МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка 2D гри на платформі Godot

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22011001

Здобувачка _____М. В. Ангел

підпис

«__» _____ 2024 р.

Керівник д-р техн. наук, професор _____А. В. Швед

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент __А. О. Алексєєва

підпис

«__» _____ 2024 р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Інженерії
Програмного Забезпечення
Є. О. Давиденко
«_22_» ___грудня___ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано здобувачу освіти групи 408 факультету комп'ютерних наук

_____Ангел Марині Валентинівні_____

(прізвище, ім'я, по батькові здобувачки освіти)

1. Тема кваліфікаційної роботи

Розробка 2D гри на платформі Godot_____

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № __269__

2. Строк представлення кваліфікаційної роботи «____» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом роботи є 2D гра розроблена на ігровому рушію Godot

4. Перелік питань, що підлягають розробці:

- аналіз аналогічних ігрових продуктів на платформі Godot та інших двигунах;

- визначення основних вимог до розроблюваної гри на основі аналізу сучасних тенденцій у геймдеві;

- розробка алгоритмів та логіки гри з використанням побудови UML-діаграм та блок-схем;

- створення прототипу гри та тестування його на відповідність вимогам;

5. Перелік графічних матеріалів:

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А.О	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи ___д-р техн. наук, професор Швед Алена Володимирівна_____

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання _____Ангел Марина Валентинівна_____

(прізвище, ім'я, по батькові здобувачки освіти)

_____

(підпис)

Дата видачі завдання «22» грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи бакалавра

Тема: «Розробка 2D гри на платформі Godot»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	22.12.2023	22.12.2023	Виконано
2	Огляд літератури за темою роботи	22.04.2024	22.04.2024	Виконано
3	Складання календарного плану КРБ	22.04.2024	22.04.2024	Виконано
4	Аналіз предметної галузі	23.04.2024	23.04.2024	Виконано
5	Розробка проєктних рішень	25.04.2024	25.04.2024	Виконано
6	Моделювання та конструювання ПЗ	26.04.2024	27.04.2024	Виконано
7	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	28.05.2024	01.06.2024	Виконано
8	Розробка спеціальної частини з охорони праці	01.06.2024	02.06.2024	Виконано
9	Відгук керівника КРБ	02.06.2024	02.06.2024	Виконано
10	Оформлення КРБ та презентації	02.06.2024	02.06.2024	Виконано
11	Попередній захист	03.06.2024	04.06.2024	Виконано
12	Рецензування	15.06.2024	17.06.2024	Виконано
13	Захист кваліфікаційної роботи	24.06.2024	25.06.2024	Виконано

Розробила здобувачка освіти _____Ангел Марина Валентинівна_____
(прізвище, ім'я, по батькові) *(підпис)*
«22» __квітня__ 2024 р.

Керівник роботи _____д-р. техн. наук, професор Швед Алена Володимирівна
(посада, прізвище, ім'я, по батькові) *(підпис)*

«22» __квітня_____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

“Розробка 2D гри на платформі Godot”

Здобувачка освіти 408 гр.: Ангел Марина Валентинівна

Керівник: д-р тех. наук, професор Швед А. В.

Кваліфікаційна робота бакалавра присвячена питанням дослідження можливостей платформи Godot та розробці ігрового застосунку в жанрі шутеру.

Актуальність теми обумовлена сучасними тенденціями в галузі індустрії відеоігор. З кожним днем зростає популярність 2D ігор серед гравців усіх вікових категорій. Розробка 2D ігрового додатку на платформі Godot відкриває безліч можливостей для ігрової творчості та реалізації унікальних ідей.

Об'єкт роботи: процес проєктування та розробки 2D ігрового застосунку на базі ігрового движку Godot.

Предмет роботи: методи програмної інженерії та інструментальні засоби розробки 2D ігрового застосунку на платформі Godot.

Мета: розробка 2D комп’ютерної гри у жанрі шутеру засобами рушія Godot, реалізувати складні алгоритми та розширені ігрові механіки.

Кваліфікаційна робота бакалавра складається з вступу, чотирьох розділів, висновків та переліку джерел посилань.

Перший розділ присвячено аналізу предметної області, що включає порівняльний аналіз популярних 2D ігрових застосунків у жанрі шутеру. За результатами проведеного аналізу сформульовано функціональні вимоги до розроблюваного ігрового застосунку.

У другому розділі наводиться опис процесу моделювання системи відповідно до визначених вимог. Використовуючи UML-діаграми, представлено структурну та поведінкову модель гри. Описано варіанти використання, діаграми послідовності та діяльності, що відображають основні ігрові процеси та взаємодії між об'єктами.

Третій розділ присвячено вибору засобів реалізації проєкту. Обґрунтовано вибір ігрового рушія Godot 4, його переваги та можливості у порівнянні з іншими рушіями. Описано використані мови програмування (GDScript) та інтегровані інструменти розробки. Детально розглянуто процес налаштування робочого середовища, організацію каталогів та файлів проєкту, а також використання сторонніх бібліотек та ресурсів для досягнення максимальної ефективності та продуктивності розробки.

Четвертий розділ зосереджено на програмній реалізації ігрового застосунку. Описано організацію робочого середовища та основну структуру проєкту, включаючи каталогізацію файлів та ресурсів. Розглянуто процес створення анімацій для персонажів та об'єктів гри, використання спрайтів та анімаційних об'єктів, а також методи управління анімацією. Описано систему колізій та взаємодії об'єктів гри, зокрема обробку зіткнень та алгоритми штучного інтелекту ворогів. Розглянуто механіки гри, пов'язані з взаємодією гравця з ворогами, включаючи систему бою та поведінкові шаблони ворогів.

Кваліфікаційна робота містить 79 сторінок, включаючи 4 розділи, 36 ілюстрації, 18 таблиць та перелік із 21 джерела посилань.

Ключові слова: ігровий застосунок, 2D ігровий додаток, Godot, оптимізація, розробка гри.

ABSTRACT

of the Bachelor`s Thesis

"Development of a 2D game on the Godot platform"

Student of group 408: Anhel Maryna Valentinovna

Supervisor: Dr. Tech. of Sciences, Professor Shved A.V.

The bachelor's qualification work is devoted to researching the capabilities of the Godot platform and developing a game application in the shooter genre.

The topicality of the topic is due to modern trends in the video game industry. The popularity of 2D games among players of all ages is increasing day by day. Development of a 2D game application on the Godot platform opens up many opportunities for game creativity and implementation of unique ideas.

Object of work: the process of designing and developing a 2D game application based on the Godot game engine.

Subject of work: software engineering methods and tools for developing a 2D game application on the Godot platform.

Goal: to develop a 2D computer game in the shooter genre using the Godot engine, to implement complex algorithms and advanced game mechanics.The bachelor's thesis consists of an introduction, four chapters, conclusions and a list of reference sources.

The first chapter is devoted to the analysis of the subject area, which includes a comparative analysis of popular 2D game applications in the shooter genre. Based on the results of the analysis, the functional requirements for the developed game application were formulated.

The second section describes the process of modeling the system according to the specified requirements. Using UML diagrams, the structural and behavioral model of the game is presented. Use cases, sequence diagrams, and activities are described, illustrating basic gameplay and interactions between objects.

The third section is devoted to the selection of means of project implementation. The choice of the Godot 4 game engine, its advantages and capabilities in comparison

with other engines is justified. The used programming languages (GDScript) and integrated development tools are described. The process of configuring the working environment, organizing project directories and files, as well as using third-party libraries and resources to achieve maximum development efficiency and productivity are discussed in detail.

The fourth chapter focuses on the software implementation of the game application. The organization of the working environment and the basic structure of the project are described, including the cataloging of files and resources. The process of creating animations for game characters and objects, the use of sprites and animated objects, as well as animation management methods are considered. The system of collisions and interaction of game objects is described, in particular, the processing of collisions and the algorithms of artificial intelligence of enemies. Game mechanics related to player interaction with enemies are reviewed, including the combat system and enemy behavior patterns.

The thesis contains 79 pages, including 4 chapters, 36 illustrations, 18 tables and a list of 21 references.

Keywords: game application, 2D game application, Godot, optimization, game development.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – програмне забезпечення

КРБ – кваліфікаційна робота бакалавра

UML – unified modeling language

UI – user interface

IDE – integrated development environment

GUI – graphical user interface

ВСТУП

Актуальність теми: обумовлена сучасними тенденціями в галузі індустрії відеоігор. З кожним днем зростає популярність 2D ігор серед гравців усіх вікових категорій. Розробка 2D ігрового додатку на платформі Godot відкриває безліч можливостей для ігрової творчості та реалізації унікальних ідей.

Об'єкт роботи: процес проєктування та розробки 2D ігрового застосунку на базі ігрового движку Godot.

Предмет роботи: методи програмної інженерії та інструментальні засоби розробки 2D ігрового застосунку на платформі Godot.

Мета: розробка 2D комп’ютерної гри у жанрі шутеру засобами рушія Godot, реалізувати складні алгоритми та розширені ігрові механіки.

Для досягнення цієї мети передбачається вирішення наступних **завдань:**

1) аналіз аналогічних ігрових продуктів на платформі Godot та інших двигунах;
2) визначення основних вимог до розроблюваної гри на основі аналізу сучасних тенденцій у геймдеві;
3) розробка алгоритмів та логіки гри з використанням побудови UML-діаграм та блок-схем;
4) створення прототипу гри та тестування його на відповідність вимогам.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ІГРОВИХ ЗАСТОСУНКІВ

1.1 Огляд сучасних ігрових жанрів

Аналіз сучасних ігрових жанрів відображає різноманітність та еволюцію ігрової індустрії. Основні жанри, такі як екшн, стратегія, рольові ігри, головоломки та інші, продовжують розвиватися та змінюватися під впливом нових технологій, творчих підходів та вподобань гравців.

Жанр екшн постійно досліджує нові способи динамічної геймплейної взаємодії та реалізації віртуозних моментів. Ігри цього жанру можуть бути від першої або третьої особи, можуть включати елементи стрілянини, бойових мистецтв або швидких реакційних дій [2].

У свою чергу, стратегічні ігри розвиваються в напрямку більшої складності та стратегічної глибини. Гравці можуть керувати цілими цивілізаціями або арміями, планувати військові операції або розвивати економіку своєї країни [3].

Рольові ігри також відомі своєю різноманітністю. Вони можуть бути одиночними чи мультиплеєрними, фокусуватися на історії або геймплейному досвіді, мати відкритий світ або бути лінійними [5].

Головоломки та інші ігри засновані на логіці та розв’язанні завдань також знаходять своїх шанувальників. Вони можуть бути простими іграми для мобільних пристроїв або складними головоломками для консолей та ПК [4].

Зрештою існують нові жанри, що виникають або стають популярними завдяки новаторським підходом до геймдизайну. Наприклад ігри-виживання, ігри з віртуальною реальністю, або ігри з великим відкритим світом.

Загалом, аналіз сучасних ігрових жанрів допомагає не лише зрозуміти різноманітність ігрової індустрії, а й виявити тенденції її розвитку та вплив на культуру в цілому.

1.2 Аналіз аналогічних ігрових продуктів

У період попередньої стадії розробки велике значення має проведення аналізу існуючих ігрових продуктів, що діють у схожому жанрі. Цей етап дозволяє отримати уявлення про те, які рішення вже існують на ринку і як можна покращити власний продукт. Аналіз дозволяє виявити недоліки конкурентів і звернути увагу на прогалини, які можна заповнити власними ідеями. Це допомагає зорієнтуватися в вподобанні гравців і підготувати оптимальну стратегію розвитку власної гри. Інноваційний підхід та унікальні функціональні можливості допоможуть привернути увагу нових користувачів і створити власну нішу на ринку ігрових продуктів. В якості аналізуємих ігор було виділено наступні проєкти:

1) dig or Die (див. табл. 1.1-1.2);
2) blood Harvest 3 (див. табл. 1.3-1.4);
3) dome Keeper (див. табл. 1.5-1.6).

Таблиця 1.1 – Аналіз ігрового застосунку Dig or Die

Параметр	Dig or Die [6]	Опис
Жанр	Виживання	Гра заснована на принципі виживання в умовах віддаленого планетарного світу.
Графіка	2D	Представлена у двовимірному просторі з деталізованою анімацією та стильними пейзажами.
Геймплей	Стратегічний	Гравцеві потрібно планувати свої дії, та вибирати оптимальні місця для будівництва
Механіка	Крафтинг	Можливість виготовлення інструментів, обладунків та будівельних матеріалів.

Кінець таблиці 1.1

Мультиплеєр	Є	Наявний мультиплеєрний режим дозволяє гравцям спільно виживати та взаємодіяти у світі гри.
Керування	Мишка, клавіатура, геймпад	Гра підтримує управління як за допомогою миші та клавіуатури, так і за допомогою геймпада.
Оцінка користувачів	4.5/5	Середня оцінка гравців на ігрових платформах та відгуки відзначають високу якість гри та захоплюючий геймплей.

Dig or Die [6] – це захоплююча гра у жанрі виживання, де гравці мають виживати у віддаленому планетарному світі, цей ігровий проєкт має двовимірну графіку з деталізованою анімацією та стильними пейзажами. Геймплей потребує стратегічного планування і вибору оптимальних місць для будівництва, а крафтинг є важливою механікою, що дозволяє виготовляти різноманітні інструменти, обладунки та будівельні матеріали. Гра підтримує мультиплеєрний режим, який дозволяє гравцям спільно виживати та взаємодіяти, і має управління за допомогою миші, клавіатури або геймпада. Високі оцінки користувачів, які становлять 4.5/5, підкреслюють її високу якість та захоплюючий геймплей.

Таблиця 1.2 – Переваги та недоліки ігрового застосунку Dig or Die

Переваги	Недоліки
Захоплюючий геймплей та стратегічна глибина	Можливість повторності в геймплеї, особливо на пізніх етапах гри
Різноманітність крафтингу та можливості будівництва	Деяка незручність у керуванні та інтерфейсі гри

Кінець таблиці 1.2

Наявність мультиплеєрного режиму для спільної гри	

Dig or Die [6] має ряд переваг і недоліків. Серед переваг виділяються захоплюючий геймплей зі стратегічною глибиною, різноманітність крафтингу та будівельних можливостей, а також наявність мультиплеєрного режиму для спільної гри. Проте гра має і свої недоліки, такі як можливість повторності в геймплеї на пізніх етапах та незручність у керуванні та інтерфейсі. Незважаючи на ці недоліки, загальна оцінка гравців свідчить про високу якість та захопливість ігрового процесу (рис. 1.1).

