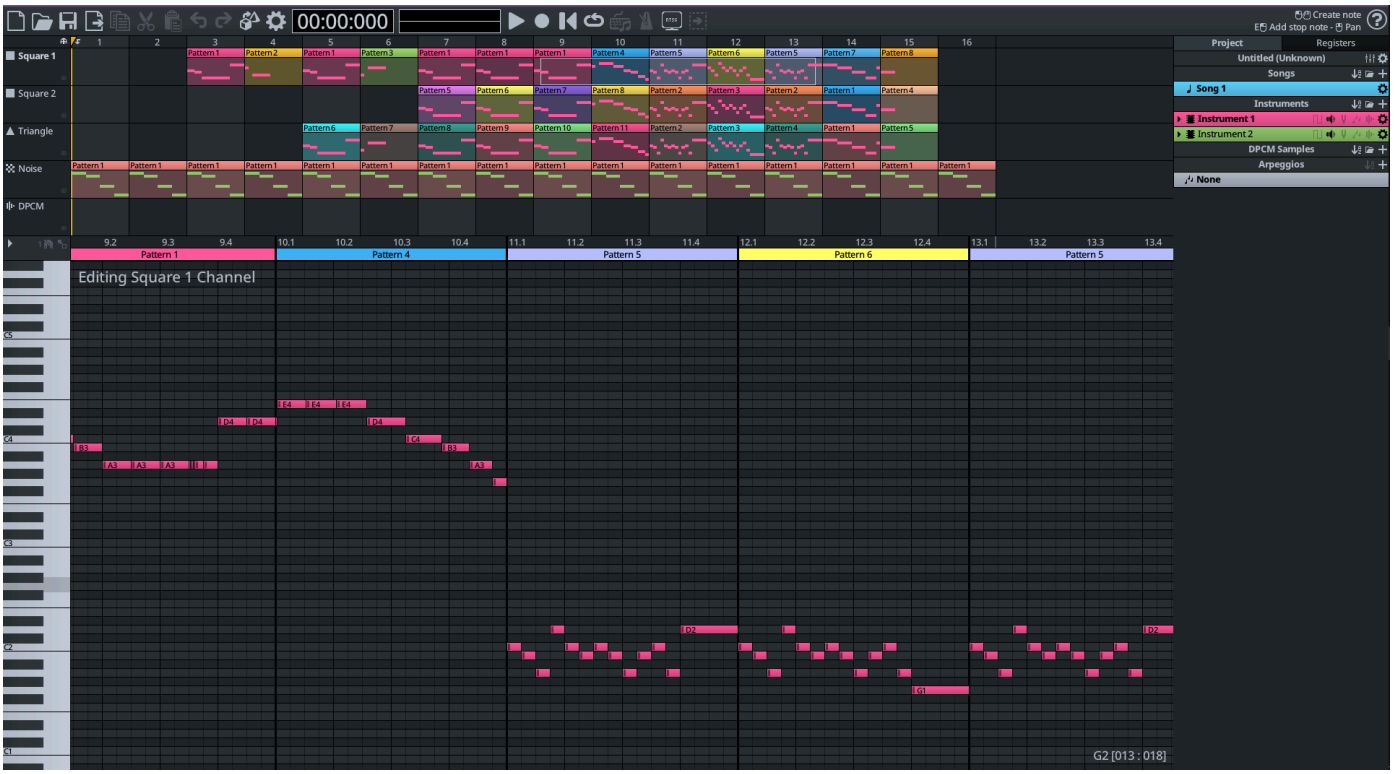


Рисунок 3.1. Проєкт композиції Endangered Castle

OST2 Wuthering

Дана композиція грає на першому рівні. Перший рівень відбувається після шторму на пляжі. Вона поєднує ритмічні хвилі шуму і акцент, нагадуючий азбуку морзе. Трек має нагнітаючий характер. На рисунку 3.2. зображений проект треку.

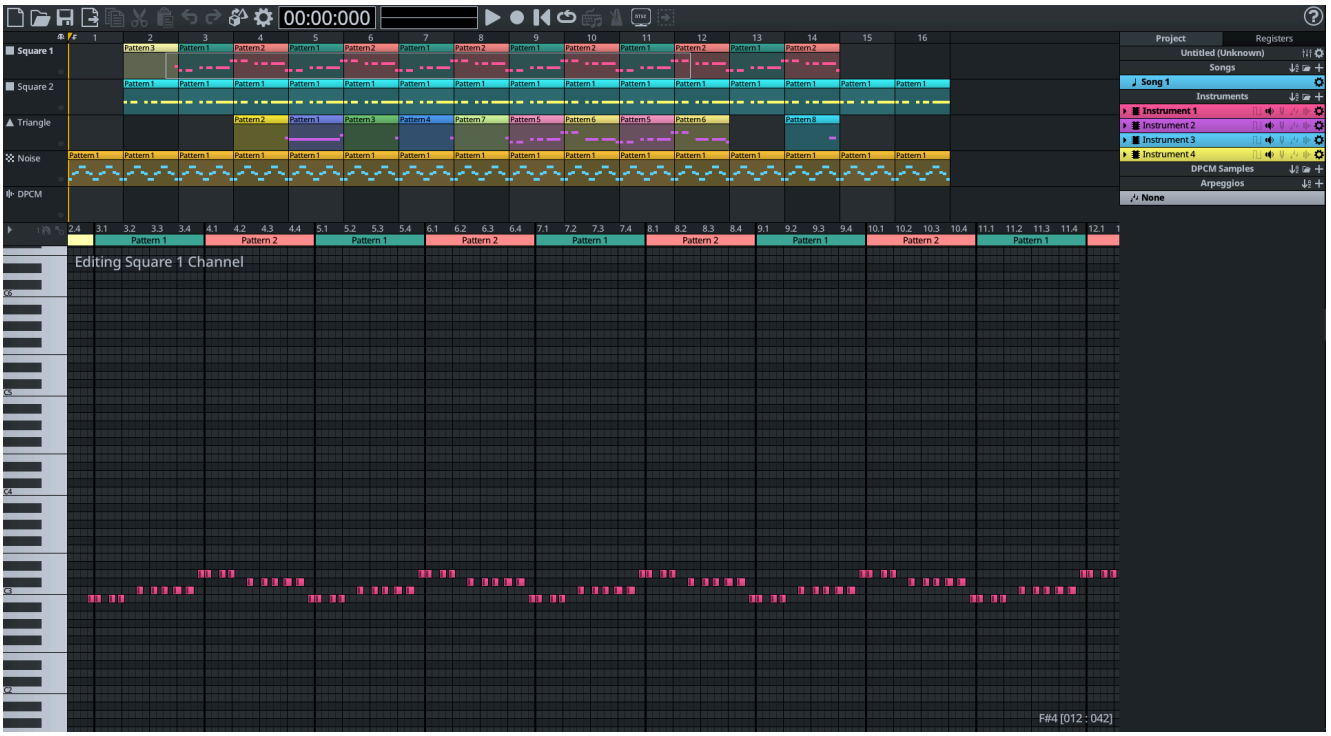


Рисунок 3.2. Проєкт композиції Wuthering

OST3 Darkness

Дана композиція грає на другому рівні, що відбувається в печері. Композиція складається з повторюваних нот і додаткових, що описує однаковість оточення печер і розвилки у них. Друга частина має піднесений характер, символізуючи надію і що кожна печера має вихід. На рисунку 3.3. зображений проект треку.

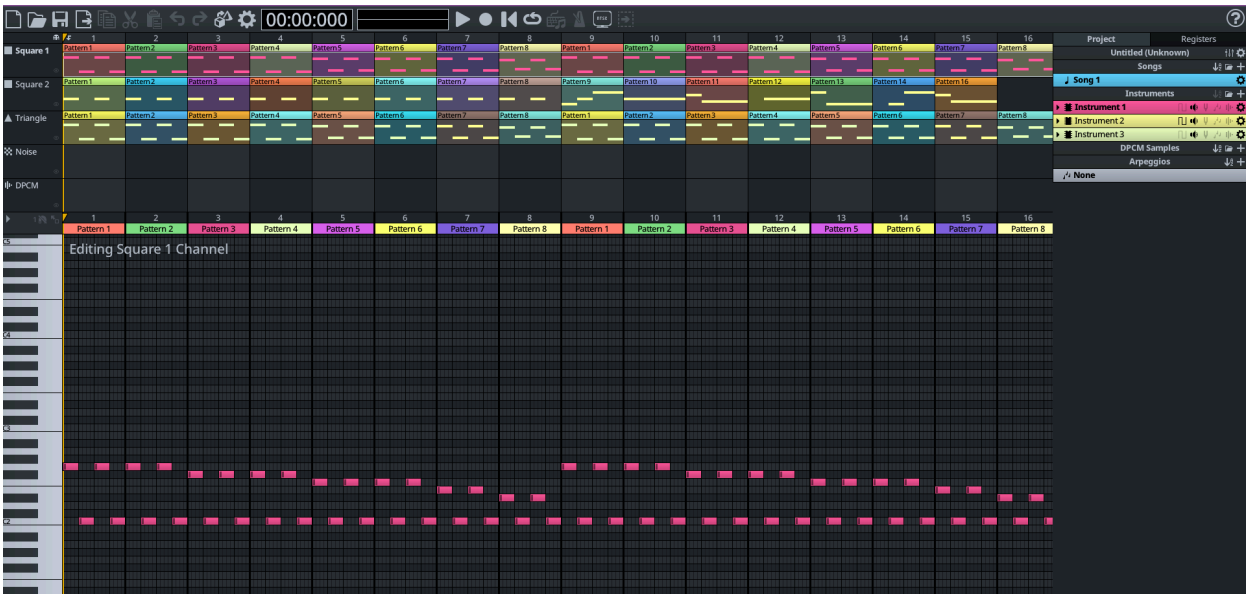


Рисунок 3.3. Проєкт композиції Darkness

OST4 Lost Royalty

Трек починається з інструменту, схожий на фанфар, який грає трохи повз нот. Це відсилає до безумства подій на острові. Проєкт треку зображено на рисунку 3.4.

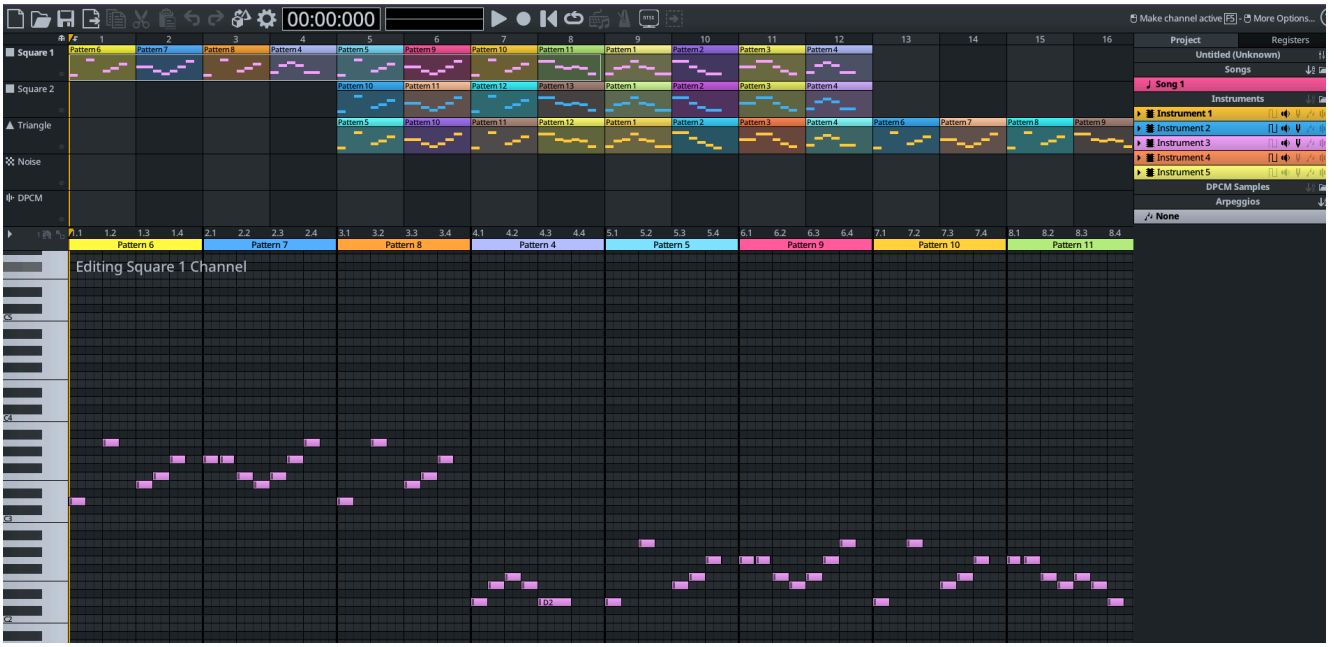


Рисунок 3.4. Проєкт композиції Lost Royalty

3.2. Візуальна частина

Графічна частина представлена в монохромній кольоровій гамі, що можна побачити на рисунку 3.5.