Рисунок 1.1 – Гра «Dig or Die»

На скріншоті зображено геймплей гри Dig or Die [6], де гравець досліджує та виживає у ворожому світі. На передньому плані видно персонажа, який відбивається від ворогів в підземеллі, оточеному різноманітними структурами та ворогами. У грі акцент зроблено на інженерії та будівництві, де гравці повинні використовувати зібрані матеріали для створення захисних споруд та різноманітних пристроїв, щоб вижити під час нічних нападів агресивних істот.

Крім того, у грі є різноманітність біомів та зміна погодних умов, які впливають на геймплей і додають викликів гравцю.

Таблиця 1.3 – Аналіз ігрового застосунку Blood Harvest 3

Параметр	Blood Harvest 3 [7]	Опис
Жанр	RPG	Гра заснована на рольовій грі, де гравці відчувають себе у ролі персонажів.
Графіка	3D	Графічний стиль використовує тривимірний простір з деталізованою графікою та реалістичними об’єктами.
Геймплей	Екшн, пригоди	Гравці зустрінуться з інтенсивними екшн сценами та захоплюючими пригодами у світі гри.
Механіка	Крафтинг, дослідження, зброя та магія	В грі присутні можливості крафтингу предметів, дослідження світу, бойові механіки з використанням різноманітної зброї та магії.
Мультиплеєр	Є	Існує можливість грати у режимі мультиплеєра, де гравці можуть спільно виживати та взаємодіяти у світі гри.
Керування	Мишка, клавіатура, геймпад	Гра підтримує різні методи управління, включаючи мишу, клавіатуру та геймпад.
Оцінка користувачів	4.2/5	Середня оцінка користувачів на різних ігрових платформах, яка відзначає високу якість гри та захоплюючий геймплей.

Blood Harvest 3 [7] – це рольова гра (RPG) з тривимірною графікою, що занурює гравців у детально опрацьований світ з реалістичними об’єктами. Геймплей поєднує в собі інтенсивні екшн сцени та захоплюючі пригоди. Гравці

мають можливість займатися крафтингом, досліджувати світ, а також використовувати різноманітну зброю та магію у боях. Гра підтримує мультиплеєрний режим, що дозволяє спільну взаємодію та виживання у грі. Управління здійснюється за допомогою миші, клавіатури або геймпада. Користувачі високо оцінили гру, надавши їй середню оцінку 4.2/5, що свідчить про її якість та захопливий ігровий процес.

Таблиця 1.4 – Переваги та недоліки ігрового застосунку Blood Harvest 3

Переваги	Недоліки
Глибокий сюжет та атмосфера	Нестабільність гри, збої та відключення під час гри
Різноманітність ігрових локацій	Високі вимоги до системи
Широкий вибір персонажів та їхніх навичок	Можливість затримок в релізах оновлень
Велика кількість завдань та секретів	

Blood Harvest 3 [7] має кілька вагомих переваг і недоліків. Серед переваг відзначають глибокий сюжет та атмосферу гри, різноманітність ігрових локацій, широкий вибір персонажів та їхніх навичок, а також велику кількість завдань і секретів для дослідження. Водночас гра має недоліки, такі як нестабільність із частими збоями та відключеннями, високі вимоги до системи, що можуть обмежити доступність для деяких гравців, та можливі затримки в релізах оновлень. Незважаючи на ці недоліки, гра все одно отримала позитивні відгуки від користувачів завдяки своїм сильним сторонам (рис. 1.2).

Рисунок 1.2 – Гра «Blood Harvest 3»

На скріншоті зображено геймплей гри Blood Harvest 3, де гравець керує персонажем у тривимірному світі з деталізованою графікою. На передньому плані видно інтенсивну бойову сцену з використанням магії та зброї проти ворогів. Гравець досліджує різноманітні локації, збирає ресурси та виконує завдання. На екрані також відображаються елементи інтерфейсу, включаючи індикатори здоров'я, мани та доступні здібності персонажа. Загальна атмосфера гри передає глибокий сюжет і захоплюючі пригоди в світі Blood Harvest 3.

Таблиця 1.5 – Аналіз ігрового застосунку Dome Keeper

Параметр	Dome Keeper [8]	Опис
Жанр	Tower Defense	Гра заснована на жанрі Tower Defense, де гравцям потрібно захищати свій дім від нападів ворожих сил.
Графіка	2D	Графічний стиль використовує двовимірний простір з деталізованими об’єктами та кольоровими ефектами.

Кінець таблиці 1.5

Геймплей	Стратегічний	Гравцям потрібно ретельно планувати розміщення своїх оборонних структур та ефективно використовувати ресурси для успішного захисту дому.
Механіка	Будівництво, розвиток	Частиною гри є будівництво та розвиток оборонних структур
Мультиплеєр	Немає	Тільки одиночний режим гри
Керування	Мишка, клавіатура	Гра підтримує керування як за допомогою миші, так і за допомогою клавіатури.
Оцінка користувачів	4.0/5	Середня оцінка гравців на різних ігрових платформах, що свідчить про задоволення від гри та інтригуючий геймплей.

Dome Keeper [8] – це захоплююча гра у жанрі Tower Defense, де гравці мають захищати свій дім від ворожих нападів. Гра представлена у двовимірному графічному стилі з деталізованими об’єктами та яскравими ефектами. Геймплей вимагає стратегічного планування розміщення оборонних структур та ефективного використання ресурсів для успішного захисту. Основними механіками є будівництво та розвиток оборонних споруд. Гра підтримує лише одиночний режим і керування здійснюється за допомогою миші та клавіатури. Середня оцінка користувачів 4.0/5 свідчить про задоволення від гри та її інтригуючий геймплей.

Таблиця 1.6 – Переваги та недоліки ігрового застосунку Dome Keeper

Переваги	Недоліки
Глибокий сюжет та атмосфера	Нестабільність гри
Різноманітність ігрових локацій	Високі вимоги до системи
Широкий вибір персонажів та їхніх навичок	Можливість затримок в релізах оновлень
Велика кількість завдань та секретів	

Dome Keeper [8] пропонує гравцям глибокий сюжет та захоплюючу атмосферу, різноманітні ігрові локації, широкий вибір персонажів з унікальними навичками, а також велику кількість завдань та секретів для дослідження. Проте гра має деякі недоліки, такі як нестабільність із частими збоями, високі вимоги до системи, що можуть обмежити доступність для деяких користувачів, та можливі затримки в релізах оновлень. Незважаючи на ці недоліки, гра отримала позитивні відгуки від користувачів, які високо оцінили її сильні сторони (рис. 1.3).

Рисунок 1.3 – Гра «Dome Keeper»

На скріншоті зображено геймплей гри Dome Keeper, що відноситься до жанру Tower Defense. Гравець виступає у ролі захисника, який повинен будувати та розвивати оборонні структури для захисту свого дому від натиску ворожих сил. Гра пропонує двовимірну графіку з кольоровими ефектами та деталізованими об'єктами. Завдяки стратегічному плануванню розміщення оборонних структур та ефективному використанню ресурсів, гравець зможе забезпечити успішний захист свого дому від ворогів.

1.3 Специфікація вимог та аналіз ігрового застосунку, що розробляється

Проєкт «Reverse» - це 2D гра що розробляється на платформі Godot, яка зосереджена на екшні та швидкій стрільбі. Гравцям належить зачищати локації від ворожих сил, використовуючи різноманітні зброї та винахідливі стратегії. Гра розробляється з метою надати користувачам захоплюючий геймплей та можливість відчути адреналін від боротьби з ворогами. Кожний рівень пропонує нові виклики та завдання, що робить гру захоплюючою та непередбачуваною. Керування гравцем виконується з легкістю за допомогою миші та клавіатури, забезпечуючи насичений геймплей (див. табл. 1.7).

Таблиця 1.7 – Опис функціоналу системи гри «Reverse»

Параметр	Опис
Назва проєкту	«Reverse»
Платформа	Godot
Жанр	Екшн, шутер.
Орієнтація	2D
Головний фокус	Головна увага спрямована на екшн та швидку стрільбу, де гравцеві належить зачищати локації від ворожих сил

Кінець таблиці 1.7

Механіка	Гра пропонує широкий вибір зброї, кожен рівень пропонує нові виклики та завдання для гравця
Геймплей	Непередбачуваний та захоплюючий геймплей, де кожен рівень відзначається новими викликами та можливостями
Керування	Зручне керування гравцем здійснюється за допомогою миші та клавіатури, забезпечуючи легкий доступ до управління.

«Reverse» - це захоплюючий екшн-шутер, розроблений на платформі Godot у 2D орієнтації. У грі головний акцент робиться на швидкій стрільбі та екшні, де гравцеві доводиться зачищати локації від ворожих сил, використовуючи різноманітну зброю та стратегії. Широкий вибір зброї та непередбачуваний геймплей з новими викликами на кожному рівні роблять гру захоплюючою та цікавою. Зручне керування гравцем за допомогою миші та клавіатури дозволяє легко керувати персонажем та насолоджуватися ігровим процесом.

Призначення гри, для якої розробляється програмне забезпечення

Призначення гри «Reverse» є створення захоплюючого ігрового досвіду, зосередженого на швидкій інтенсивній стрільбі та екшні.

Погодження, ухвалені в програмній документації

У програмній документації було ухвалено використання платформи Godot для розробки гри «Reverse» з метою забезпечення ефективного та зручного інструменту для розробки ігор.

Межі проєкту

Крайня дата завершення робіт над проєктом – 03.05.2024

Сфера застосування

Гра розрахована на широке коло користувачів, які зацікавлені в екнш-іграх з інтенсивною стрільбою та швидким геймплеєм.

Характеристики користувачів

Користувачу необхідні стабільний доступ до мережі Інтернет, клавіатура та мишка, ноутбук чи персональний комп’ютер з монітором.

Загальна структура та склад системи

Система «Reverse» включає наступні компоненти:

1) гра на базі платформи Godot, що відповідає за відтворення геймплею та графічні ефекти;
2) редактор рівнів для створення та редагування ігрових рівнів та локацій;
3) двигун фізики для симуляції руху об’єктів та взаємодії у грі;
4) інструменти для розробки графічних ресурсів, звуків та музики для гри;
5) система управління, що дозволяє гравцям керувати персонажем та взаємодіяти з грою за допомогою клавіатури та миші.

Обмеження

Головним обмеженням гри «Reverse» є необхідність стабільного доступу до мережі інтернет для користувачів.

Функції ігрового додатку «Reverse**»**

Функція управління налаштуванням гри

Опис функції

Ця функція дозволяє гравцеві настроювати параметри гри, такі як рівень складності, гучність звуку, розмір екрану. Вона забезпечує гнучкість і зручність для користувача, дозволяючи адаптувати гру під свої потреби та вподобання.

Вхідна та вихідна інформація

1) вхідна інформація: налаштування, які користувач бажає змінити;
2) вихідна інформація: підтвердження внесених змін та їхнє відображення у грі.

Функціональні вимоги

Функція має забезпечити можливість зміни налаштувань гри в реальному часі, а також зберігання змін для майбутнього користування.

Функція генерації випадкового рівня

Опис функції

Ця функція відповідає за створення нового рівня гри з випадковими елементами, що додає різноманіття та цікавість до геймплею. Вона забезпечує можливість перегравати рівень гри, оскільки кожен новий рівень буде унікальним.

Вхідна та вихідна інформація

1) вхідна інформація: параметри для генерації рівня;
2) вихідна інформація: створений випадковий рівень, який може бути використаний для гри.

Функціональні вимоги

Функція повинна забезпечувати генерацію рівнів з урахуванням встановлених параметрів та забезпечувати відповідну складність та рівновагу гри.

Джерела та зміст вхідної інформації

У ігровому застосунку «Reverse» головним джерелом інформації є сам гравець. Він створює нові дані шляхом виконання різних дій у грі, таких як реєстрація облікового запису та проходження рівнів.

Нормативно-довідкова інформація

Для розроблюваної гри відсутні вимоги до нормативно-довідкової інформації, таких як класифікація або довідники.

Вимоги до технічного забезпечення

Вимоги до технічного забезпечення гри «Reverse» не є суворими в сучасних умовах. Гаджет, на якому буде виконуватись гра, повинен мати доступ до мережі інтернет, екран, підтримку фізичної або екранної клавіатури.

Вимоги до програмного забезпечення

Архітектура програмної системи

Система складатиметься з головного движка Godot для розробки ігрового середовища, включаючи редактор і ресурси для створення гри, а також власне ігрового додатку.

Системне програмне забезпечення

1) використання платформи Godot для розробки гри, яка підтримує мови програмування, такі як GDScript, C#, та VisualScript;

2) зручне інтегроване середовище розробки для Godot, забезпечене різноманітними інструментами для створення графіки, анімацій, звуку та ігрової логики.

Мережне програмне забезпечення

1) операційна система з підтримкою Godot, така як Windows, macOS або Linux;

2) рекомендований редактор коду або інтегроване середовище розробки, сумісне з Godot, таке як Visual Studio Code або GDScript IDE.

Програмне забезпечення ведення інформаційної бази

Godot забезпечує можливість збереження інформації про гру та її стан безпосередньо в проєкті гри.

Мова і технологія розробки ПЗ

1) використання мов програмування, які підтримуються Godot, такі як GDScript, C# та VisualScript;

2) використання Godot Engine для створення ігрового середовища та інтеграції всіх елементів гри.

Вимоги до зовнішніх інтерфейсів:

Інтерфейс користувача

Інтерфейс користувача гри «Reverse» повинен відповідати всім стандартам для забезпечення найкращого досвіду гравця. Головний інтерфейс повинен бути розділений на навігаційну панель, контейнер для відображення головного контенту та нижню панель. Під час гри, змінним є лише контейнер для головного контенту, що забезпечить зручну навігацію.

Апаратний інтерфейс

Апаратним інтерфейсом гри буде пристрій, на якому користувач буде грати, такий як персональний комп’ютер або ноутбук.

Програмний інтерфейс

Для розробки гри «Reverse» на платформі Godot будуть використовуватись функціональності та можливості самого движка Godot, зокрема мови програмування GDScript, C#, а також інші інструменти, доступні в Godot для створення інтерфейсу гри та її логіки.

Властивості програмного забезпечення:

Доступність

Гра «Reverse» має мати низький поріг початку користування, щоб користувачі могли легко зануритися у гру без необхідності вивчення складних інструкцій.

Продуктивність

Обсяги даних, що передаються під час гри, повинні бути невеликими, щоб навіть при низькій швидкості інтернету не виникало серйозних затримок під час гри.

Простота використання

Користувацький інтерфейс гри має бути простим та зрозумілим, щоб гравці могли легко засвоїти управління та основні правила гри.

Функціональність

Гра повинна мати широкий спектр функціоналу, спрямований на створення захоплюючого геймплею та задоволення потреб гравців у швидкій грі.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було проведено детальний аналіз існуючих конкурентів на ринку гри «Reverse». Цей аналіз дозволив ідентифікувати основні недоліки конкурентів та визначити шляхи до їх вирішення. Також була розглянута система, що розробляється, включаючи список основного функціоналу, сценарії роботи та використовувані технології. Була надана специфікація вимог до розроблюваної системи, що містить повний опис її поведінки, функціональні та нефункціональні вимоги, а також обмеження. Цей

розділ становить фундамент для подальшої розробки та реалізації гри «Reverse», надаючи чітке розуміння потреб користувачів та основних цілей проєкту.

В результаті аналізу конкурентів Dome Keeper, Blood Harvest 3 та Dig or Die було виявлено кілька основних недоліків, що можуть бути важливими для користувачів. Наприклад, у Dome Keeper було виявлено відсутність мультиплеєрного режиму, що може обмежити соціальний аспект гри для гравців, а в Blood Harvest 3 зазначено нестабільність гри та високі вимоги до системи, що може вплинути на загальний досвід гравців та їхню задоволеність.

Детальний аналіз цих недоліків дозволив визначити шляхи до їх вирішення у розроблюваній грі «Reverse». Наприклад, для покращення враження від гри може бути реалізовано стабільну оптимізацію гри, щоб забезпечити плавний геймплей навіть на менш потужних пристроях. Також може бути врахована можливість додавання мультиплеєрного режиму, щоб створити більше можливостей для соціальної взаємодії між гравцями.

Специфікація вимог до розроблюваної системи, що була надана у цьому розділі, визначає повний обсяг функціональності, який повинен бути реалізований у грі «Reverse», включаючи сценарії роботи та вимоги до використовуваних технологій. Цей розділ є важливим фундаментом для подальшої розробки та реалізації гри, оскільки він надає чітке розуміння потреб користувачів та основних цілей проєкту, що допомагає забезпечити успішну реалізацію та задоволення від гри для користувачів.

2 МОДЕЛЮВАННЯ ІГРОВОГО ЗАСТОСУНКУ

2.1 Діаграма варіантів використання

Короткий use case

Користувач грає у гру «Reverse», щоб насолоджуватися геймплеєм та покращити свої навички у керуванні героєм та стрільбі. Він від”криває гру та переходить до головного меню, де обирає складність рівня гри. Після вибору налаштувань, він починає грати. Після закінчення гри він може переглянути свої досягнення та зберегти прогрес.

Поверхневий use case

Головний сценарій (успішний):

Користувач має бажання насолоджуватися грою та покращити свої навички. Він відкриває гру «Reverse», обирає складність та починає грати.

Альтернативні сценарії:

1) користувач не має облікового запису у системі. Він все одно може насолоджуватися грою, але йому може бути недоступна можливість збереження результатів або перегляду своїх досягнень на сервері;

2) користувач вибирає режим гри але втрачає з'єднання з інтернетом на більш ніж 30 секунд. Гра може призупинитися або відключити можливість збереження результатів, поки з'єднання не буде відновлено;

3) користувач забув свій пароль. Він може скористатися функцією відновлення паролю, якщо ця опція доступна;

4) сервер, на якому гра розміщена, має технічні негаразди. Гра може працювати в обмеженому режимі або бути тимчасово недоступною;

5) користувач випадково закриває вікно гри. Він може знову відкрити гру та продовжити грати з того місця, де він залишився.

Повний use case

Таблиця 2.1 – Повний use case

Scope	Гра «Reverse» - 2D гра-шутер у якій гравець стріляє на рухливих ворогів та уникає їх атак.
Level	Мета користувача (user-goal)
Primary	Actor Гравець
Preconditions	Встановлення та запуск гри на пристрої
Stakeholders and interests	1) гравець: зацікавлений у відтворенні гри та отриманні задоволення від геймплею; 2) розробник гри: зацікавлений у створенні цікавої та добре функціонуючої гри для користувачів.
Main Success Scenario	1) гравець запускає гру «Reverse»; 2) гравець обирає рівень складності; 3) гравець відправляється в гру та починає стріляти та уникати ворогів; 4) гравець досягає кінцевої мети рівня (наприклад, вбиває всіх ворогів або долає боса); 5) гравець переходить до наступного рівня або завершує гру.
Result	Гравець зазнає задоволення від гри, можливо, покращує свої навички стрільби та уникання
Extensions	1) гравець закриває гру: Гравець може почати гру знову з останнього збереженого місця або повернутися в головне меню; 2) технічні проблеми з грою або пристроєм, які перешкоджають гравцеві продовжувати грати;

Кінець таблиці 2.1

	3) гравець вибирає вийти з гри; 4) гравець вибирає використання додаткових можливостей (наприклад, відключення звуку або зміна мови); 5) гравець вибирає перегляд досягнень чи статистики після закінчення гри.
Special Requirements	1) наявність пристрою, на якому можна запустити гру; 2) можливість взаємодії з грою за допомогою клавіатури або іншого введення.
Frequency of Occurrence	Система повинна бути здатна обробляти дані з великою частотою

Гра «Reverse» - це захоплюючий 2D шутер, в якому гравець керує персонажем, який стріляє на рухливих ворогів та уникає їх атак. Основна мета гравця - досягти кінцевої мети рівня, вбиваючи ворогів або долаючи боса, та просунутися до наступного рівня. Головний акцент гри зосереджений на екшні та стрільбі, пропонуючи непередбачуваний та захоплюючий геймплей. Гравець може вибрати рівень складності та взаємодіяти з грою за допомогою клавіатури та миші. Гра задовольняє користувача та допомагає покращувати навички стрільби та уникання ворожих атак.

2.2 Діаграма послідовності

Unified Modeling Language (UML) - це стандарт для специфікації програмного забезпечення, регульований Object Management Group. Цей стандарт визначає набір правил і термінів для опису системного моделювання. Використання UML надає графічні елементи для моделювання різних аспектів системи.

У процесі аналізу системи важливо враховувати способи взаємодії користувачів з програмним забезпеченням. Для моделювання цієї взаємодії часто

використовуються діаграми взаємодії, такі як діаграми послідовності. Наприклад, на діаграмі №1 (рис. 2.1) можна побачити взаємодію користувача з системою під час проходження рівня гри.

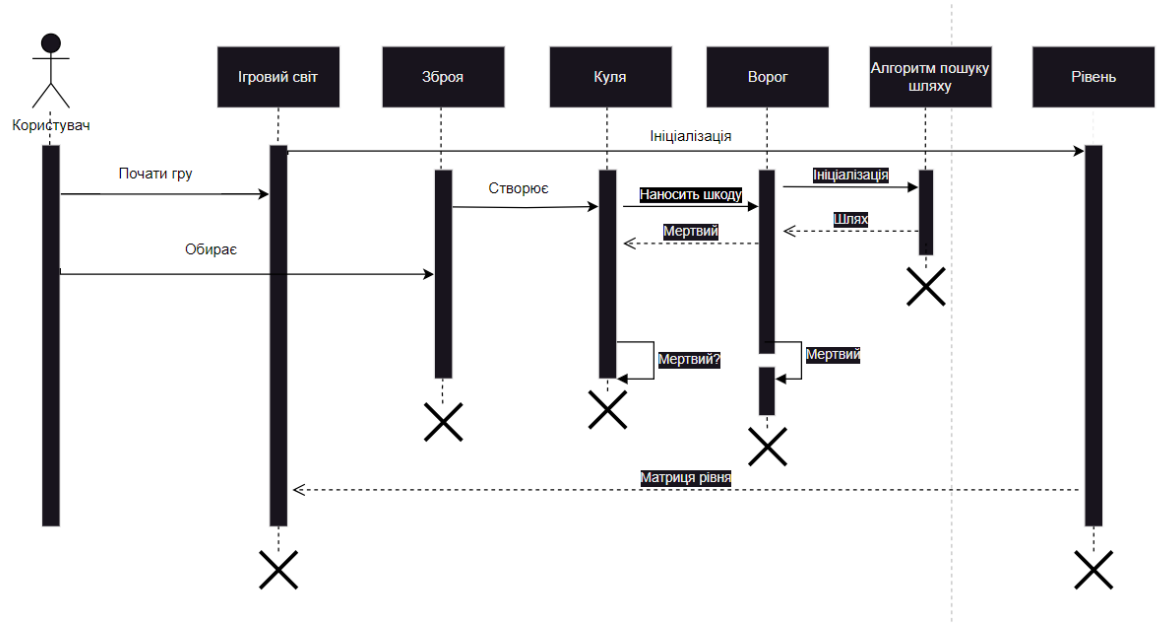

Рисунок 2.1 – Діаграма послідовності гри «Reverse»

На діаграмі послідовності зображено сценарій взаємодії користувача із грою, починаючи з моменту вибору зброї і закінчуючи поведінкою ворога. Процес починається, коли користувач запускає гру та обирає зброю. Далі обрана зброя створює кулю, яка рухається в напрямку ворога. При потраплянні кулі у ворога, відбувається нанесення шкоди. Після цього система перевіряє, чи ворог залишився живий, або ж він загинув від отриманих ушкоджень. Якщо ворог не мертвий, він продовжує рух за алгоритмом пошуку шляху, що визначає його подальшу поведінку в ігровому світі. Ця діаграма демонструє послідовність дій та взаємодій між користувачем, зброєю, кулею та ворогом, забезпечуючи чітке уявлення про логіку ігрового процесу.

2.3 Діаграма діяльності

Діаграми діяльності використовуються для опису можливих процесів всередині системи та подій, які відбуваються під час цих процесів. Вони є

аналогами блок-схем і зображуються у вигляді графів, де вершини представляють дії, а ребра – переходи між цими діями.

Для ігрового додатку, а саме 2D гри «Reverse» було розроблено 3 діаграми діяльності:

1) діаграма діяльності гравця (Player Activity Deagram) (рис. 2.2);
2) діаграма діяльності ворогів (Enemy Activity Diagram) (рис. 2.3);
3) діаграма діяльності арсеналу (Weaporny Activity Diagram) (рис. 2.4).

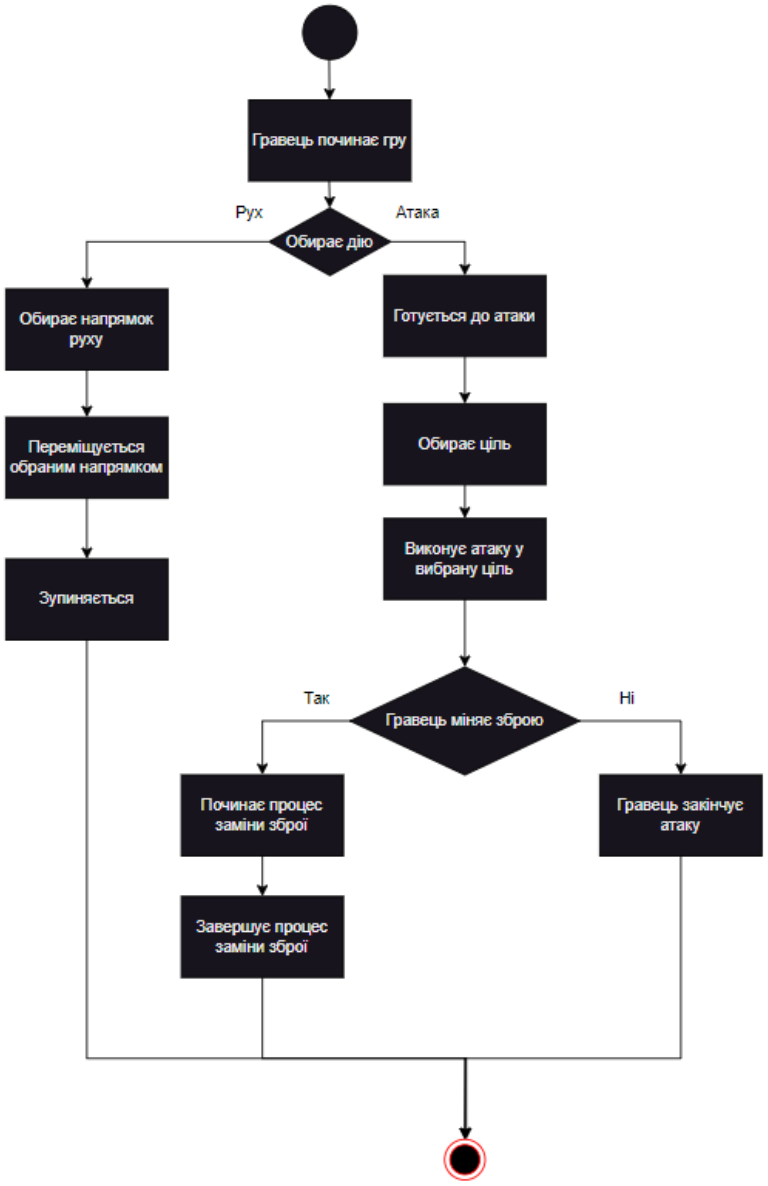

Рисунок 2.2 – Діаграма діяльності гравця

Гравець починає гру та обирає між двома основними діями: рухом або стрільбою. Якщо обрано рух, гравець визначає напрямок руху та переміщується обраним шляхом. Після досягнення місця призначення він зупиняється. У випадку обраної стрільби, гравець готується до неї та обирає ціль. Після вибору цілі він виконує постріл у вибрану мішень. Якщо гравець натискає клавішу перезарядки, він розпочинає процес перезарядки зброї та завершує його. У випадку, якщо клавішу перезарядки не натиснуто, гравець припиняє стріляти. Кінець дії.