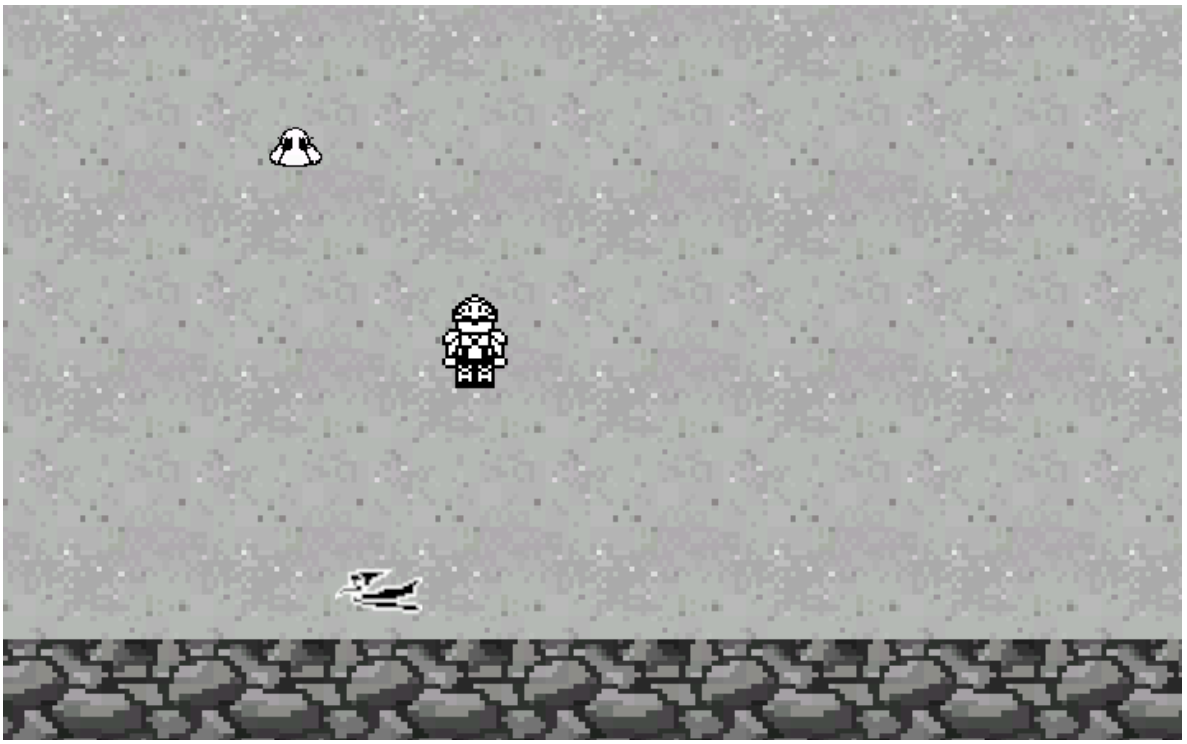


Рисунок 3.5 Візуальний стиль гри

Демо гри відбувається в печері і замку, для рівнів було створено характерні текстури матеріалів. Матеріали поверхонь створювалися таким чином, щоб не було помітно явного стику між тайлами. Відсутність явно виражених деталей допомагає уникнути сконцентруванню уваги на елементі, що теж сприяє замиленню сприйняття. Приклади зображені на рисунках 3.6 - 3.9.

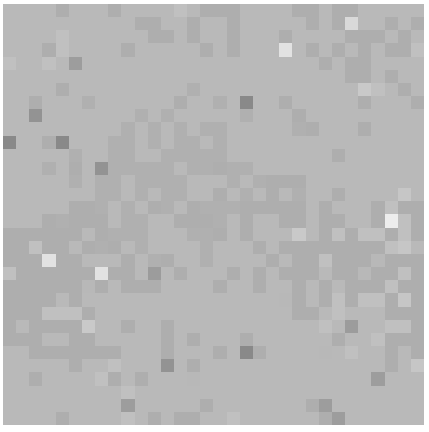


Рисунок 3.6. Підлога першого рівня

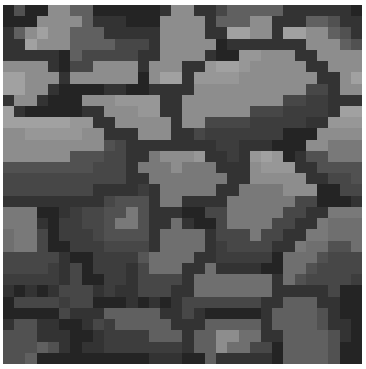


Рисунок 3.7. Стіна першого рівня

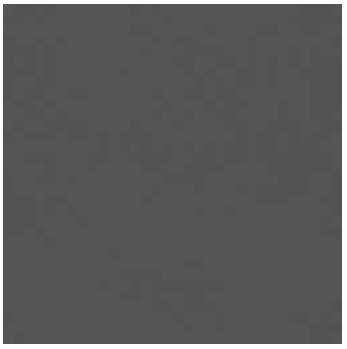


Рисунок 3.8. Підлога другого рівня

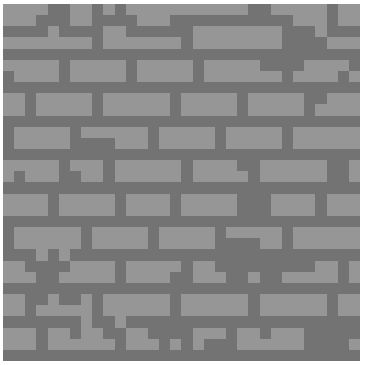


Рисунок 3.9. Стіна другого рівня

Щоб оточення не здавалося пустим, були додані інтерактивні і статичні об’єкти. Приклади зображено на рисунку 3.10 - 3.11.

Рисунок 3.10. Предмети оточення

Рисунок 3.11. Предмети оточення в грі

Образ головного героя базувався середньовічних лицарах. Персонаж має характерну пластинчатий обладунок і шолом з піднятим забралом. У персонажа присутні анімації очікування і ходьби, приклад наведено на рисунку 3.12.

Рисунок 3.12. Головний герой і його анімації

Перехід між анімаціями створений за допомогою аніматора Unity(рис.3.13).

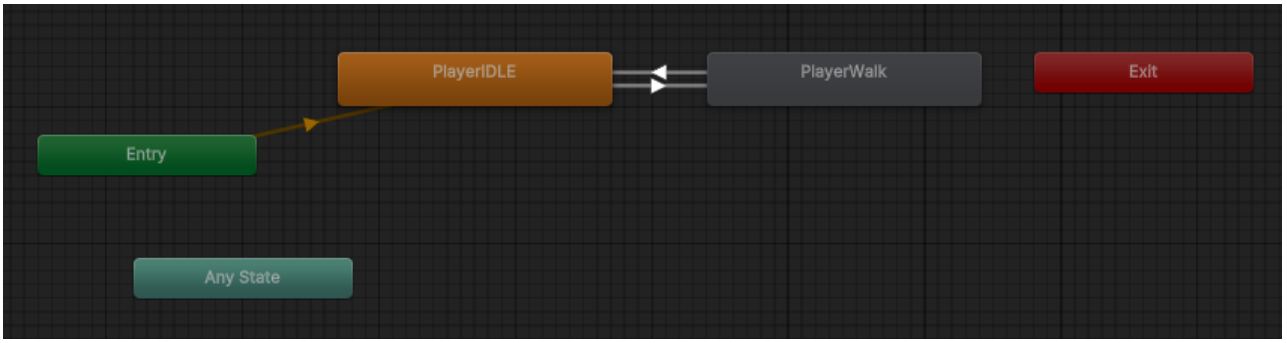


Рисунок 3.13. Аніматор

Дана схема представляє собою перемикач між станами Idle(спокою) і Walk(ходьби). У разі, коли швидкість персонажа менша за 0.1 – програється анімація спокою. У випадку швидкості, відмінної від нуля, персонаж має анімацію ходьби. Аналогічні схеми використані для всіх мобів, окрім слимака. Він має лише одну анімацію.

Атаки головного героя мають вигляд взмаху від меча, їх анімації зображено на рисунку 3.14.

Рисунок 3.14. Анімації атак персонажу

В грі на рівнях присутні вороги: слимак, гоблін і привид. Кожен має свою унікальну анімацію(рис 3.15-3.16).

Рисунок 3.15. Анімації ворогів

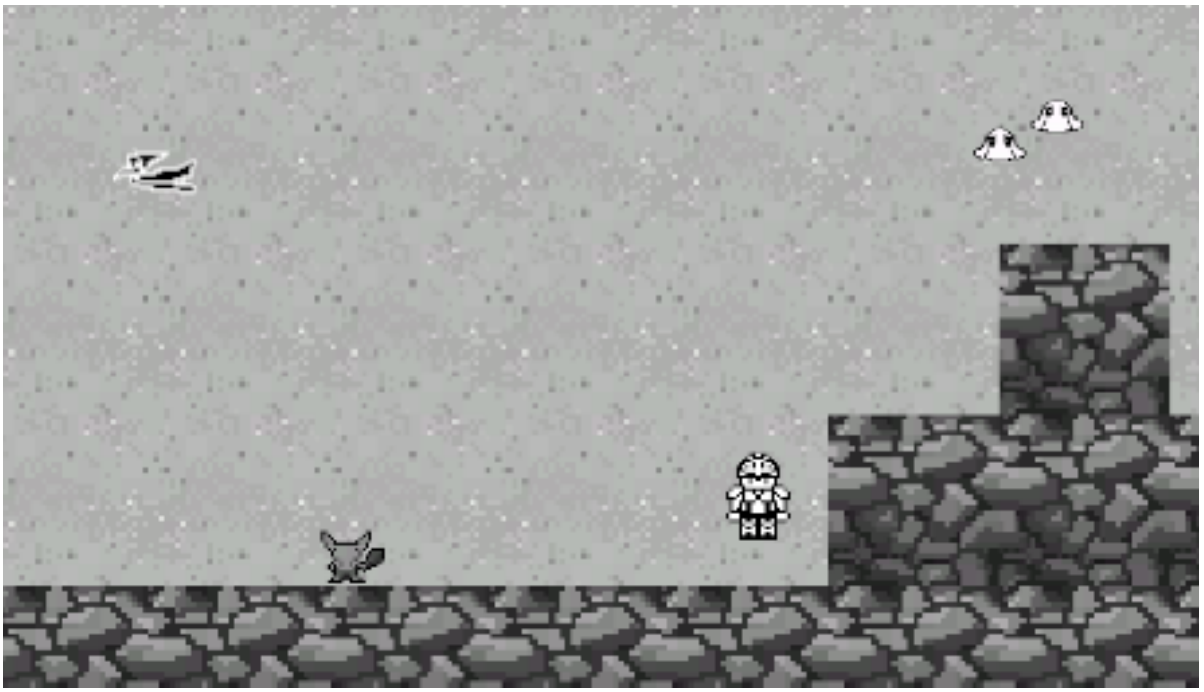


Рисунок 3.16. Представлення ворогів в грі

Також у грі присутні спеціальні ефекти, які програються при певних обставинах, таких як: лікування, отримання пошкоджень або їх нанесення(рис 3.17).

Рисунок 3.17. Спеціальні ефекти

Останній бос демо представляє собою гігантського кам’, представленого на рисунку 3.18.