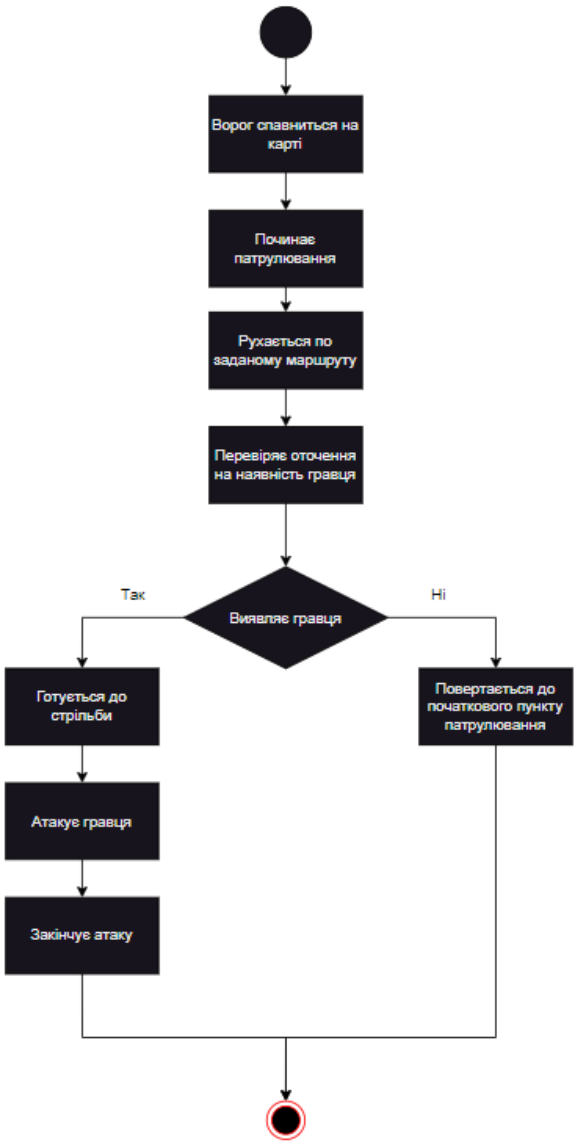

Рисунок 2.3 – Діаграма діяльності ворогів

Ворог спавниться на карті та розпочинає свою діяльність з патрулювання визначеної області. Він рухається по заданому маршруту, відслідковуючи оточення на наявність гравця. Якщо ворог виявляє гравця, він готується до стрільби, відкриває вогонь у напрямку гравця та продовжує стріляти до закінчення атаки. У разі, якщо гравця не виявлено, ворог повертається до свого початкового пункту патрулювання. Кінець дій.

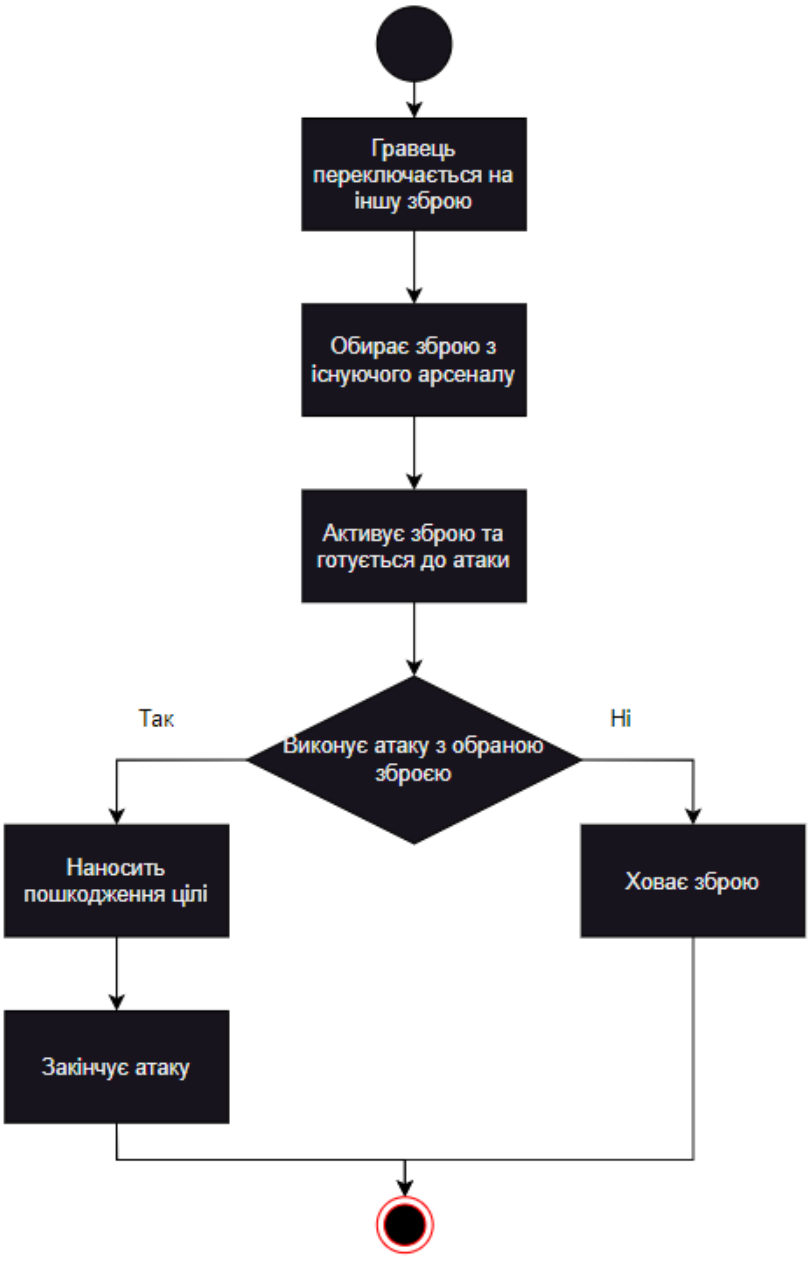

Рисунок 2.4 – Діаграма діяльності арсеналу

Гравець може переключатися на іншу зброю, обираючи її з існуючого арсеналу. Після обрання зброї він активує її та готується до стрільби. Якщо гравець вирішує виконати атаку з обраною зброєю, він наносить пошкодження цілі та завершує атаку. У випадку, якщо гравець не виконує атаку з обраною зброєю, він ховає зброю.

Висновки до розділу 2

Другий розділ кваліфікаційної роботи присвячено питанням моделювання різних аспектів гри. Використовуючи діаграми використання та діаграми діяльності було відображено основні сценарії роботи.

Діаграми діяльності надають можливість візуально відобразити алгоритми руху гравця, механіку стрільби та обробку взаємодії з ворогами. Це дозволяє зрозуміти, як саме взаємодіють різні елементи гри та які кроки виконуються під час виконання певних дій у грі. Наприклад, діаграма діяльності може проілюструвати процес виявлення ворогів гравцем та виконання стрільби у відповідь на їхню атаку.

За допомогою діаграм взаємодії описуються принципи спілкування між різними об’єктами гри, такими як гравець та вороги. Це дозволяє краще розібратися в механіках гри та визначити, як саме взаємодіють між собою різні елементи, що сприяє покращенню якості геймплею та загального досвіду гравця.

3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ІГРОВОГО ЗАСТОСУНКУ

3.1 Ігровий рушій та середовище розробки

У даному розділі ми проведемо детальний аналіз геймдвижка Godot Engine як потенційного інструмента для розробки 2D шутера «Reverse». Ціль аналізу полягає в тому, щоб краще розуміти можливості цього геймдвижка, його переваги для створення 2D ігор та порівняти його з альтернативами. Це дозволить нам зробити обґрунтований вибір та обрати оптимальний інструмент для подальшої розробки гри «Reverse». Такий підхід допоможе забезпечити успішну і ефективну реалізацію проєкту, враховуючи його потреби та вимоги до функціональності та продуктивності.

Проведемо аналіз рушія Godot Engine (див. табл. 3.1-3.2):

Таблиця 3.1 – Аналіз рушія

	Параметр	Godot Engine
1	Відкритий код	Так
2	Мови програмування	GDScript, C#, C++ (часткова підтримка)
3	Платформи	Windows, macOS, Linux, iOS, Android, HNML5
4	Редактор	Інтегрований редактор Godot
5	Графічний движок	Орієнтований на 2D, також має 3D можливості
6	Ком’юніті	Активна та дружня
7	Навчання	Документація, відеоуроки, спільнота
8	Розширюваність	Можливість розширення функціоналу через плагіни та розширення
9	Продуктивність	Легкий та швидкий в роботі, ефективний ресурсозберігаючий движок
10	Вартість	Безкоштовний, вільний для використання у комерційних проєктах
11	Підтримка мобільних пристроїв	Є підтримка мобільних пристроїв

Кінець таблиці 3.1

12	Візуальний редактор	Інтегрований, з можливістю роботи у власних редакторах
13	Відладка	Вбудований дебагер та інструменти відладки
14	Фізика	2D та 3D фізика
15	Звук	Повна підтримка аудіо, можливість роботи з аудіо ресурсами
16	Анімація	Повна система анімації з підтримкою скелетної анімації та іншими видами
17	Мережевий код	Підтримка мережевого програмування

Godot Engine - це потужний і безкоштовний ігровий рушій з відкритим вихідним кодом, який надає широкі можливості для розробки як 2D, так і 3D ігор. Він підтримує різні мови програмування, такі як GDScript, C# і C++, та працює на різних платформах, включаючи Windows, macOS, Linux, а також мобільні платформи iOS та Android. Графічний движок Godot орієнтований на 2D, але також має можливості для 3D розробки. Його інтегрований редактор, сприятлива спільнота та різноманітні навчальні ресурси, такі як документація та відеоуроки, роблять його привабливим вибором для початківців та досвідчених розробників. Godot також відомий своєю продуктивністю, ефективністю та можливістю розширення функціоналу за допомогою плагінів та розширень. Вбудований дебагер та інструменти відладки спрощують процес розробки, а повна підтримка звуку, фізики та анімації робить Godot Engine повноцінним інструментом для створення ігор різного рівня складності. Завдяки активному розвитку та постійним оновленням, Godot Engine стає все більш популярним серед розробників по всьому світу. Його гнучкість і доступність роблять його ідеальним для реалізації різних творчих ідей.

Таблиця 3.2 – Переваги та недоліки

	Переваги	Недоліки
1	Відкритий вихідний код	Обмежена підтримка C++
2	Безкоштовний	Деякі обмеження у продуктивності
3	Можливість розширення	Менша кількість готових ресурсів
4	Інтегрований редактор	Можливі проблеми із сумісністю
5	Підтримка різних мов	Менша кількість досвідчених розробників
6	Підтримка мобільних платформ	

Godot Engine відкриває безліч можливостей для розробки ігор, особливо у жанрі 2D. Його безкоштовність, відкритий вихідний код та широкий функціонал роблять його привабливим вибором для розробників. Інтегрований редактор, підтримка різних мов програмування та розширення функціоналу через плагіни роблять процес розробки зручним та ефективним. Необхідно також враховувати активну та дружню спільноту, яка може надати підтримку та вирішити проблеми. Однак, варто враховувати обмежену підтримку мови програмування C++ та деякі обмеження у продуктивності, а також можливі проблеми зі сумісністю та меншу кількість готових ресурсів порівняно з іншими геймдвижками. Усе це слід ураховувати при виборі Godot Engine для розробки гри.

Порівняння з альтернативами

Порівняльний аналіз Godot Engine з Unity та Unreal Engine допоможе краще зрозуміти їхні особливості та визначити найбільш підходящий для конкретного проєкту ігровий рушій. У цій таблиці представлено основні характеристики кожного движка, такі як підтримувані мови програмування, редактори,

продуктивність, вартість та інші параметри, що можуть бути важливими для розробки 2D шутеру (див. табл. 3.3).

Таблиця 3.3 – Порівняння та аналіз

Параметр		Godot Engine	Unity	Unreal Engine
1	Відкритий код	Так	Ні	Ні
2	Мови програмування	GDScript, C#, C++	C#, JavaScript, Boo, C++	C++, Blueprint, C#
3	Платформи	Windows, macOS, Linux, iOS, Android	Windows, macOS, Linux, iOS, Android, WebGL	Windows, macOS, Linux, iOS, Android
4	Редактор	Інтегрований редактор Godot	Інтегрований редактор Unity	Інтегрований редактор Unreal
5	Графічний движок	Орієнтований на 2D також має 3D можливості	3D та 2D	3D та 2D
6	Ком’юніті	Активна та дружня	Активна, велика спільнота	Активна, велика спільнота
7	Навчання	Документація, відеоуроки, спільнота	Документація, відеоуроки, спільнота	Документація, відеоуроки, спільнота
8	Розширюваність	Можливість розширення функціоналу через плагіни та розширення	Можливість використання плагінів та активної спільноти	Можливість використання плагінів та активної спільноти
9	Продуктивність	Легкий та швидкий в роботі, ефективний ресурсозберігаючий движок	Забезпечую високу продуктивність та швидкість	Забезпечую високу продуктивність та швидкість

Кінець таблиці 3.3

10	Вартість	Безкоштовний, вільний для використання у комерційних проєктах	Безкоштовний, але з платними планами для підприємства та великих команд	Безкоштовний, але з платними планами для підприємства та великих команд
11	Підтримка мобільних пристроїв	Є підтримка платформ мобільних пристроїв	Є підтримка платформ мобільних пристроїв	Є підтримка платформ мобільних пристроїв
12	Візуальний редактор	Інтегрований, з можливістю роботи у власних редакторах	Інтегрований, з можливістю роботи у власних редакторах	Інтегрований, з можливістю роботи у власних редакторах
13	Відладка	Вбудований дебагер та інструменти відладки	Вбудований дебагер та інструменти відладки	Вбудований дебагер та інструменти відладки

Порівняльний аналіз Godot Engine, Unity та Unreal Engine надає глибоке розуміння їхніх переваг та обмежень у контексті розробки 2D шутеру. Godot відрізняється відкритістю та безкоштовністю, має легкий та продуктивний ресурсозберігаючий движок, а також інтегрований візуальний редактор та дебагер. Unity вражає розширеною функціональністю та розширенням за рахунок широкого спектру плагінів, а Unreal Engine виділяється потужними 3D можливостями та високою продуктивністю. Кожен з них має свої унікальні переваги, і вибір між ними залежатиме від конкретних вимог та завдань проєкту.

Після ретельного аналізу альтернативних геймдвижків та їхніх можливостей, вирішено обрати Godot Engine для розробки 2D шутеру. Цей вибір обгрунтований кількома ключовими факторами. По-перше, Godot Engine

відкритий код і безкоштовний для використання у комерційних проєктах, що робить його доступним та економічно вигідним рішенням для розробки гри. Крім того, він має інтегрований редактор та дебагер, що спрощує процес розробки та налагодження.

У порівнянні з конкурентами, Godot Engine відзначається легкістю використання та продуктивністю, а також ефективним ресурсозберігаючим движком, що особливо важливо для розробки 2D шутерів. Його активна та дружня спільнота надає великий обсяг документації та відеоуроків, що сприяє швидкому вивченню та вирішенню можливих проблем.

Крім того, Godot Engine має повний набір функцій для розробки 2D ігор, включаючи підтримку аудіо, анімації, фізики та мережевого програмування. Його інтегрований візуальний редактор і підтримка мов програмування GDScript, C# та C++ забезпечують гнучкість та можливості розширення. Загалом, Godot Engine ідеально відповідає вимогам та цілям нашого проєкту, надаючи нам потужні інструменти для створення високоякісного та ефективного 2D шутеру.

3.2 Мова програмування

У цьому підрозділі проведемо аналіз різних мов програмування, які підтримуються Godot Engine, та їхнє використання для реалізації функціоналу гри.

Давайте розглянемо таблицю порівняння мов програмування, таких як GDScript, C# та VisualScript для реалізації функціоналу 2D шутеру (див. табл. 3.4):

Таблиця 3.4 – Порівняння мов програмування

Особливості		GDScript	C#	VisualScript
1	Швидкість	Середня	Висока	Низька
2	Зручність	Дуже зручна	Зручна	Дуже зручна
3	Інтеграція	Повна	Повна	Повна

Кінець таблиці 3.4

4	Підтримка	Офіційна	Офіційна	Офіційна
5	Екосистема	Обмежена	Розширена	Обмежена
6	Відладка	Повна	Повна	Повна
7	Спільнота	Активна	Активна	Активна
8	Навчання	Доступна	Доступна	Доступна
9	Переносимість	Платформонезалежна	Платформонезалежна	Платформонезалежна

Тепер давайте розглянемо другу таблицю, що стосується переваг та недоліків мов програмування для Godot Engine (див. табл. 3.5):

Таблиця 3.5 – Переваги та недоліки мов програмування

	GDScript	C#	VisualScript
Переваги			
Легка вивченість	Легко засвоюється, схожа на Python	Висока швидкість та продуктивність	Візуальний, простий для новачків
Інтеграція	Повністю інтегрована в Godot	Відмінна підтримка в середовищі VisualStudio	Інтегрована візуальна побудова графіків
Переносимість	Підтримується на всіх платформах Godot	Платформонезалежна	Підтримується на всіх платформах Godot

Кінець таблиці 3.5

Недоліки			
Обмежена швидкість	Повільніше за C#	Порівняно великий обсяг коду для виконання простих завдань	Складніше використання для складних функцій
Обмежена підтримка	Менше бібліотек та інтеграцій	Потреба встановлювати додаткове середовище розробки	Обмежені функціональні можливості

Ми детально проаналізували три основні мови програмування, які підтримуються Godot Engine для розробки 2D шутеру: GDScript, C# та VisualScript. Кожна з цих мов має свої переваги та недоліки, які варто враховувати при виборі для конкретного проєкту.

GDScript - легка для вивчення мова програмування, що добре інтегрована в середовище Godot. Вона підтримується на всіх платформах, на яких працює ігровий рушій, і забезпечує гнучкість у розробці завдяки широкому спектру вбудованих функцій.

C# - мова програмування, яка відрізняється високою швидкістю та продуктивністю, що особливо корисно для великих проєктів. Її інтеграція з середовищем Visual Studio забезпечує зручність у розробці та дебагу.

VisualScript - візуальна мова програмування, яка пропонує простий інтерфейс для новачків і відмінно підходить для складання простих логічних виразів. Однак для складних функцій може виявитися не так ефективною.

Кожна з цих мов має свої особливості, і вибір конкретної залежить від потреб проєкту, рівня володіння розробників та специфіки розробки.

3.3 Інтегровані інструменти розробки

Проведемо аналіз інструментів, які є складовою частиною інтегрованого середовища розробки (IDE) Godot Engine. Ці інструменти надають можливості для створення та налагодження різноманітних елементів гри, від візуальних компонентів до аудіо та програмного коду. Нижче подано огляд основних інтегрованих інструментів розробки у Godot:

1) редактор сцен;
2) редактор анімацій;
3) інструменти роботи з аудіо;
4) редактор ассетів;
5) модуль розробки скриптів;

Аналіз цих інструментів допоможе визначити їхню відповідність для конкретного проєкту та визначити стратегії їх використання для досягнення поставлених цілей розробки гри.

Редактор сцен

Першим кроком у створенні гри в Godot є створення сцен - просторів, де відбувається вся ігрова дія. Редактор сцен у Godot дозволяє розміщувати та налаштовувати об'єкти, налаштовувати їх властивості та взаємодію між ними. Для докладного розгляду цього важливого інструменту, розглянемо його переваги та недоліки (див. табл. 3.6):

Таблиця 3.6 – Переваги та недоліки

Редактор сцен	
Переваги	Недоліки
- інтуїтивний інтерфейс дозволяє швидко освоїти основні функції; - підтримка шарування об'єктів.	- обмежена функціональність для складних анімацій або роботи з великою кількістю об'єктів.

Цей інструмент є центральним для розробки гри в Godot. Він надає можливість візуально створювати та редагувати сцени, додавати об'єкти, налаштовувати їх параметри та взаємодію між ними.

Редактор анімацій

Щоб зробити гру живою та цікавою, потрібно додати до неї анімацію. Редактор анімацій у Godot дозволяє створювати рухи та ефекти для об'єктів у грі. Наразі ми розглянемо, як цей інструмент працює, а також його переваги та недоліки (див. табл. 3.7):

Таблиця 3.7 – Переваги та недоліки

Редактор анімацій	
Переваги	Недоліки
- простий у використанні, зручний інтерфейс; - можливість створювати складні анімації без програмування.	- обмежені можливості порівняно з професійними програмами для анімації.

Цей інструмент дозволяє створювати та редагувати анімаційні ефекти для об'єктів у грі. Він підтримує ключові кадри, інтерполяцію, криві руху та багато іншого.

Інструменти роботи з аудіо

Звук - важлива частина будь-якої гри, яка створює атмосферу та підсилює іммерсію. У Godot є власні інструменти роботи з аудіо, які дозволяють додавати та редагувати звукові ефекти, музику та інші аудіо-ресурси. Давайте детально розглянемо їхні можливості та обмеження (див. табл. 3.8):

Таблиця 3.8 – Переваги та недоліки

Інструменти роботи з аудіо	
Переваги	Недоліки
- легкість використання; - зручний інтерфейс; - підтримка різних форматів аудіо.	- обмежені можливості для складної обробки аудіо порівняно з професійними аудіо редакторами.

Ці інструменти дозволяють додавати та редагувати аудіо ефекти та музику в гру. Вони підтримують різні формати файлів та дозволяють контролювати гучність та інші параметри звуку.

Редактор ассетів

Графічні ресурси, такі як спрайти, текстури та анімації, грають важливу роль у візуальному вигляді гри. Редактор ассетів у Godot дозволяє керувати цими ресурсами, а також імпортувати нові. Давайте розглянемо його переваги та недоліки (див. табл. 3.9):

Таблиця 3.9 – Переваги та недоліки

Редактор ассетів	
Переваги	Недоліки
- простий у використанні; - підтримка різних форматів файлів; - можливість попереднього перегляду графіки.	- обмежені можливості для редагування та створення складних графічних ефектів.

Цей інструмент використовується для керування графічними ресурсами, такими як спрайти, текстури, шейдери та анімації.

Модуль розробки скриптів

Геймплей та логіка гри часто вимагають програмування. У Godot є власні інструменти розробки скриптів, які дозволяють програмувати геймплейну логіку та інтерактивність гри. Давайте розглянемо його можливості та обмеження (див. табл. 3.10):

Таблиця 3.10 – Переваги та недоліки

Модуль розробки скриптів	
Переваги	Недоліки
- гнучкість у виборі мови програмування; - можливість швидкого прототипування; - зручний інтерфейс.	- обмежені можливості порівняно з іншими інтегрованими середовищами розробки; - відсутність деяких продвинутих функцій та інструментів.

Цей модуль дозволяє програмувати геймплей та взаємодію об'єктів у грі. Він підтримує кілька мов програмування, таких як GDScript, C#, Python та VisualScript.

Висновки до розділу 3

У цьому розділі було детально розглянули різноманітні інструменти, що входять у склад інтегрованого середовища розробки (IDE) Godot Engine. Кожен із цих інструментів має свої унікальні можливості та обмеження, які важливо враховувати під час розробки гри.

Аналіз геймдвижка Godot Engine дозволив з'ясувати, що цей рушій має значні переваги у вигляді відкритого джерела, легкості в освоєнні та гнучкості у розробці 2D і 3D ігор. Порівняння з альтернативами, такими як Unity та Unreal Engine, показало, що Godot виявляється конкурентоспроможним рішенням, зокрема для невеликих або середніх проєктів, але може вимагати додаткових зусиль для складних ігрових проєктів.

Аналіз доступних мов програмування на Godot Engine (GDScript, C#, VisualScript) дозволив зрозуміти, що кожна мова має свої переваги та обмеження. GDScript відрізняється простотою та нативною інтеграцією з Godot, але може бути менш ефективним для великих проєктів. C# відкриває додаткові можливості для досвідчених розробників та забезпечує більшу швидкість роботи, але може вимагати додаткового часу на освоєння. VisualScript надає можливість програмувати без необхідності писати код, що робить його ідеальним для початківців та для швидкого прототипування.

Godot Engine - це потужний інструмент для розробки ігор, який надає широкий спектр можливостей для реалізації творчих ідей. Вибір конкретних інструментів та мов програмування повинен базуватися на потребах та специфікаціях кожного конкретного проєкту, а також на рівні володіння розробником цих інструментів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

4.1 Організація робочого середовища та основна структура проєкту

Організація робочого середовища та структура проєкту є важливими аспектами при розробці гри або додатку. Вона допомагає підтримувати порядок, спрощує навігацію та дозволяє легше керувати ресурсами. Нижче наведено опис структури каталогів та файлів у проєкті Godot 4:

Проєктний каталог: Головний каталог проєкту, який містить усі ресурси, сцени, скрипти та інші файли (рис. 4.1).

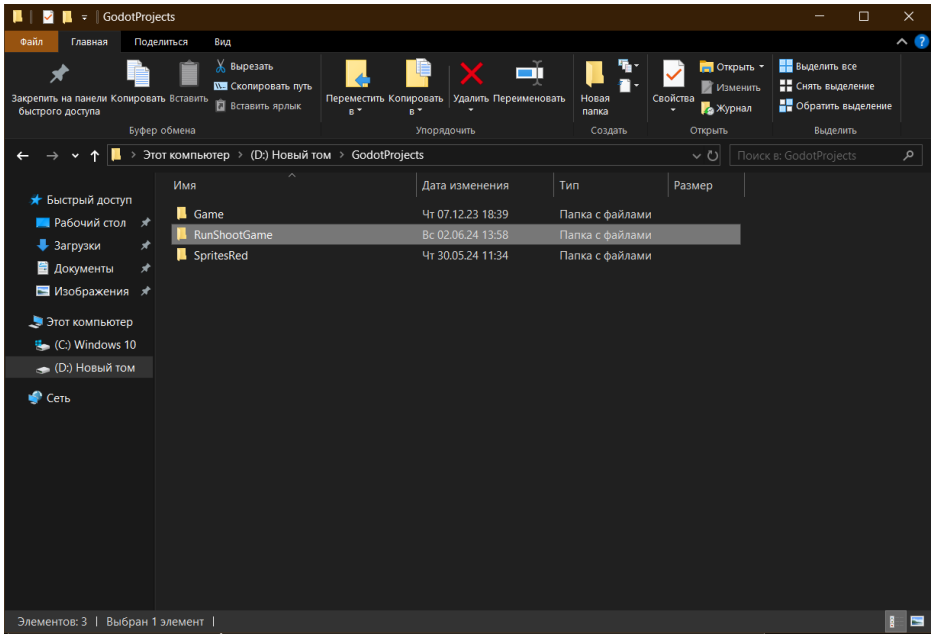

Рисунок 4.1 – Проєктний каталог

Каталоги та підкаталоги:

1) **scenes/**: каталог для зберігання сцен проєкту. Кожна сцена зазвичай представляє собою окремий рівень, екран або об’єкт у грі (рис. 4.2).

- Main_menu.tscn: сцена головного меню гри;
- World.tscn: сцена першого рівня гри;
- Player.tscn: сцена гравця;
- In_game_menu.tscn: сцена внутрішньоігрового меню.

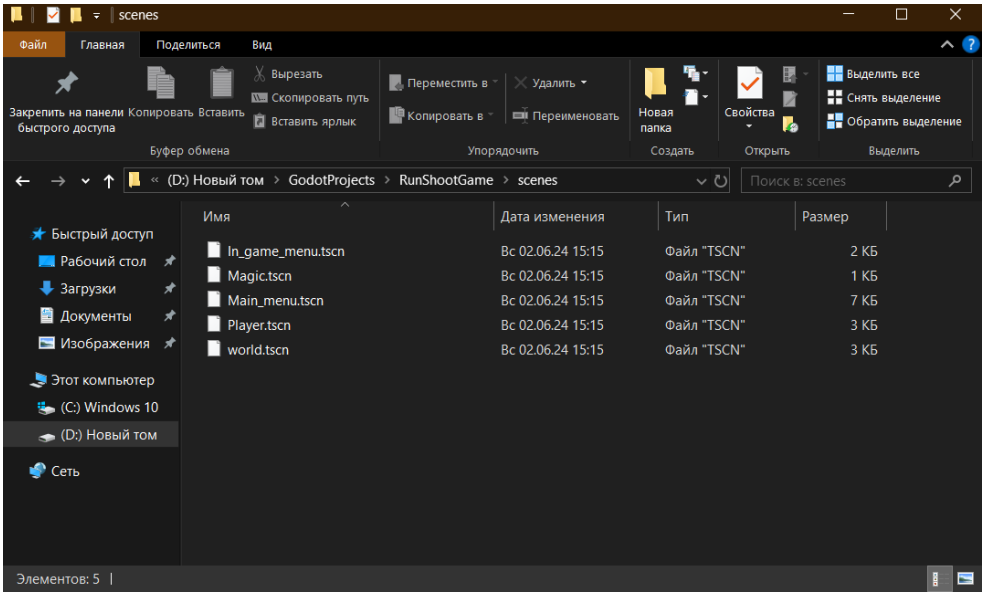

Рисунок 4.2 – Каталог scenes

2) **scripts/**: каталог для зберігання скриптів. Скрипти зазвичай написані на GDScript, але можуть також використовувати C# або інші підтримувані мови (рис. 4.3).

- In_game_menu.gd: внутришньоігрове меню (див. Додаток А);
- Leaves.gd: анімації об’єкту «Листья»;
- Magic.gd: анімації атаки;
- Main_menu.gd: головне меню гри (див. Додаток А);
- Player.gd: гравець (див. Додаток А);
- Soul.gd: анімації об'єкту «Душа» (див. Додаток А).

Каталог scripts/ відіграє важливу роль у структурі проєкту, забезпечуючи організоване зберігання всіх скриптів, що визначають поведінку та функціонал різних компонентів гри. Кожен файл скрипту відповідає за конкретний аспект гри, що робить код більш модульним та легким для підтримки. Наприклад, скрипт In_game_menu.gd відповідає за функціонал внутрішньоігрового меню, що забезпечує зручність управління під час гри, тоді як Player.gd визначає поведінку гравця, включаючи рухи, взаємодію з оточенням та атаки. Ця організація дозволяє

легко знаходити та модифікувати потрібні скрипти, що сприяє ефективному розвитку та підтримці проєкту.