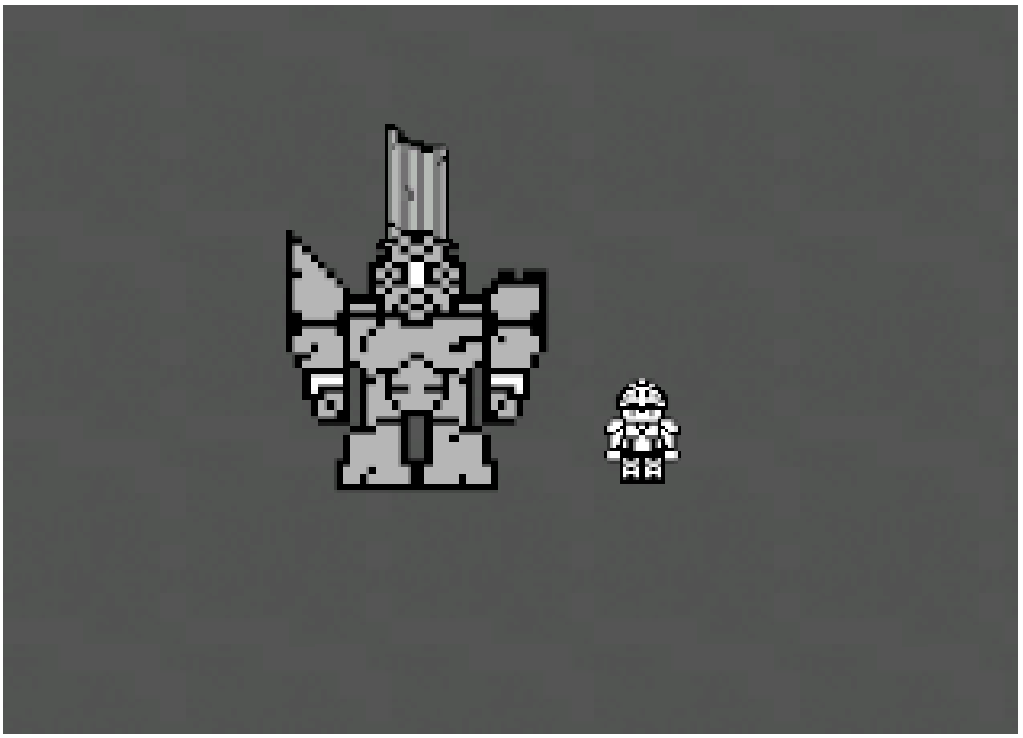


Рисунок 3.18. Голем

Також гра має власний вказівник миші(рис.3.19).

Рисунок 3.19. Вказівник

Зменшеним логотипом гри і її основним маскотом для кращої впізнавальності, було обрано гобліна. Його піктограму можна побачити на іконці запуску гри (рис. 3.20).

Рисунок 3.20. Ярлик запуску гри в файловій мережі Windows

3.3. Інтерфейс

Інтерфейс так само представлено в монохромному стилі. Так, як гра розроблялась в якості демо, вкладку налаштувань було вирішено замінити описом керування. Для зручності використано графічний опис клавіш. Рисунки 3.21-3.23.

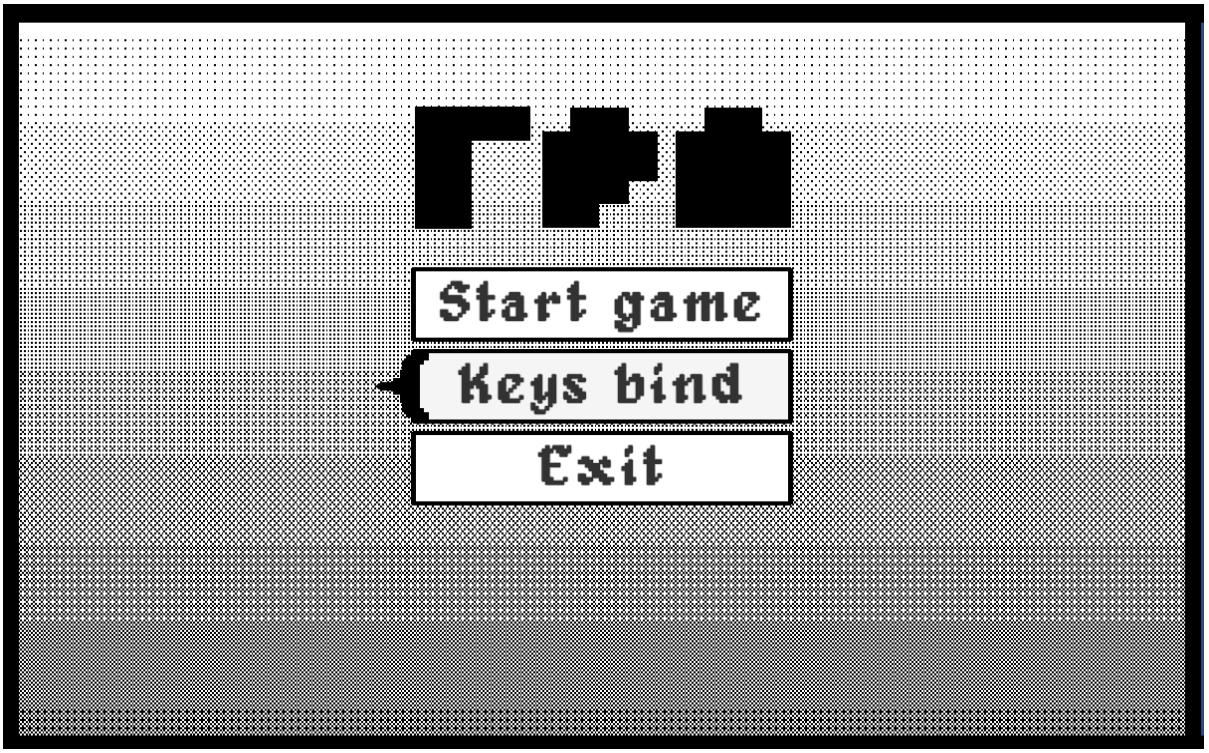


Рисунок 3.21. Головне меню

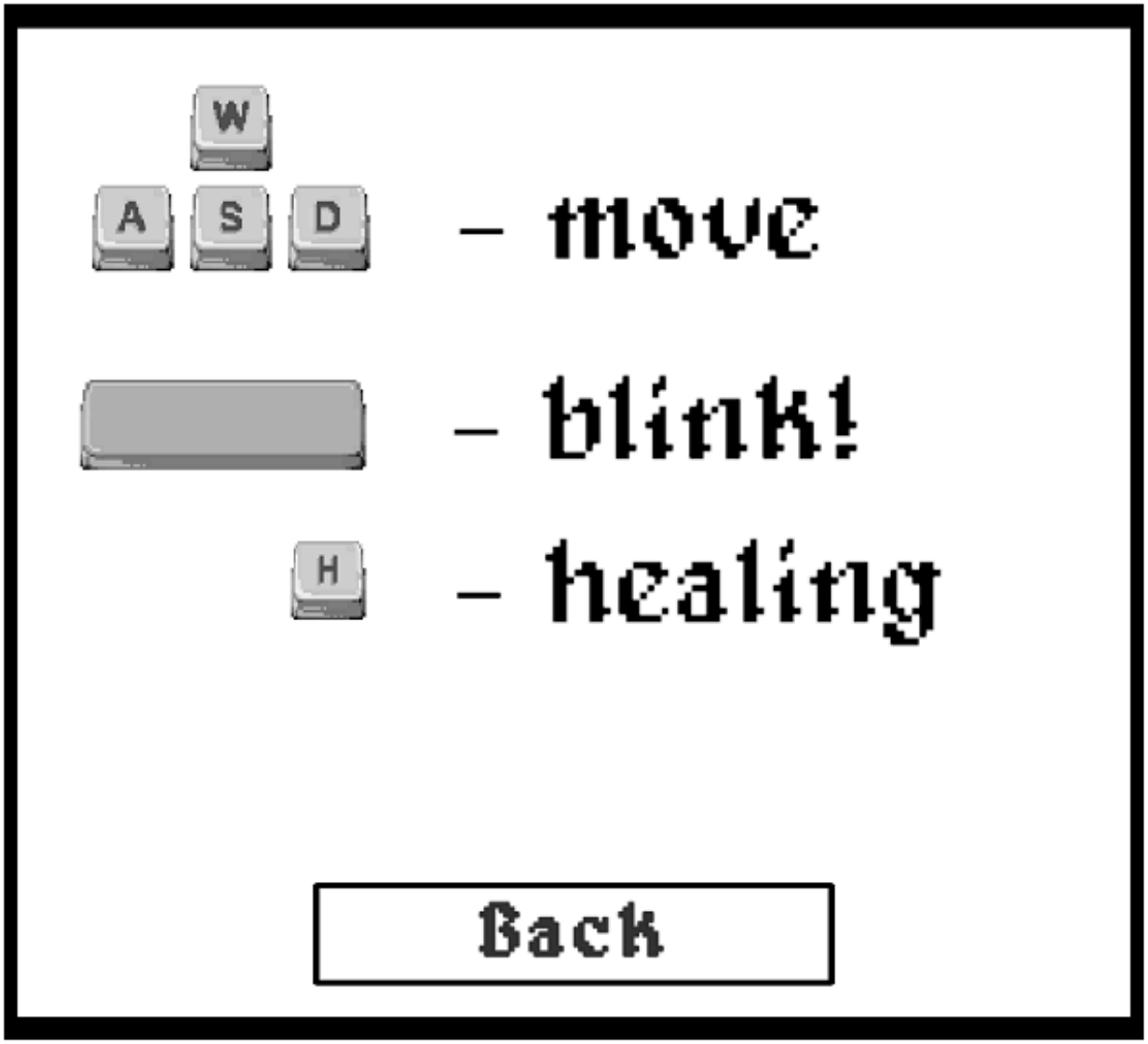


Рисунок 3.22. Меню Keys bind

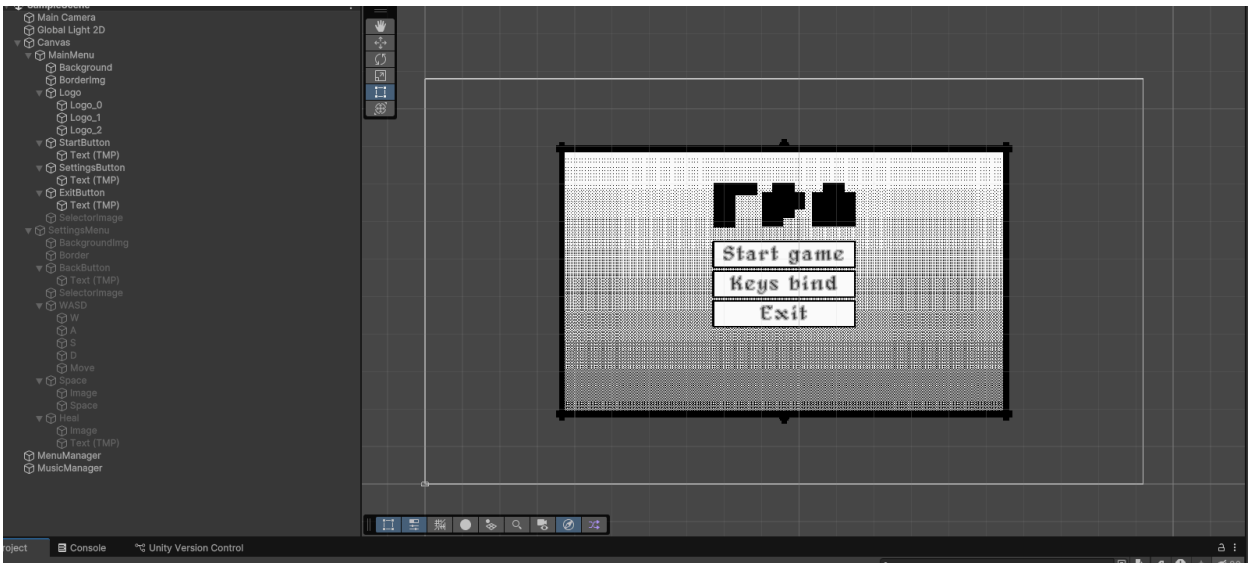


Рисунок 3.23. Відображення сцени головного меню в Unity

Для зручності і можливості зупинити ігровий процес, було додано невелике меню паузи, зображене на рисунку 3.24-3.25.