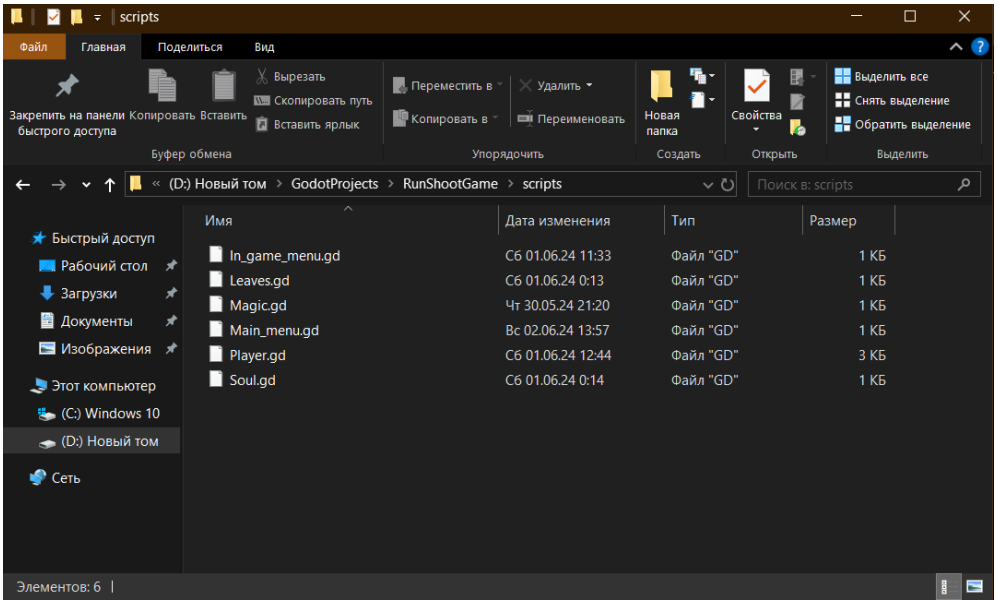

Рисунок 4.3 – Каталог scripts

3) **assets/**: каталог для зберігання активів, таких як зображення, звуки, шрифти тощо (рис. 4.4).

- Images/: підкаталог для зберігання зображень;
- Sounds/: підкаталог для зберігання звуків та музики.

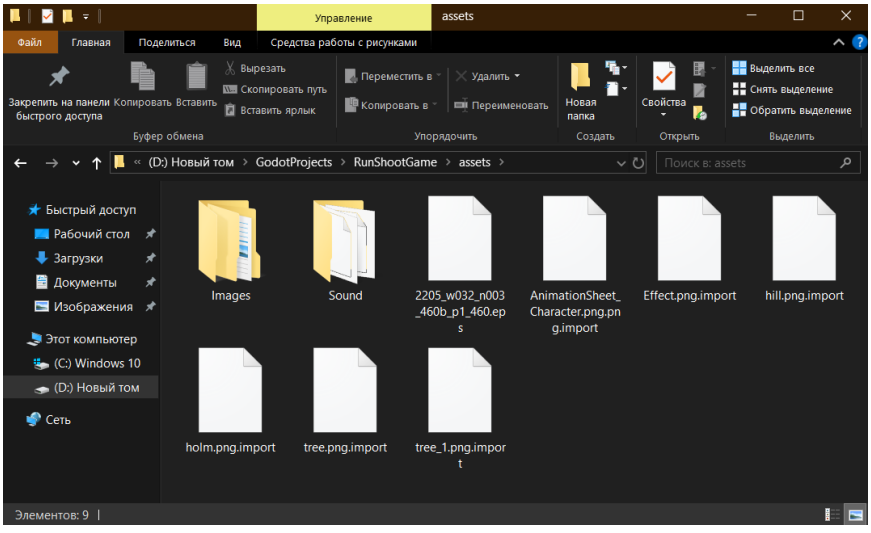

Рисунок 4.4 – Каталог assets

4) **addons/**: каталог для зберігання додатків та плагінів, що додають функціональність до Godot (рис. 4.5).

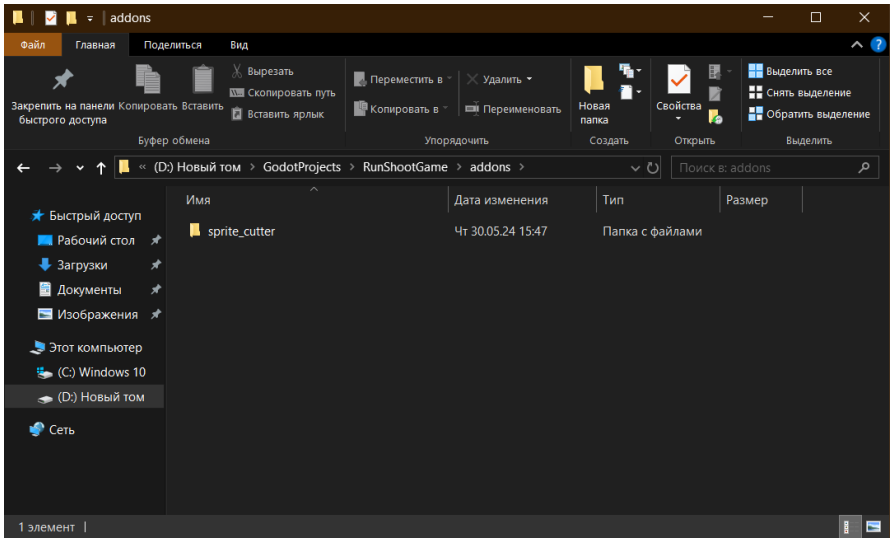

Рисунок 4.5 – Каталог addons

5) animations:/: каталог для зберігання анімацій (рис. 4.6).

- ball_animations.tres: анімація атакуючих спрайтів;
- leaves_animations.tres: анімація листя;
- player_animations.tres: анімація гравця;
- soul_animations.tres: анімація душі.

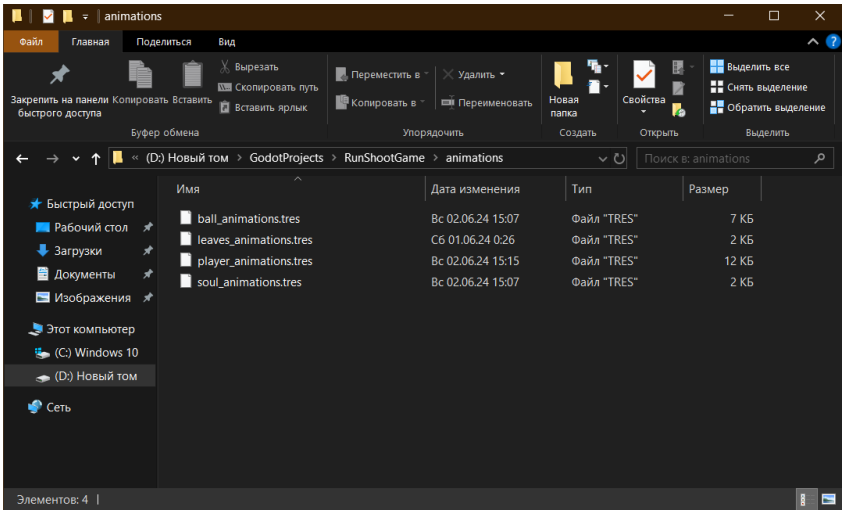

Рисунок 4.6 – Каталог animations

Загальний вигляд структури у проєкті (рис. 4.7):

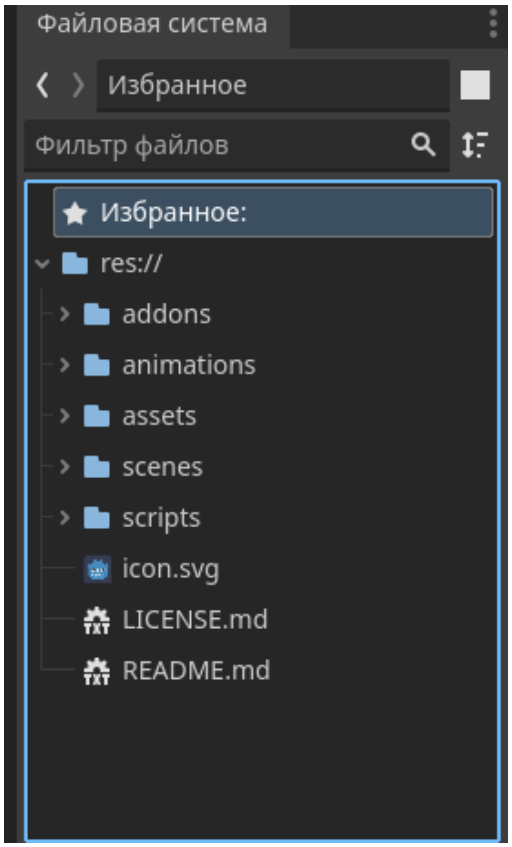

Рисунок 4.7 – Структура проєкту

Ця структура дозволяє легко орієнтуватися у проєкті та знаходити необхідні файли. Звісно, структура може бути налаштована відповідно до потреб проєкту, але загальні принципи організації залишаються схожими.

Включення ресурсів у проєкт:

Успішна розробка гри вимагає ефективного використання різних типів ресурсів, зокрема спрайтів, музики та анімацій. Ці ресурси відіграють ключову роль у створенні захоплюючого ігрового досвіду, надаючи візуальну привабливість, звукове супроводження та динаміку гри. У Godot 4 існують потужні інструменти для роботи з цими ресурсами, які дозволяють розробникам легко інтегрувати та управляти ними.

Головне меню:

Головне меню гри - це перша зустріч гравця з вашим проєктом. Воно відображається при запуску гри та включає в себе не лише набір функцій для

навігації, але й створює настрій та перші враження. Розглянемо, як зробити головне меню захопливим і привабливим для гравців (рис. 4.8).

Рисунок 4.8 – Головне меню

Головне меню гри повинне бути привабливим та простим у використанні. Ось деякі ключові елементи функціоналу та зовнішнього вигляду, які були включені:

1) фонове зображення [10]**:** затемнене або прозоре фонове зображення створить атмосферу та додасть глибини до інтерфейсу;

2) кнопки навігації [21]**:** чітко виділені кнопки з текстом або іконками для навігації, наприклад, «Play», «Load», «Continue» «Options», «Exit»;

3) анімації**:** динамічні анімації у меню гри також впливають на атмосферу та на глибину інтерфейсу;

4) музика: приємна мелодія, що грає в фоні (рис. 4.10), створює атмосферу та занурює гравця у світ гри ще до початку гри [13].

У даному випадку було додано анімований об’єкт «Soul» (рис. 4.9). Цей об’єкт відіграє важливу роль у створенні атмосфери гри, додаючи динаміки та візуальної привабливості першому рівню. Анімація «Soul» не тільки покращує загальний вигляд сцени, але й може бути інтерактивним елементом, з яким гравець взаємодіє, що робить гру більш захоплюючою.

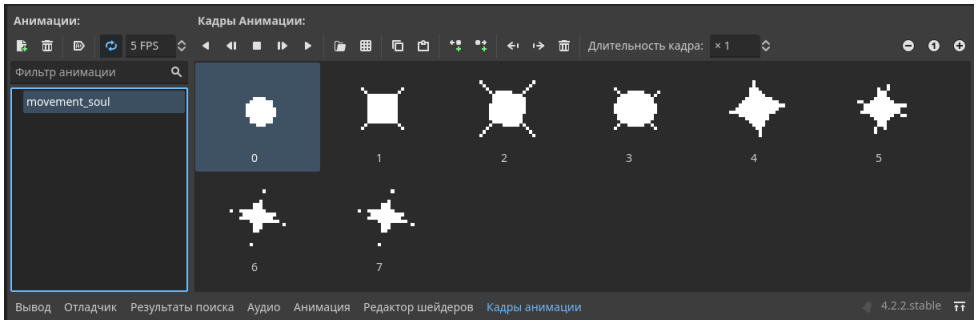

Рисунок 4.9 – Кадри анімації

Анімація об'єктів у головному меню надає йому динамічності та живої енергії, створюючи привабливий та захоплюючий вигляд.

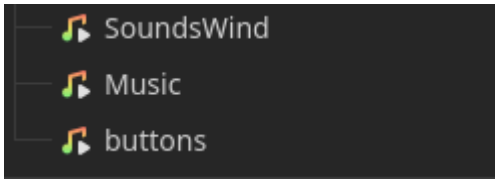

Рисунок 4.10 – Обрана музика та звукові ефекти

Ця музика була обрана з метою створення особливої атмосфери та враження гравців, допомагаючи підкреслити настрій та ідеї проєкту.

Перший рівень гри:

Перший рівень гри - це важливий момент у відкритті ігрового досвіду. Він позначає початок пригоди гравця у створеному проєкті, де він першим чином взаємодіє з ігровим світом, занурюється у гру завдяки музики (рис. 4.12) та анімаціям. Розглянемо, як створити захопливий і привабливий перший рівень, щоб зробити гру цікавою та захоплюючою для гравців (рис. 4.11). Дизайн першого рівня має бути ретельно продуманим, щоб залучити гравця з самого початку та дати йому зрозуміти основні механіки гри. Використання візуальних ефектів, анімацій та звукового супроводу сприяє глибшому зануренню в ігровий процес. Крім того, перший рівень має викликати інтерес до подальшого дослідження світу гри та стимулювати гравця до подальших дій. Таким чином,

створення якісного першого рівня є ключовим елементом у забезпеченні позитивного досвіду гравця та його утримання в грі.