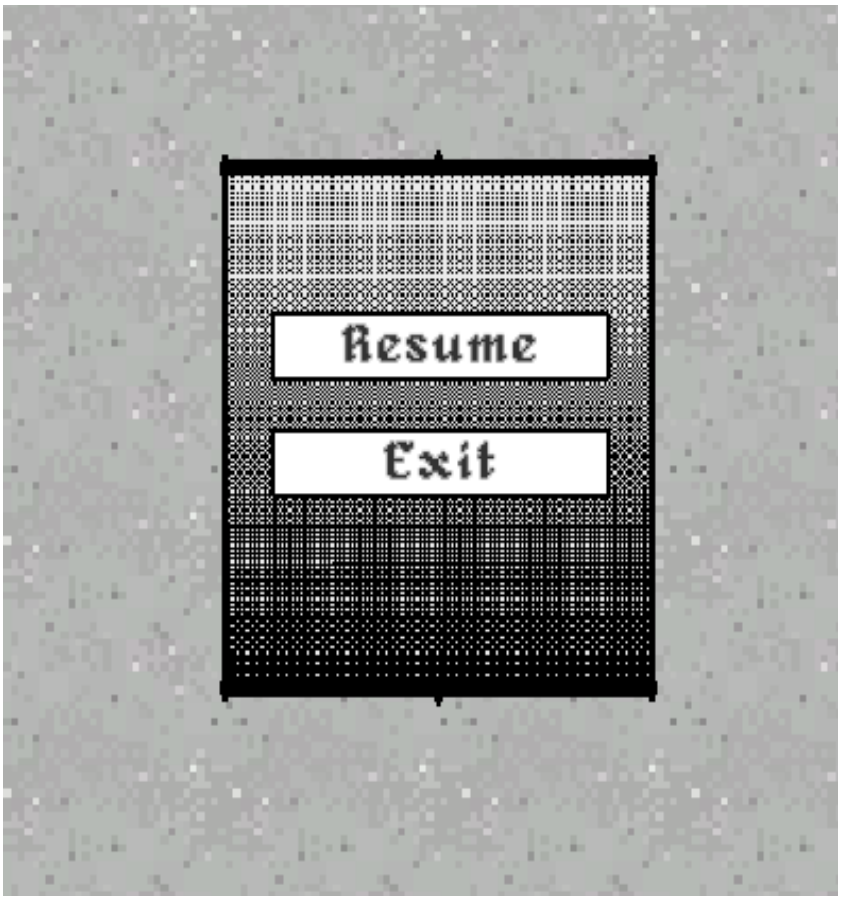


Рисунок 3.24. Пауза

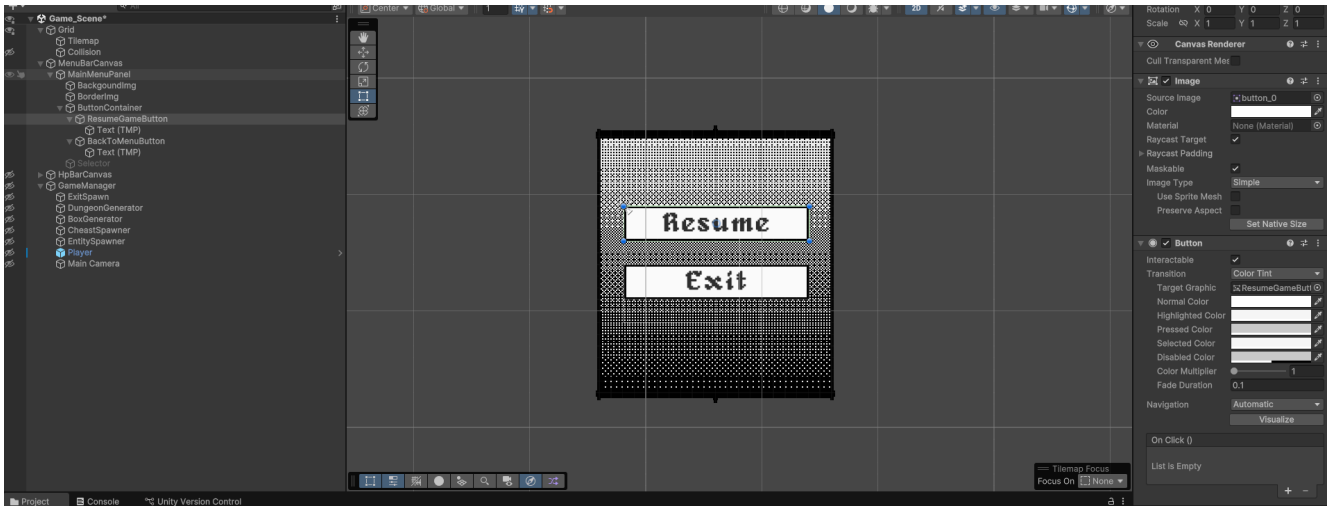


Рисунок 3.25. Представлення паузи в Unity

Після проходження останнього босу рівня гравець отримує повідомлення про перемогу, зображене на рисунку 3.26.

Рисунок 3.26. Повідомлення про перемогу

У разі, якщо шкала здоров’я падає до нуля, гравець отримує сповіщення про поразку(рис.3.27) і буде повернений до головного меню.

Рисунок 3.27. Сповіщення про поразку

На екрані постійно видима шкала очок здоров’я персонажа. Вона змінюється відповідно кількості отриманих пошкоджень або лікування. Шкала використовує слайдер, з накладеним на нього візуалом інтерфейсу. На рисунку 3.28-3.29. зображено приклад шкали, з неповною кількістю здоров’я.

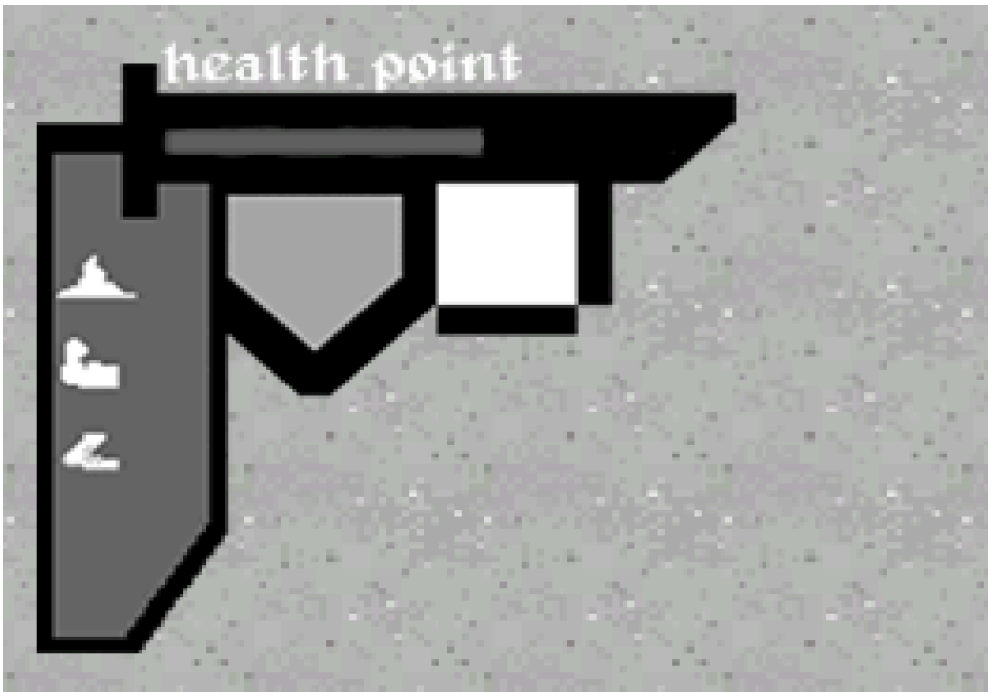


Рисунок 3.28. Шкала здоров’я

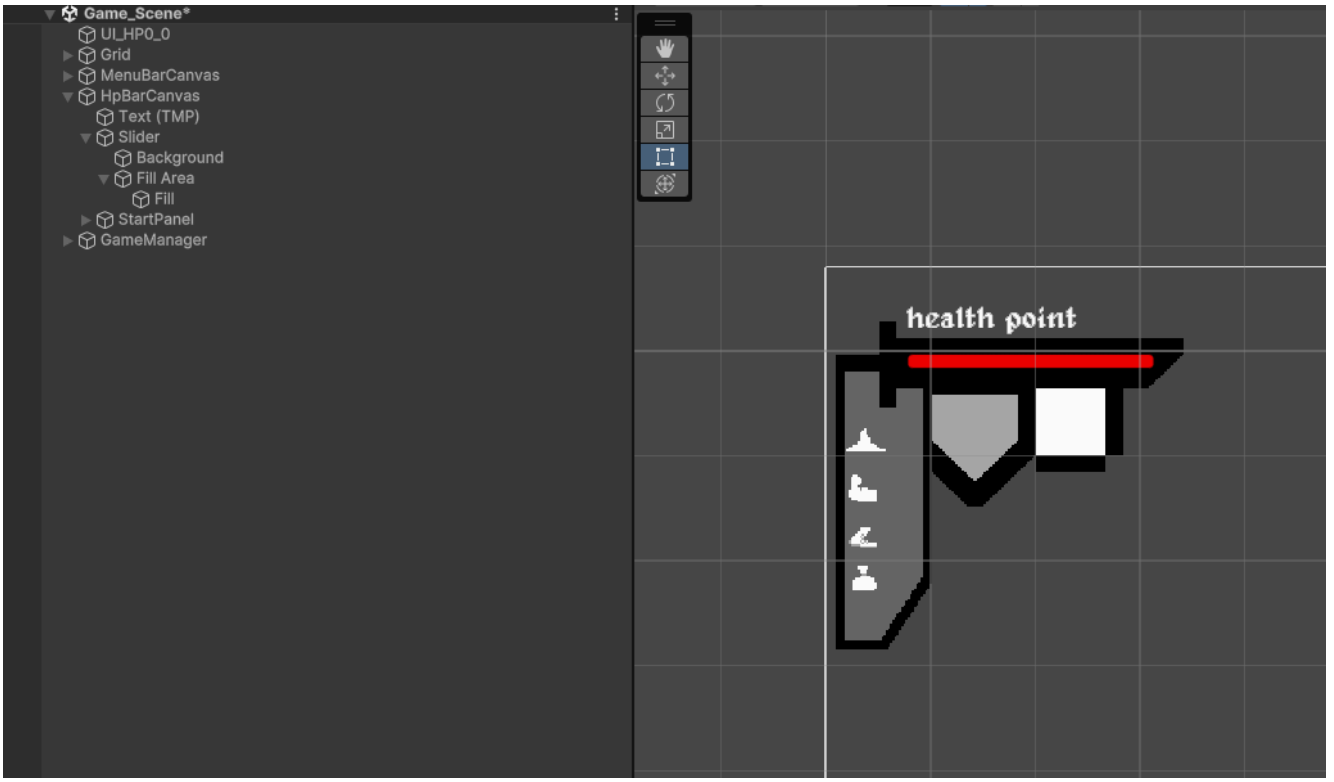


Рисунок 3.29. Представлення в Unity

Програмна частина інтерфейсу надана в додатку Б.

Розділ 4. ОНЛАЙН ДИСТРИБ'ЮЦІЯ І МАРКЕТИНГ

4.1. Steam

Основною платформою розповсюдження проєкту буде найбільший сервіс онлайн дистриб'юції ігор Steam[28]. Steamworks[29] дозволяє зареєструватися будь-кому в якості розробника і продавати ігри. Розробник має сплатити 100USD за кожен проект. Сервіс дозволяє створити сторінку гри, додати трейлери, скріншоти, опис і віковий рейтинг, після чого Valve[30] перевірить сторінку і затвердить. Гру можна випустити одразу або запланувати релізне вікно. У сервіси присутня функція "Wishlist"[], яка дозволяє бачить кількість людей, які додали гру в бажане. Wishlist — головний індикатор потенційного успіху. Макет сторінки зображено на рисунку 4.1.

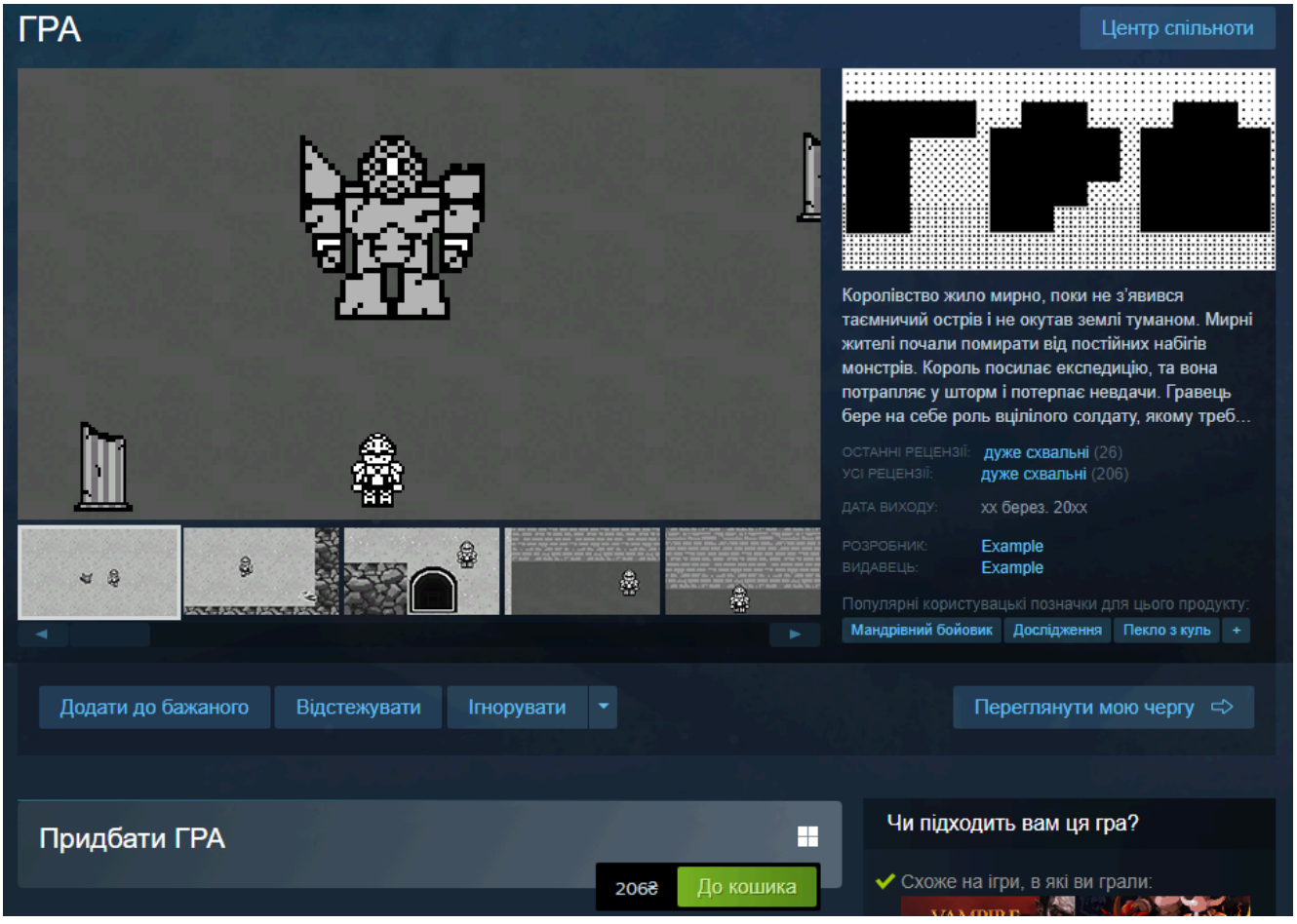


Рисунок 4.1. Макет сторінки

Гра буде випущена в моделі публікації Early Access. Дана модель дозволяє продавати продукт, ще до завершення розробки. Цей підхід дозволяє отримувати зворотній зв’язок, тестувати механіки і частково фінансувати розробку. Після закінчення раннього доступу ціну продукту можна залишити або підвищити. До ризиків такого підходу входять можливість поганих відгуків, тиск від спільноти, довге перебування в ранньому доступі може нашкодити іміджу компанії.

Steam бере комісію 30% від продажу гри. Після продажів на 1000USD передплата 100USD повертається. Після продажів на понад 10000000USD комісія зменшується до 25%, а після 50000000 – до 20%.

Також є можливість створити Steam Trading Cards. Для цього необхідно 5+ ілюстрацій, емодзі, фон профілю і бейдж. Вони мотивують гравців грати довше або купувати гру. Steam ділить прибуток від продажу карток на внутрішньому ринку Steam . Розробник отримує 10% від кожної операції на ринку.

Також Steam дозволяє генерувати ключі активації продукту, що дозволяє продавати продукт на інших маркетплейсах, за умови, що ціна буде не нижча.

В якості додаткового контенту, можна продавати музичні композиції гри за невелику ціну.

У підсумку маємо таблицю 4.1. з перевагами.

Таблиця 4.1.1. Переваги дистриб’юції в Steam

Категорія	Фіча	Що дає
Дистрибуція та продаж	Steam Store	Продаж гри в найбільшому PC-магазині
Спільнота	Community Hub	Форум, скріншоти, відео
	Steam Groups	Група фанатів гри
	Оголошення та знижками	Знижки, участь в Steam Sale
	Керування ціною та оновлення	Пости про апдейти, новини
	Мультимовність	Локалізація сторінки гри
Інтерактивність	Steam Trading Cards	Карти, бейджі, монетизація
	Ключі Steam	Продаж через інші платформи
	Steam Achievements	Досягнення, статистика
	Steam Cloud	Хмарне збереження, сейви
	Платіжна інфраструктура	Valve бере на себе оплату, повернення
	Beta-канали	Окремі білди для тестів
	Steam Events	Події та анонси гри

Просування	Wishlist & Followers	Ключовий фактор видимості
	Tags & Metadata	Правильне тегування
	Steam Next Fest	Демо-подія для трафіку
	Analytics Dashboard	Перегляди, wishlist, покупки
	Conversion Tracking	Зовнішній трафік (YouTube, Reddit тощо)
Мережа	Steam Networking	p2p-з'єднання або матч-хостинг
	Steam Voice Chat	Голосовий чат
	Matchmaking	Система лоббі/матчмейкінгу
Безпека	Steam DRM	Захист від піратства
	Valve Anti-Cheat (VAC)	Античит для мультиплеєра
	Crash Reporting	Автоматичні логи збоїв
	Steamworks SDK	Інтеграція API
Оновлення та контент	Auto-updates	Автоматичне оновлення гри
	DLC support	Продаж додаткового контенту
	Free Weekends	Безкоштовні періоди гри

4.2.Humble Bundle

Humble Bundle[31] — інтернет-сервіс, що спеціалізується на продажі цифрових копій комп'ютерних ігор для платформ Microsoft Windows, macOS і Linux. Проєкт Humble Bundle виріс із серії комерційних експериментів з продажу наборів з декількох ігор з моделі «Плати скільки хочеш». Вона спеціалізується на продажі продуктів у наборах. Таким чином, можна продавати Steam ключі гри.

Humble бере 25% з продажів, якщо це бандл, розробник отримує від 15% до 40%, в залежності від домовленості, решта – на благодійність та Humble. Для публікації гри, треба звернутися до humble і пройти відбір. У разі невдачі, продукт можна викласти у Humble store – магазин, аналогічний Steam. Список переваг викладено в таблиці 4.2. Типове зображення комплектів можна побачити на рисунок 4.2.

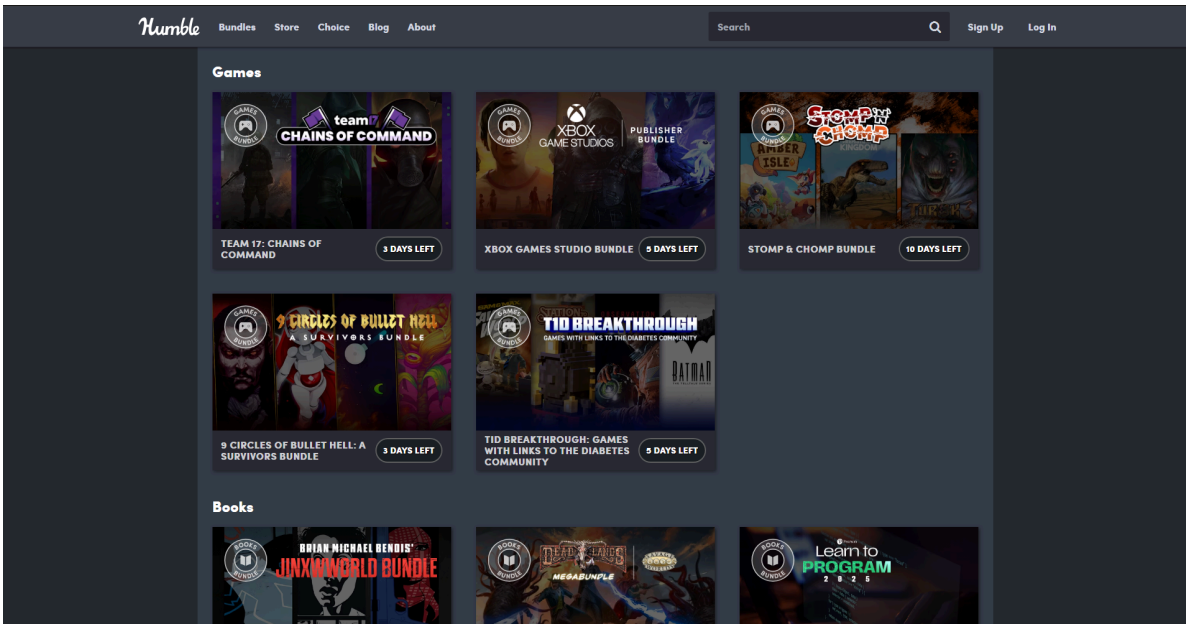


Рисунок 4.2 Інтерфейс користувача Humble

Таблиця 4.2. Переваги публікації на платформах Humble

Перевага	Опис
Гнучка модель оплати	Користувач платить скільки хоче, частина доходу може йти на благодійність, розробника і Humble.
Великий трафік	Популярні бандли отримують сотні тисяч переглядів і продажів.
Промоція через благодійність	Залучення нової аудиторії, яка підтримує благодійні ініціативи.
Підтримка Steam-ключів	Можна продавати Steam-версії гри через Humble.
Humble Store	Постійний магазин, де можна продавати гру без бандлів.
Humble Widgets	Інтегрування кнопки купівлі гри на своєму сайті (з мінімальною комісією).
Humble Games (видавництво)	Можливість співпраці з видавцем, що спеціалізується на інді.
Роялті та прозорість	Чітка система поділу прибутку між розробником, Humble і благодійністю.
Добра репутація	Платформа має лояльну аудиторію геймерів і геймдев-спільноти.