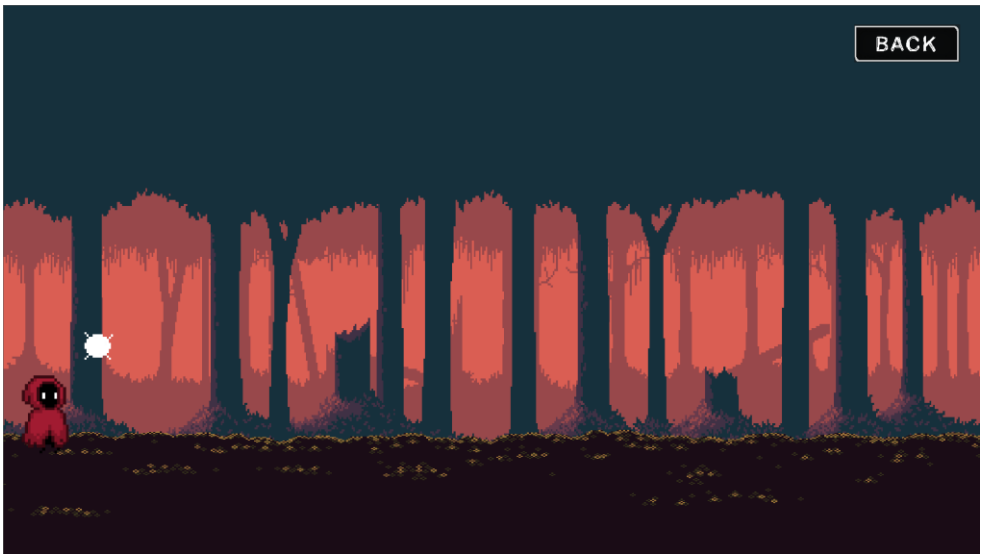

Рисунок 4.11 – Перший рівень гри

Перший рівень гри визначає початок подорожі гравця у ігровому світі. Цей рівень відображає першу зустріч з викликами, перешкодами та можливостями, які чекають на гравця. Він важливий для встановлення настрою та ритму гри, створення відчуття інтриги та поглиблення зв'язку між гравцем і ігровим світом. Перший рівень має зацікавити, залучити та залишити гравця з бажанням досліджувати подальші можливості гри.

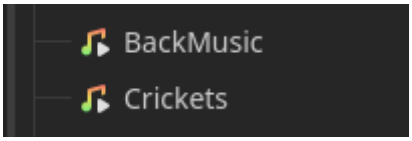

Рисунок 4.12 – Обрана музика та звукові ефекти

Обрана спокійна музика та звукові ефекти цвіркунів [12] допомагають створити особливу атмосферу лісу, що оточує перший рівень гри. Мелодія плавно відтворюється в фоновому режимі, переносячи гравця у світ спокою та загадковості, який спонукає на його дослідження. Звукові ефекти цвіркунів додають реалізму та глибини, підкреслюючи природну красу лісу та роблячи гру більш імерсивною.

Організація сцен та взаємодії між ними

Організація сцен у проєкті відіграє важливу роль у створенні структури гри та управлінні переходами між різними частинами ігрового досвіду. Наразі перехід між рівнем гри було створено у головному меню та у окремій сцені внутрішньоігрового меню. Давайте розглянемо кнопки головного та внутрішньоігрового меню, за що вони відповідають (рис. 4.13):

Рисунок 4.13 – Кнопки головного меню

Було додано такі кнопки навігації**:**

5) Play: відповідає за перехід на перший рівень гри (рис. 4.14);
6) Load: відповідає за перехід в меню гри де можна обрати рівень на якому хоче опинитися гравець;
7) Continue: завантажує гравця на останнє збереження;
8) Options: відчиняє вікно налаштувань;
9) Exit: виконує вихід з гри (рис. 4.15).

Перехід на рівень гри виконується простим скриптом кнопки «Play»:

```
func _on_play_pressed():
    buttons.play()
    await buttons.finished
    get_tree().change_scene_to_file("res://Scenes/world.tscn")
```

Рисунок 4.14 – Скрипт переходу на перший рівень гри

Коли гравець натискає на кнопку «Play» виконується скрипт що завантажує обрану сцену гри.

```
func _on_exit_pressed():
    buttons.play()
    await buttons.finished
    get_tree().quit()
```

Рисунок 4.15 – Скрипт виходу з гри

Коли гравець натискає на кнопку «Exit» виконується скрипт що завершує сеанс ігрового процесу. Також було додано внутрішньоігрове меню в якому знаходиться кнопка «Back» (рис. 4.16), що повертає гравця у головне меню (рис. 4.17).

Рисунок 4.16 – Кнопка внутрішньоігрового меню

Додавання активної кнопки "Back" для переходу назад у меню гри є важливим елементом, який полегшує навігацію гравців та забезпечує зручний ігровий досвід.

```
func _on_button_pressed():
    buttons.play()
    await buttons.finished
    get_tree().change_scene_to_file("res://Scenes/Main_menu.tscn")
```

Рисунок 4.17 – Скрипт повернення у головне меню

Коли гравець натискає на кнопку «Back» виконується скрипт що повертає гравця назад до головного меню.

4.2 Реалізація анімації персонажів та об’єктів

Реалізація анімації для персонажів та об'єктів у грі є важливою частиною процесу розробки, яка додає живості та динаміки до ігрового світу. Нижче розглянемо процес імплементації анімації, використання спрайтів або анімаційних об'єктів, а також методи управління анімацією:

Процес імплементації анімації

1) створення анімаційних спрайтів або анімаційних об'єктів**:** використання програм для редагування графіки або інструментів для створення анімацій, щоб створити анімаційні кадри для персонажів та об'єктів у грі;

2) імпорт анімацій: імпорт створених анімацій до ігрового движка. У багатьох движках, таких як Unity чи Unreal Engine, анімації можуть бути імпортовані у вигляді файлів або спрайтових атласів;

3) налаштування анімаційних параметрів**:** встановлення параметрів анімації, таких як швидкість відтворення, повторення, зациклення тощо.

Створення анімацій через спрайти - це поширений метод для надання живості персонажам та об'єктам у грі, саме цей метод було використано протягом створення гри.

Давайте проаналізуємо анімації головного героя. Ми розглянемо рухові витвори, які надають нашому головному персонажу особливого шарму та виразності у тому числі анімації: attack (рис. 4.18), death (рис. 4.19), idle (рис.

4.20), jump (рис. 4.21), run (рис. 4.22), sit (рис. 4.23), unconscious (рис. 4.24), walk (рис. 4.25).

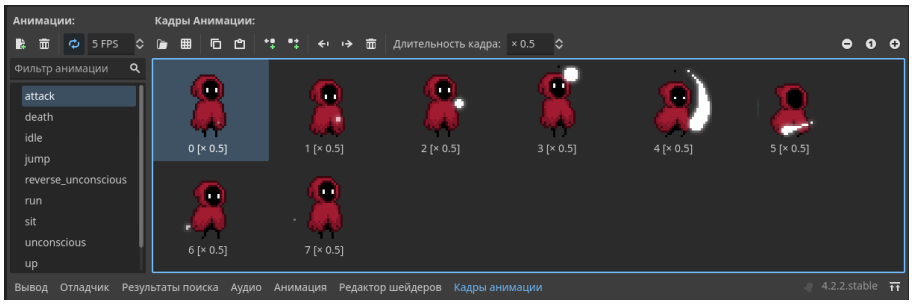

Рисунок 4.18 – Анімація attack

На цьому зображенні ми бачимо анімацію атаки персонаж. Кожен кадр цієї анімації втілює в собі майстерну гру рухом, відтворюючи кожен етап атаки з вражаючою деталізацією.

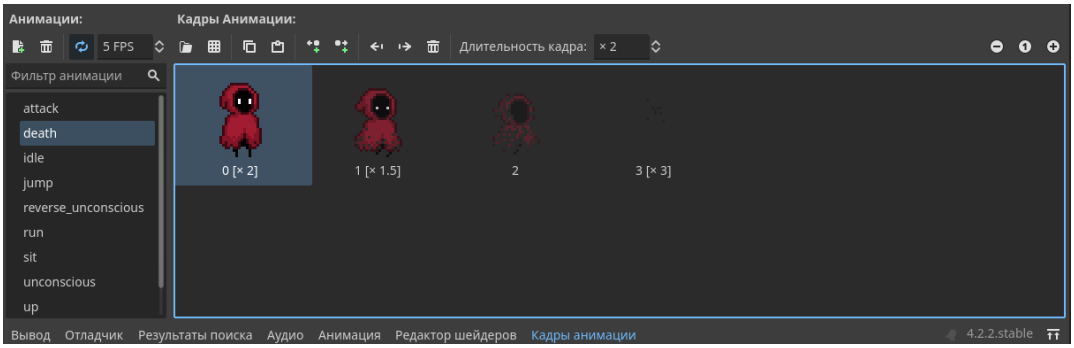

Рисунок 4.19 – Анімація death

На цьому зображенні зображена анімація смерті персонажа. Кожен кадр показує різні етапи смерті, від передсмертних поступових рухів до остаточного падіння. Через цю анімацію передається вразливість та моменти напруги перед останнім витягом персонажа. Реалістичність і деталізація рухів підкреслюють серйозність ситуації та можуть викликати емоційний відгук у гравця. Включення подібних анімацій покращує загальну якість гри, роблячи її більш захопливою та емоційно насиченою. Крім того, така анімація допомагає створити глибший наратив, де кожна смерть персонажа має вагу та значення. Це також стимулює гравця обережніше підходити до гри, уникаючи повторення помилок.

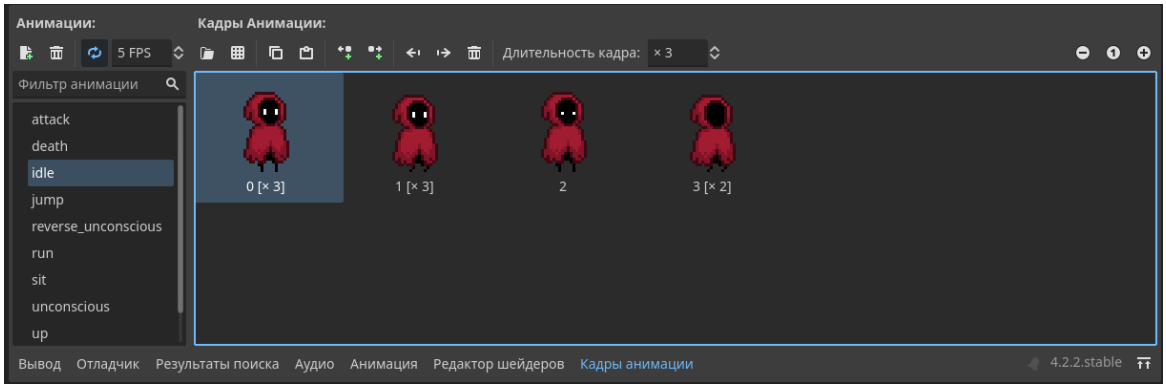

Рисунок 4.20 – Анімація idle

На зображенні представлена анімація "idle" персонажа, яка відтворюється у моменти, коли гравець не надає жодних команд управління персонажем. Кожен кадр цієї анімації демонструє статичні пози, де персонаж може знаходитись у різних станах спокою або очікування. Ця анімація служить для того, щоб зберігати вигляд персонажа в моменти його бездіяльності, додавши реалістичності та живості до геймплею.

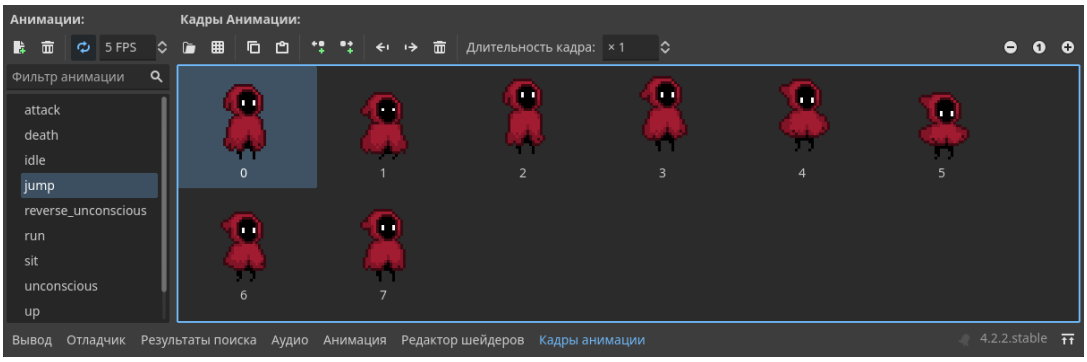

Рисунок 4.21 – Анімація jump

На цьому зображенні зображена анімація стрибка персонажа. Кожен кадр цієї анімації передає різні етапи стрибка, від початкового згину колін до моменту максимального піднімаються у повітря. Ця анімація демонструє динаміку та енергію, які супроводжують стрибок персонажа у грі. Крім того, кожен кадр відображає позу персонажа, яка підкреслює його активність та напруженість у момент стрибка. Ця анімація додає реалістичності та іммерсії до ігрового процесу, роблячи геймплей більш привабливим для гравця

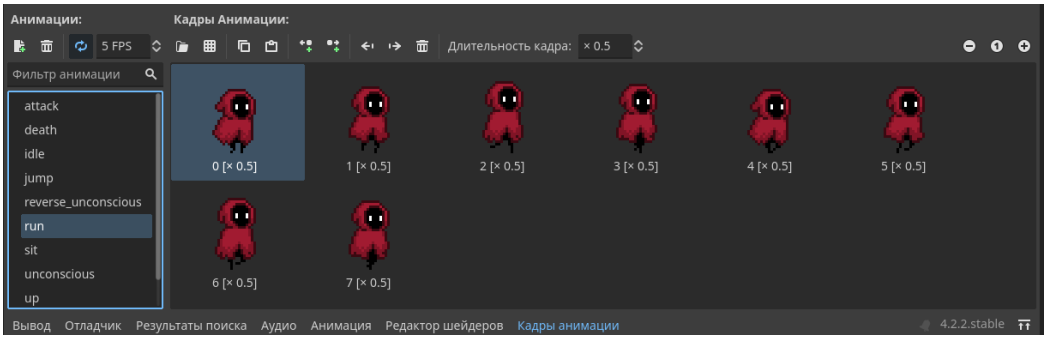

Рисунок 4.22 – Анімація run

На цьому зображенні зображена анімація бігу персонажа. Кожен кадр анімації показує різні фази руху персонажа під час бігу: від розгону до моменту максимальної швидкості, а потім до зменшення швидкості або зупинки. Крім того, у кожному кадрі відображено позу тіла, яка підкреслює динаміку руху та енергію персонажа. Ця анімація додає живості ігровому процесу, роблячи біг більш реалістичним та захопливим для гравця.