4.3. Itch.io

Itch.io[32] — це незалежна платформа для розповсюдження ігор (в основному інді), яка відома своєю відкритістю, гнучкістю та демократичністю. Якщо Steam — це супермаркет, то Itch.io — це місцевий творчий ярмарок. Його часто використовують інді-розробники, геймджемери та експериментатори. Розробник сам вибирає ціну продукту та комісію платформи. Для публікації достатньо лише створити акаунт і завантажити гру. За замовчуванням платформа бере 10% комісії, та це число можна зменшити до 0%. На рисунку 4.3 зображено типовий вигляд сторінки Itch.io

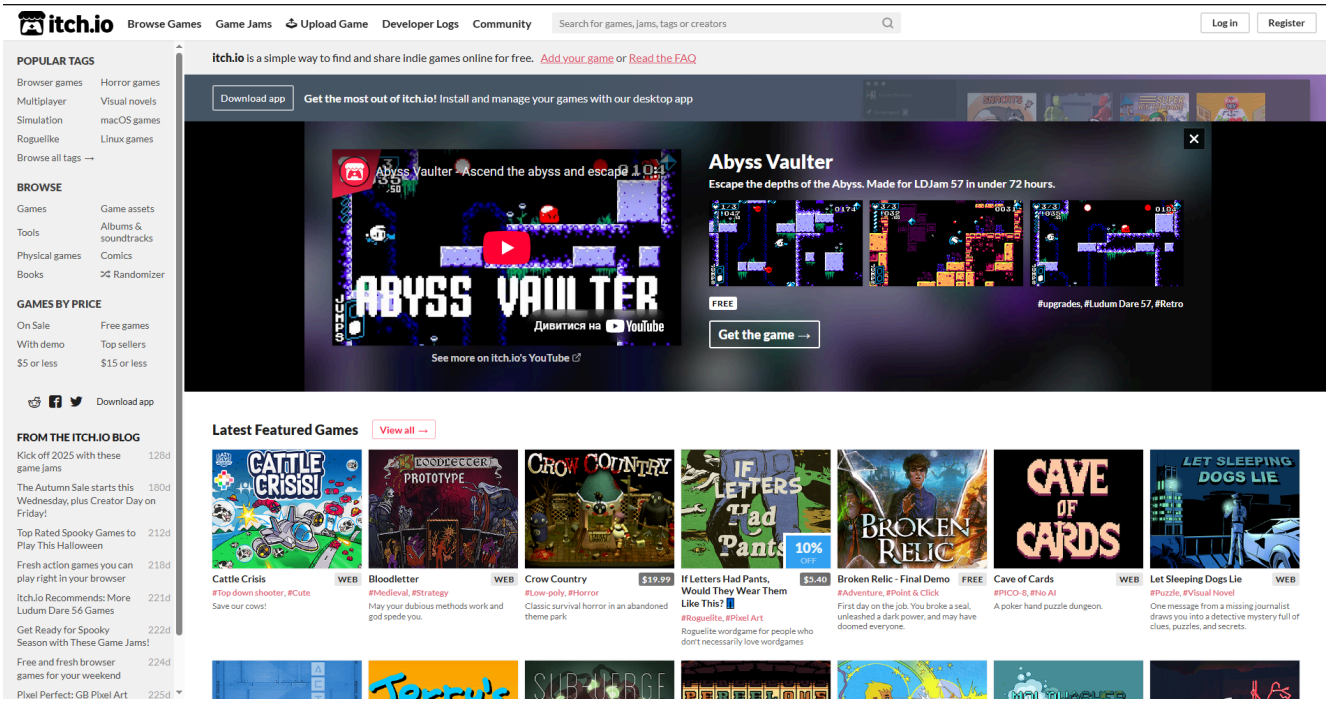


Рисунок 4.3. Itch.io

4.4. Маркетинг

Для просування будуть використані основні платформи розповсюдження контенту: Youtube, X, TikTok. Враховуючи вимоги і можливості, маємо таблицю 4.4.1. з доступними варіаціями розповсюдження інформації про продукт.

Таблиця 4.3. Варіації розповсюдження гри в мережах

Платформа	Дія	Результат
YouTube	Трейлер гри	30–90 секунд геймплею з музикою та логотипом
YouTube	Devlog / історія розробки	Відео про створення гри: behind the scenes
YouTube	Інфо-відео для інфлюенсерів	Короткий гайд: що це за гра, жанр, фішки
YouTube	Розсилка ключів	Писати напряму стрімерам/ютуберам (навіть маленьким)
YouTube	SEO-оптимізація відео	Назва, теги, опис з ключовими словами
TikTok	Геймплейні кліпи	5–15 секунд WOW-моментів або фейлів
TikTok	Меми та гумор	Смішні сцени або порівняння з іншими іграми
TikTok	Закадровий голос / текст	Надай контекст або емоцію до відео
TikTok	Використання трендів	Популярні звуки, формати, челенджі — адаптувати під гру
TikTok	Взаємодія з фанатами	Відповіді на коментарі, дуети
Twitter/X	GIF/скріншоти з текстом	Яскраві кадри + цікава підводка
Twitter/X	Devlog-треди	Наприклад: "як я зробив босфайт за 2 дні "
Twitter/X	Використання хештегів	#indiedev #screenshotsaturday #wishlistwednesday
Twitter/X	Участь у геймдев-подіях	Screenshot Saturday, IndieDevHour, Steam Next Fest
Twitter/X	Нетворкінг з іншими інді	Фолов, коментарі, взаємодія з розробниками

4.5.Дорожня карта і майбутні оновлення

Road map (укр. «дорожня карта») — це план або стратегічний документ, який показує етапи досягнення певної мети, розвиток продукту, проєкту, технології або бізнесу з урахуванням часу. Це візуальне представлення того, що, коли і як має бути зроблено.[36]

Дорожня карта часто використовується розробниками, як спосіб привабити нинішніх покупців майбутніми розробками. Детальна дорожня карта зображена у таблиці 4.4.

Таблиця 4.4 Дорожня мапа

Вікно	Апдейт
Другий квартал 2025р	Реалізація демо версії гри
Третій квартал 2025р	Реалізація системи предметів
	Створення сторінок гри у онлайн крамницях
	Початок маркетингової кампанії
Четвертий квартал 2025р	Реалізації механіки жаб’ячого зору
	Створення повноцінної рівневої системи
	Введення системи атрибутів персонажа
Перший квартал 2026р	Введення сюжету
	Балансування предметів
	Вихід в ранній доступ
Другий квартал 2026р	Збір відгуків
	Балансування та додавання предметів
	Виправлення багів
Третій квартал 2026р	Реліз
Четвертий квартал 2026р	Додавання безкінечного режиму
	Відлагодження і балансування

Другий квартал 2025

• Реалізація демо-версії гри – Створення базової ігрової збірки з ключовими механіками: рух, бій, генерація рівнів, 1-2 типи ворогів — для демонстрації ігрового циклу.

Третій квартал 2025

• Реалізація системи предметів – Додавання предметів/перків, які впливають на геймплей (зброя, бонуси, прокляття тощо).

• Створення сторінок гри у онлайн-крамницях – Реєстрація гри на платформах (наприклад, Steam, Itch.io) з трейлером, описом, скриншотами.

• Початок маркетингової кампанії – Створення соцмереж, початок постів, Devlog-відео, обговорення з гравцями.

Четвертий квартал 2025

• Реалізація механіки "жаб’ячого зору" – Механіка обмеженого огляду гравця (наприклад, світло навколо персонажа, решта затемнена).

• Створення повноцінної рівневої системи – Процедурна генерація багатьох рівнів з переходами, зростанням складності.

• Введення системи атрибутів персонажа – Сила, спритність, витривалість — базові характеристики, що впливають на бої, швидкість тощо.

Перший квартал 2026

• Введення сюжету – Інтеграція сюжетних вставок, персонажів, кат-сцен або текстових подій.

• Балансування предметів – Тестування та налаштування ефектів предметів для справедливого геймплею.

• Вихід у ранній доступ – Публічний реліз для обмеженої аудиторії з регулярними оновленнями.

Другий квартал 2026

• Збір відгуків – Аналіз коментарів гравців, оглядів, форумів.

• Балансування та додавання предметів – Виправлення проблем з балансом і створення нових предметів.

• Виправлення багів – Фікс критичних і дрібних помилок.

Третій квартал 2026

• Реліз – Повноцінний запуск гри: офіційний трейлер, оновлена сторінка гри, маркетинг, огляди.

Четвертий квартал 2026

• Додавання безкінечного режиму – Новий режим з нескінченними рівнями і поступовим ускладненням.

• Відлагодження і балансування – Полірування геймплею, покращення UX, виправлення залишкових помилок, тонке налаштування балансу.

ВИСНОВКИ

В результаті виконання дипломного проєкту, було проведено детальний аналіз жанру Rogue-like і найбільших представників жанру на ринку. Виділені основні жанрові особливості, такі як: варіативність, випадковість, реіграбельність.

Проведена робота з дослідженням можливостей рушія Unity і його модулів. Використання можливостей мови C# у поєднанні з бібліотеками Unity для створення ігрових механік, інтерфейсу, сценаріїв.

З нуля розроблений концепт гри, затверджений однорідний візуальний стиль для інтерфейсу, оточення, персонажів. Спроектовані межі, правила, шаблони, яких має дотримуватися проєкт. Закладені основні ідеї, напрямки розробки для майбутніх оновлень.

Були створені макети ігрових сцен, меню, інтерфейсів. На їх основі розробленні повноцінні елементи інтерфейсу. На основі описів локацій, створені текстури матеріалів, елементи оточення, які б підходили до рівнів. Створені спрайти неігрових персонажів, анімації їх і їх атакам.

Проведений аналіз музичних композицій у старих іграх, побудовано рамки обмежень для збереження характерного звучання. Створено унікальні треки під кожну сцену.

З використанням усіх наробок, макетів, збереженням правил і характерного візуального стилю, було створено демо гри, з характерними жанру особливостями, унікальним монохромим стилем, основними механіками і інтерфейсом.

Враховуючи велику ціну і обмеженість фізичної дистриб’юції, проаналізовані варіанти продажу проекту через онлайн сервіси Steam, Humble, Itch.io. Розглянуті варіанти просування інформаційної кампанії з використанням актуальних соціальних мереж, сайтів відеохостингу. Побудована дорожня карта майбутніх оновлень і встановлені рамки їх виходу.

Хоча даного проєкту і не достатньо для повноцінного релізу, та дана робота являє собою розширений приклад game design document [33], який можна представити зацікавленому видавцеві або відкрити краундфайндингову кампанію.

СПИСОК ЛІТЕРАТУРИ

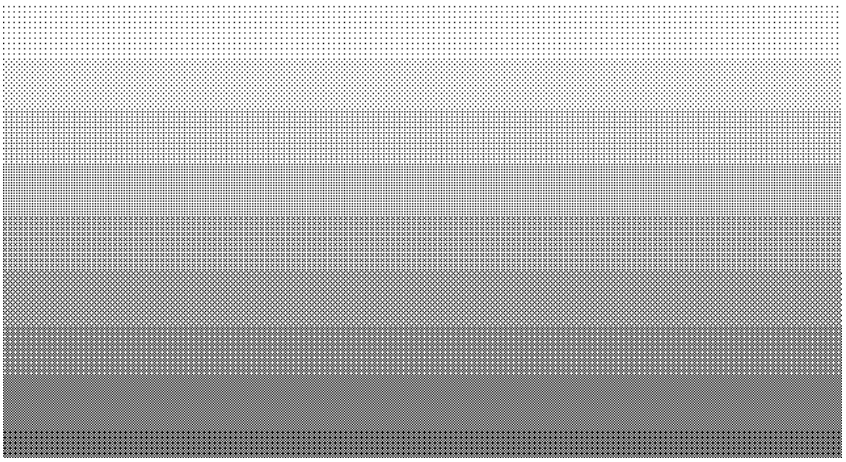
1. Avatar: The Way of Water. UML: https://en.wikipedia.org/wiki/Avatar:_The_Way_of_Water#:~:text=Budget-,%24350%E2%80%93460%C2%A0million,-%5B2%5D https://en.wikipedia.org/wiki/Cyberpunk_2077#:~:text=%24436%20million%20and%20%24441%20million
2. Інді-ігри. UML: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B4%D1%96-%D1%96%D0%B3%D1%80%D0%B8
3. Українські інді-ігри, які точно варто придбати та пройти. UML: https://tayemnakimnata.com/ukrayinski-indi-igry-yaki-tochno-varto-prydbaty-ta-projty-onovlyuyetsya/
4. How Indie Games Are Becoming Industry Hits. UML: https://devotedstudios.com/how-indie-games-are-becoming-industry-hits/
5. Indie Games: Shaping the Future of the Gaming Universe. UML: https://negativefive.vc/uncategorized/indie-games-impact/
6. Why the Roguelike Genre Has Become So Popular. UML: https://www.cbr.com/roguelike-video-games-genre-best
7. Unity 6 User Manual. UML: https://docs.unity3d.com/Manual/
8. Roguelike Tutorials. UML: https://rogueliketutorials.com/
9. Procedural Dungeon Generation Algorithm. UML: https://www.gamedeveloper.com/programming/procedural-dungeon-generation-algorithm
10. r/roguelikedev. UML: https://www.reddit.com/r/roguelikedev/
11. How to start making pixel art #1. UML: https://medium.com/pixel-grimoire/how-to-start-making-pixel-art-2d1e31a5ceab
12. Список найпопулярніших ігор жанру URL: https://steam250.com/tag/roguelike
13. Стаття про найкращих представників жанру URL:https://www.rockpapershotgun.com/best-roguelike-games-pc

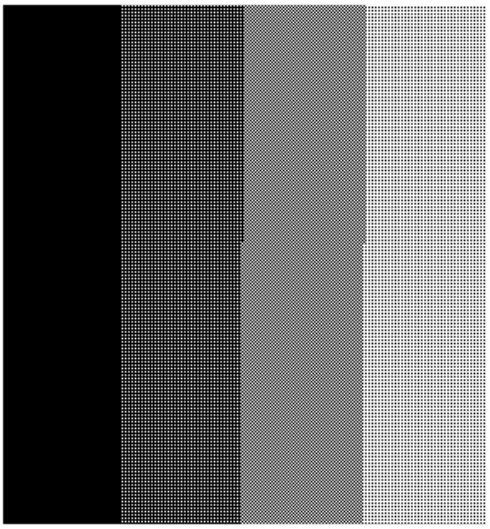
14. Найвище оцінені rogue ігри URL:https://www.metacritic.com/browse/game/all/roguelike/all-time/metascore/?releaseYearMin=1958&releaseYearMax=2025&genre=roguelike&page=1
15. The Binding of Isaac. URL: https://store.steampowered.com/app/113200/The_Binding_of_Isaac/
16. Hades. URL: https://store.steampowered.com/app/1145360/Hades/
17. Why I haven't played Hades. URL: https://www.youtube.com/watch?v=C9S83Xmuq4A
18. Spelunky 2. URL: https://store.steampowered.com/app/418530/Spelunky_2/
19. Final Fantasy 1. URL:https://uk.wikipedia.org/wiki/Final_Fantasy_I
20. Дерево цілей URL: http://moodle2.snu.edu.ua/mod/book/view.php?id=46497&chapterid=853
21. Проектування. URL: https://w.wiki/E7wbhttps://w.wiki/E7wb
22. Ескізний проект. URL:https://en.wikipedia.org/wiki/Engineering_design_process#:~:text=of%20possible%20alternatives.-,Preliminary%20design,-%5Bedit%5D
23. Діаграма прецедентів. URL: https://en.wikipedia.org/wiki/Use_case_diagram
24. Aseprite URL: https://en.wikipedia.org/wiki/Aseprite
25. Photoshop. URL: https://uk.wikipedia.org/wiki/Adobe_Photoshop
26. Reaper. URL: https://uk.wikipedia.org/wiki/REAPER
27. Третє покоління консолей URL: https://en.wikipedia.org/wiki/Third_generation_of_video_game_consoles
28. Steam URL: https://uk.wikipedia.org/wiki/Steam
29. Steamworks URL: https://developer.valvesoftware.com/wiki/Steamworks
30. Valve URL: https://en.wikipedia.org/wiki/Valve_Corporation
31. Humble BundleURL: https://en.wikipedia.org/wiki/Humble_Bundle
32. GDD URL: https://en.wikipedia.org/wiki/Game_design_document
33. Famistudio URL: https://famistudio.org/
34. Roadmap URL: https://en.wikipedia.org/wiki/Technology_roadmap

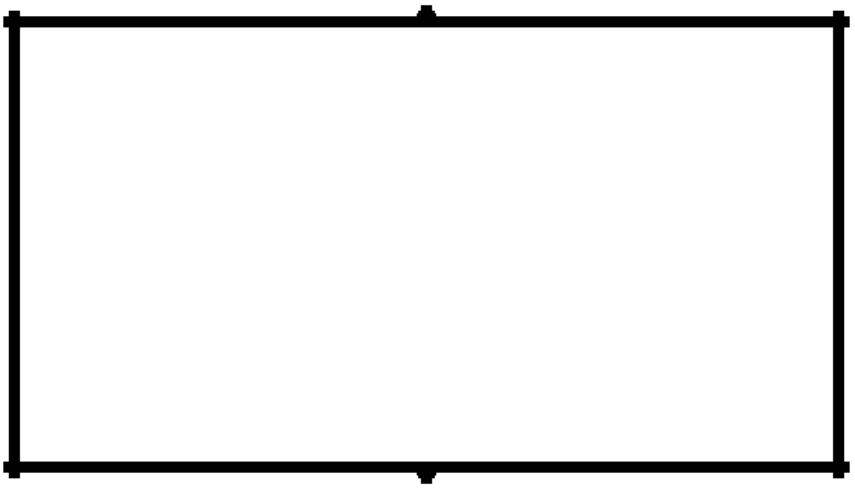
Додаток А. Кадри анімацій, спрайти, ефекти, текстури

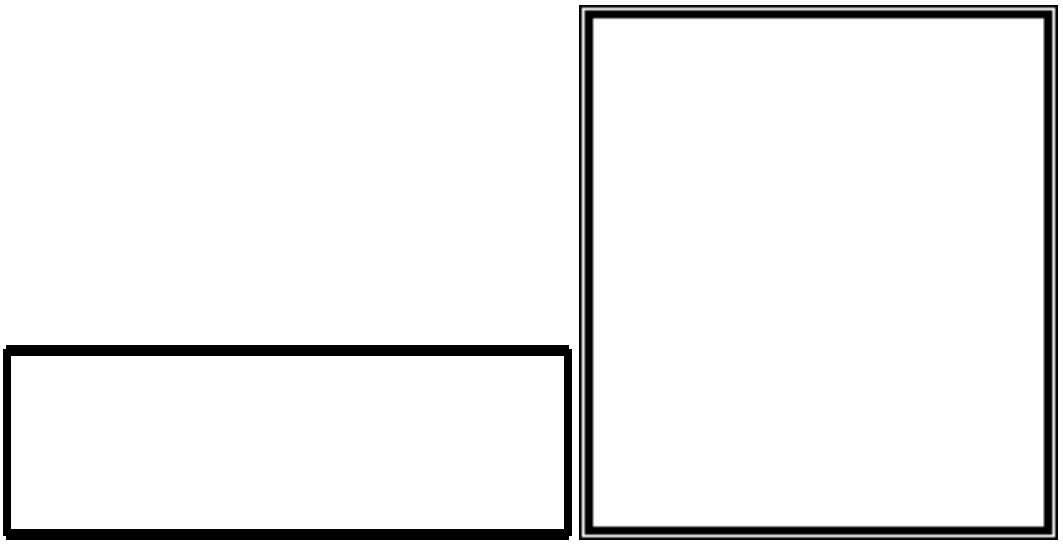
1. Спрайти оточення

2. Елементи інтерфейсу

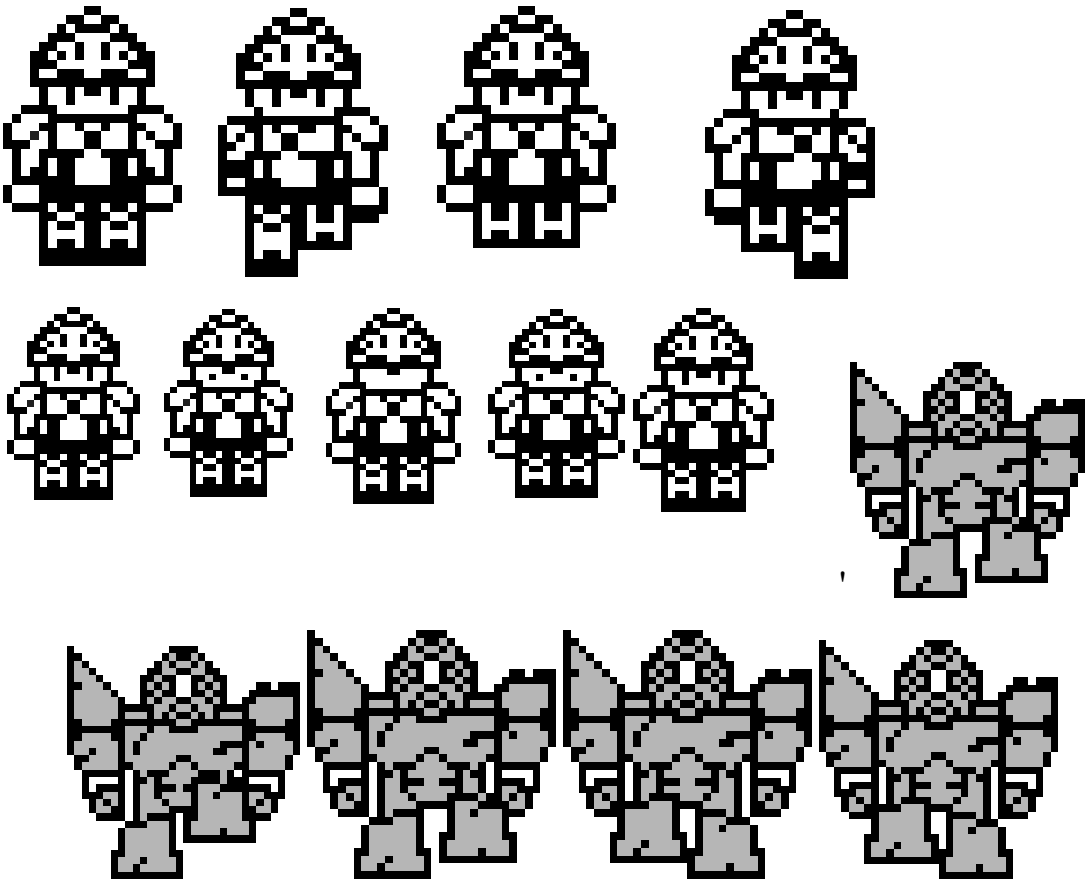


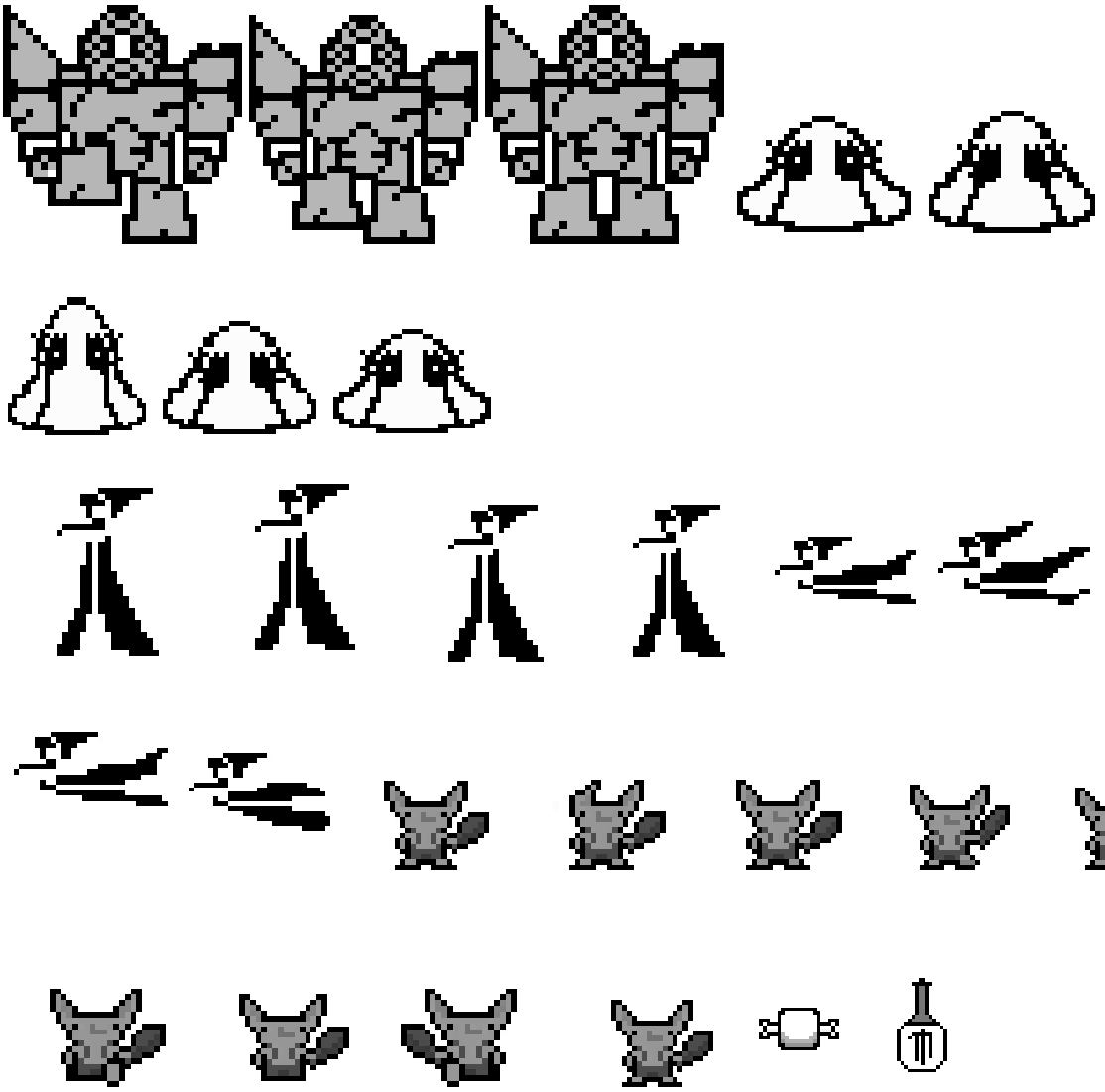






3. Спрайти і анімації
4.

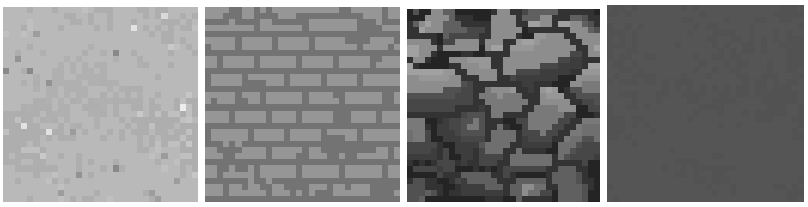




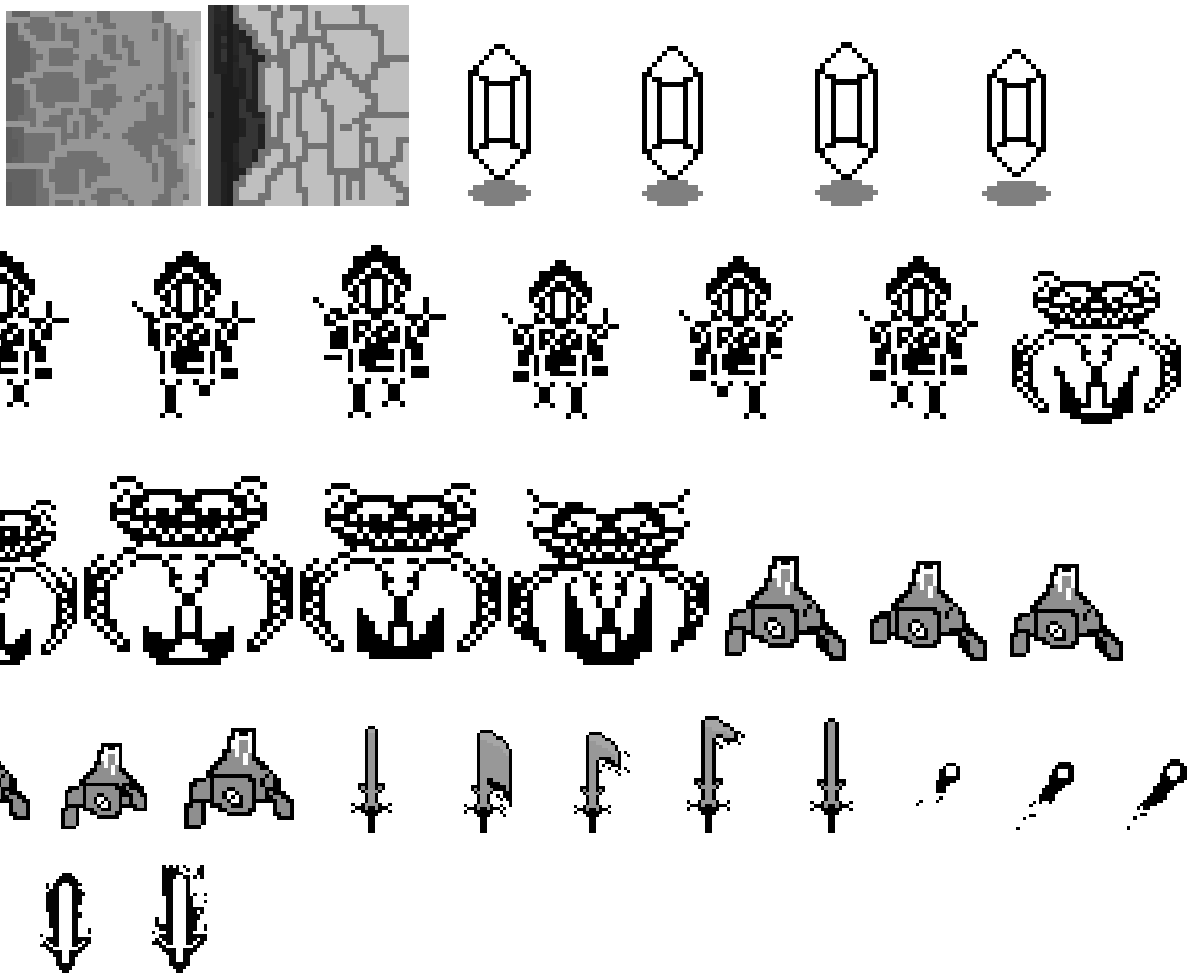
4.Анімації куль

5.Спеціальні ефекти

6.Текстури



7.Контент на майбутні оновлення



Додаток Б. Лістинг коду

1. PlayerHealthUI.cs

```
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Slider))]
public class PlayerHealthUI : MonoBehaviour
{
    [Header("Îáîâ'ÿçêîâ³ ïîñëäíÿ")]
    [SerializeField] private PlayerHealth playerHealth;
    [SerializeField] private Image fillImage;

    [Header("Íàëàøòóâàííÿ")]
    [SerializeField] private Color fullHealthColor = Color.green;
    [SerializeField] private Color criticalHealthColor = Color.red;
    [Range(0, 1)][SerializeField] private float criticalThreshold = 0.3f;

    private Slider healthSlider;

    private void Awake()
    {
        healthSlider = GetComponent<Slider>();
        if (fillImage == null) fillImage = GetComponentInChildren<Image>();

        if (playerHealth == null)
        {
            Debug.LogError("PlayerHealth íå çíàéäåíî!");
            return;
        }


        healthSlider.minValue = 0;
```

```
        healthSlider.maxValue = playerHealth.MaxHealth;


        UpdateHealth(playerHealth.CurrentHealth);
    }

    private void Start()
    {

        playerHealth.OnHealthChanged += UpdateHealth;
    }

    private void UpdateHealth(int newHealth)
    {
        healthSlider.value = newHealth;
        UpdateHealthColor((float)newHealth / playerHealth.MaxHealth);
        Debug.Log($"Îíîâëåíî HP: {newHealth}/{playerHealth.MaxHealth}");
    }

    private void UpdateHealthColor(float healthPercent)
    {

        if (healthPercent <= criticalThreshold)
        {
                fillImage.color = Color.Lerp(criticalHealthColor, fullHealthColor,
healthPercent / criticalThreshold);
        }
        else
        {
```

```
                fillImage.color = Color.Lerp(fullHealthColor, criticalHealthColor, (1 - healthPercent) / (1 - criticalThreshold));
            }
        }

        private void OnDestroy()
        {
            if (playerHealth != null)
                playerHealth.OnHealthChanged -= UpdateHealth;
        }
    }
```

2.GameMenu.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class GameMenu : MonoBehaviour
{
    [Header("UI Elements")]
    public GameObject menuPanel;
    public Button playButton;
    public Button backToMenuButton;

    [Header("Scene Settings")]
    public string mainMenuScene = "SampleScene";
    public string gameScene = "GameScene";

    private bool isPaused = false;

    private void Start()
```

```
{
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;

    if (!SceneExists(mainMenuScene))
    {
        Debug.LogError($"Scene '{mainMenuScene}' not found!");
        backToMenuButton.interactable = false;
    }

    if (!SceneExists(gameScene))
    {
        Debug.LogError($"Scene '{gameScene}' not found!");
    }

    playButton.onClick.AddListener(ResumeGame);
    backToMenuButton.onClick.AddListener(ReturnToMainMenu);

    menuPanel.SetActive(false);
}

private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        TogglePause();
    }
}

private bool SceneExists(string sceneName)
```

```
{
    for (int i = 0; i < SceneManager.sceneCountInBuildSettings; i++)
    {
        var scenePath = SceneUtility.GetScenePathByBuildIndex(i);
        var scene = System.IO.Path.GetFileNameWithoutExtension(scenePath);
        if (scene == sceneName) return true;
    }
    return false;
}

public void TogglePause()
{
    isPaused = !isPaused;
    Time.timeScale = isPaused ? 0f : 1f;
    menuPanel.SetActive(isPaused);

    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
}

public void ResumeGame()
{
    TogglePause();
}

public void ReturnToMainMenu()
{
    Time.timeScale = 1f;
    SceneManager.LoadScene(mainMenuScene);
}
```

```
    public void StartGame()
    {
        Time.timeScale = 1f;
        SceneManager.LoadScene(gameScene);
    }
}
```

3. MenuManager.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;
public class MenuManager : MonoBehaviour
{
    public GameObject mainMenu;
    public GameObject settingsMenu;
    public AudioSource musicSource;

    public void StartGame()
    {
        SceneManager.LoadScene("Game_Scene");
    }

    public void OpenSettings()
    {
        mainMenu.SetActive(false);
        settingsMenu.SetActive(true);
    }

    public void CloseSettings()
    {
```

```
        settingsMenu.SetActive(false);
        mainMenu.SetActive(true);
    }

    public void ExitGame()
    {
        Application.Quit();
    }
}
```

4.PreStory.cs

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class StartScreenController : MonoBehaviour
{
    [Header("UI Elements")]
    [SerializeField] private GameObject startPanel;
    [SerializeField] private TextMeshProUGUI storyText;
    [SerializeField] private Button continueButton;

    [Header("Story Settings")]
    [TextArea(3, 10)]
    [SerializeField] private string storyContent = "\r\nYou have been transported to a magical dimension. Find the gate to the main boss's chambers by passing through his dungeon. Defeat him and choose your right to freedom.";
    [SerializeField] private float typewriterSpeed = 0.05f;

    private void Awake()
    {
```

```
        if (startPanel == null) startPanel = GameObject.Find("StartPanel");
                                    if (storyText == null) storyText =
startPanel.GetComponentInChildren<TextMeshProUGUI>();
                                if (continueButton == null) continueButton =
startPanel.GetComponentInChildren<Button>();

        continueButton.onClick.AddListener(HidePanel);
    }

    private void Start()
    {
        startPanel.SetActive(true);
        storyText.text = "";
        Time.timeScale = 0f;
        StartCoroutine(TypewriterEffect(storyContent));
    }

    private System.Collections.IEnumerator TypewriterEffect(string text)
    {
        foreach (char letter in text.ToCharArray())
        {
            storyText.text += letter;
            yield return new WaitForSecondsRealtime(typewriterSpeed);
        }
    }

    private void HidePanel()
    {
        startPanel.SetActive(false);
```

```
            Time.timeScale = 1f;
        }


        [ContextMenu("Show Panel")]
        public void TestShowPanel()
        {
            startPanel.SetActive(true);
            storyText.text = storyContent;
            Time.timeScale = 0f;
        }
    }
```

5.ButtonSelector.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class ButtonSelector : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler
{
    public RectTransform selectorImage; // Îá'ºêò ñåëåêòîðà (image)
    public float offsetX = -40f;        // Çñóâ ë³âîðó÷ â³ä êíîïêè

    private RectTransform buttonRect;

    private void Awake()
    {
        buttonRect = GetComponent<RectTransform>();

        // Õîâàºìî ñåëåêòîð ïðè ñòàðò³
```

```
        if (selectorImage != null)
            selectorImage.gameObject.SetActive(false);
    }

    public void OnPointerEnter(PointerEventData eventData)
    {
        if (selectorImage != null)
        {
            selectorImage.gameObject.SetActive(true);

            Vector3 targetPosition = buttonRect.position;
            targetPosition.x += offsetX;

                        selectorImage.position = new Vector3(targetPosition.x,
buttonRect.position.y, buttonRect.position.z);
        }
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        if (selectorImage != null)
        {
            selectorImage.gameObject.SetActive(false);
        }
    }
}
```