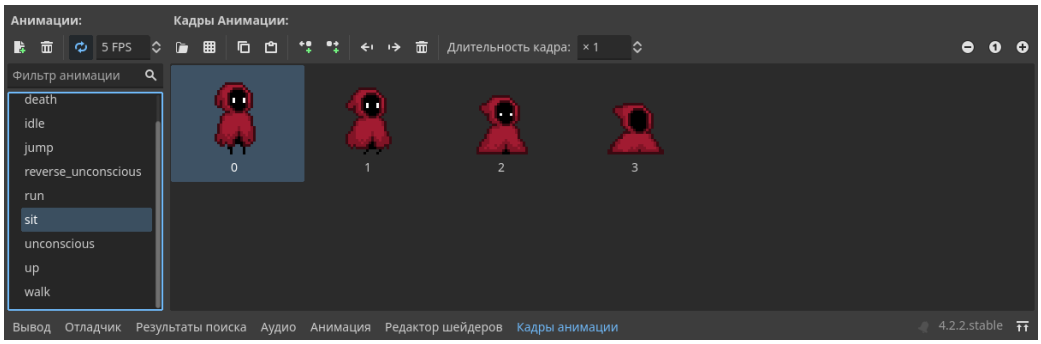

Рисунок 4.23 – Анімація sit

На зображенні представлена анімація "sit" персонажа. Кожен кадр анімації передає різні пози персонажа в сидячому положенні. Ця анімація включає рухи ніг та тіла персонажа, щоб додати реалізму і виразності. Анімація "sit" може бути використана в різноманітних ситуаціях у грі, таких як відпочинок, розмова з іншими персонажами або виконання певних завдань. Анімація також сприяє загальному зануренню гравця в ігровий світ, забезпечуючи більш глибоке взаємодію з персонажем. Такі деталі підвищують якість гри, роблячи її більш привабливою та захопливою для користувачів.

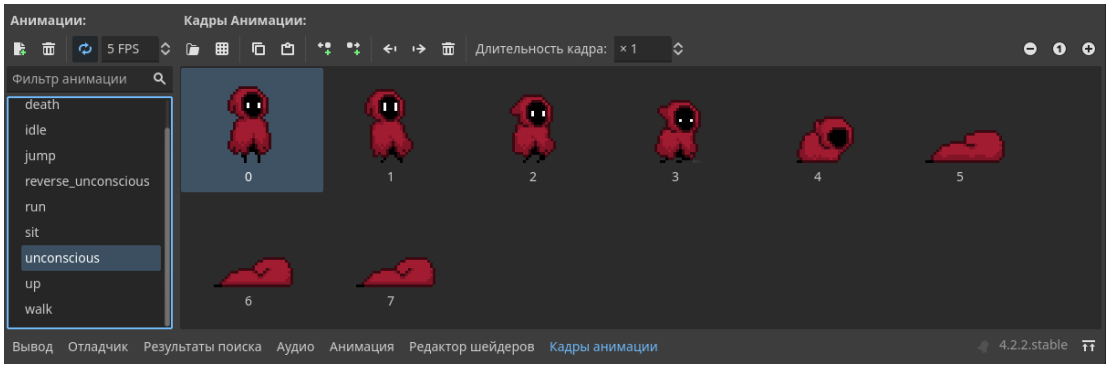

Рисунок 4.24 – Анімація unconscious

На зображенні відображена анімація "unconscious" персонажа. Кожен кадр цієї анімації може показувати різні стани безсвідомості, такі як перебування у непритомному стані після потрапляння у важку ситуацію або отримання великих травм. Анімація може включати різні пози тіла, які відображають ступінь травматизму або вираз безсвідомості. Ця анімація може бути використана для надання глибини та реалізму сценам бою чи небезпеки в грі, підкреслюючи драматизм та напругу ситуації.

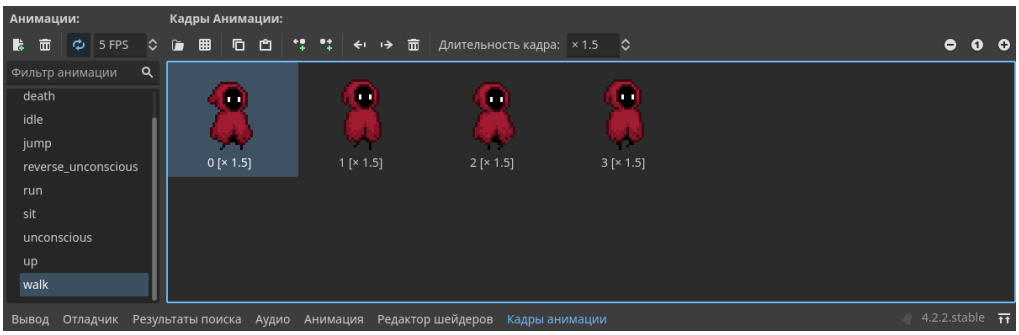

Рисунок 4.25 – Анімація walk

На цьому зображенні представлена анімація "walk" (ходьба) персонажа. Кожен кадр цієї анімації передає різні етапи руху персонажа під час ходьби: звисання однієї ноги, крок другою ногою, рух рук тощо. Ця анімація відображає натуральність та плавність руху персонажа, допомагаючи створити реалістичну ілюзію прогулянки. Кожен кадр також підкреслює динаміку та грацію руху, що робить геймплей більш природним і захопливим для гравця.

Це були основні анімації персонажа у грі, давайте розглянемо та проаналізуємо анімації магічних здібностей персонажа:

На даний момент розробки гри було додано декілька анімацій магічних здібностей [11], а саме магічний шар з різними елементами та впливом на ворогів: fireball (рис. 4.26), lightningball (рис. 4.27), waterball (рис. 4.28).

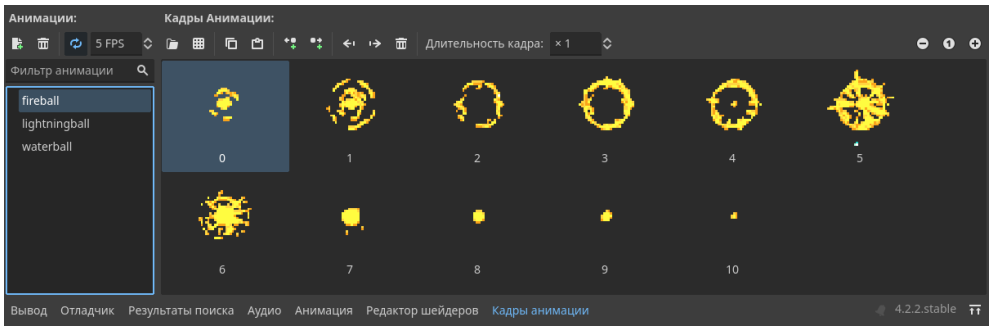

Рисунок 4.26 – Анімація fireball

На цьому зображенні представлена анімація здібності "fireball" персонажа. Кожен кадр цієї анімації передає різні етапи створення та випуску вогняної кулі: збір енергії, формування кулі та відправлення її у ворожого об'єкта. Кожен кадр також може включати спеціальні візуальні ефекти, такі як полум'я або іскри, що робить анімацію більш захоплюючою та реалістичною для гравця

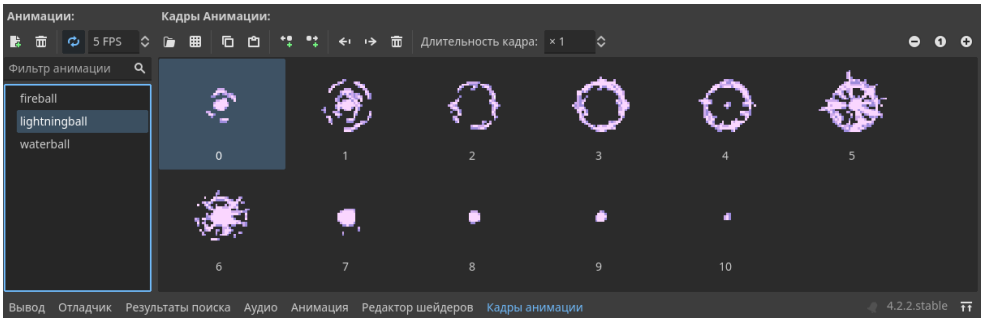

Рисунок 4.27 – Анімація lightningball

На цьому зображенні представлена анімація здібності "lightningball" персонажа. Кожен кадр цієї анімації передає різні етапи створення та випуску кулі блискавки: збір енергії, формування кулі та випуск її у ворожого об'єкта. Кожен

кадр може також включати візуальні ефекти, такі як блискавка або зарядження енергії, що робить анімацію більш захоплюючою та реалістичною для гравця.

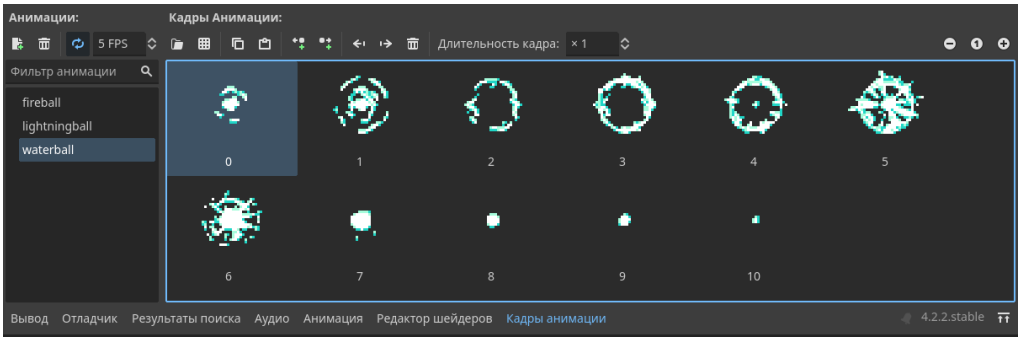

Рисунок 4.28 – Анімація waterball

На зображенні представлена анімація здібності "waterball" (водяна куля) персонажа. Кожен кадр цієї анімації демонструє процес створення та випуску кулі води: збір енергії, формування кулі та її направлення у ворожого об'єкта. Ця анімація передає грацію та елегантність водного елементу, а також може підкреслювати його силу, відображаючи бурхливість водяних потоків або шум при випуску кулі. Кожен кадр може також включати візуальні ефекти, такі як бризки води чи сяйво, що додає реалізму анімації.

Давайте також розглянемо анімацію такого об’єкту як «Soul» (рис. 4.29)

Рисунок 4.29 – Анімація movement_soul

На зображенні представлена анімація руху душі. Кожен кадр цієї анімації може відображати різні етапи руху душі, такі як підйом, спад або плавний перехід з однієї точки в іншу. Ця анімація може мати плавні та плаваючі рухи, що

надають душі ефірності та легкості. Кожен кадр включає візуальні ефекти, такі як мерехтіння чи світіння, що створює враження містичності та магічності душі.

Кожна анімація в грі має свої власні настройки швидкості та напрямку, що надає їй унікальність і виразність. Ці параметри дозволяють кожному русі, кожному ефекту відтворюватися з відповідною динамікою та реалістичністю. Вони допомагають створити атмосферу та підкреслити сюжетні нюанси гри, роблячи ігровий досвід більш насиченим та захоплюючим для гравця.

4.3 Взаємодія гравця з ворогами та об’єктами гри

Система колізій та взаємодії об'єктів гри відіграє важливу роль у створенні ігрового середовища. Колізії визначають, як об'єкти взаємодіють один з одним, включаючи зіткнення гравця з ворогами або предметами. Обробка цих колізій визначає події, які відбуваються у грі, такі як завдання шкоди гравцю або активація певних дій.

Механіки гри, пов'язані з взаємодією гравця з ворогами, включають в себе систему бою з різними видами атак та захисту, а також ворожі поведінкові шаблони. Ці механіки роблять гру цікавішою та викликають більше емоційних відгуків у гравця.

Система колізій є ключовим елементом в організації взаємодії між різними об'єктами у грі. Вона визначає, як об'єкти реагують на зіткнення та як це впливає на ігровий процес. Розглянемо, як система колізій реалізована для гравця, ворогів та атак персонажа.

Колізії у гравця

У гравця система колізій відповідає за кілька важливих аспектів:

1) рух по землі**:** колізії визначають, як гравець взаємодіє з поверхнями, забезпечуючи правильну фізику руху та запобігаючи проходженню крізь об'єкти;

2) отримання урону від ворогів**:** коли гравець зіштовхується з ворогом, колізії визначають, чи завдається шкода гравцю та скільки саме шкоди;

Колізії у ворогів

Для ворогів система колізій виконує подібні функції:

1) рух по землі**:** колізії забезпечують правильну фізику руху ворогів, дозволяючи їм пересуватися по поверхнях гри;

2) отримання урону від гравця**:** коли атака гравця потрапляє у ворога, система колізій визначає, чи завдано шкоди ворогу та який обсяг цієї шкоди.

Колізії у атак персонажа

Атаки персонажа також мають свої колізії, які відіграють вирішальну роль у бою:

1) попадання по ворогу**:** колізії визначають, чи влучає атака в ворога, що залежить від точного зіткнення між об'єктом атаки та ворогом;

2) нанесення урону**:** якщо атака потрапляє в ворога, колізії визначають, чи завдано шкоди та скільки саме.

Ці компоненти системи колізій забезпечують реалістичність та інтерактивність ігрового процесу, дозволяючи гравцям і ворогам взаємодіяти один з одним у правдоподібний спосіб.

Щодо штучного інтелекту ворогів, він забезпечує реалістичну поведінку та стратегії атаки. Методи руху включають у себе прості алгоритми слідування за гравцем та складніші алгоритми, які враховують обставини навколишнього середовища. Стратегії атаки можуть варіюватися в залежності від типу ворога та його характеристик.

4.4 Тестування та інструкції користувача комп’ютерної гри

Тестування комп’ютерної гри є невід'ємною частиною процесу розробки, оскільки воно забезпечує високу якість кінцевого продукту і задоволення користувачів. Тестування допомагає виявити та виправити помилки, баги, які можуть вплинути на ігровий досвід. Відсутність належного тестування може призвести до численних проблем, таких як збої гри, втрата прогресу користувача

або навіть безпекові ризики. Це може серйозно пошкодити репутацію розробника та знизити комерційний успіх гри.

Більше того, тестування дозволяє перевірити, чи відповідає гра початковим технічним вимогам та очікуванням користувачів. Це важливо не лише для забезпечення функціональності гри, але і для її відповідності ринковим стандартам та конкурентним продуктам. Тестування також допомагає оцінити продуктивність гри на різних апаратних платформах та в різних умовах експлуатації, що є критичним для мультиплатформних ігор.

Також важливо зазначити, що тестування є важливим етапом для перевірки зручності використання інтерфейсу та загального користувацького досвіду. Це включає оцінку інтуїтивності управління, логічності навігації та загальної привабливості гри. Всі ці фактори є ключовими для залучення та утримання користувачів.

Тестування комп'ютерної гри проводиться в кілька етапів, кожен з яких має свою мету та завдання. Основні етапи тестування включають:

1) альфа-тестування: Перший етап внутрішнього тестування, який проводиться командою розробників. На цьому етапі основна увага приділяється виявленню основних помилок і багів, які можуть заважати базовій функціональності гри;

2) бета-тестування: Залучення обмеженого кола зовнішніх користувачів для тестування гри в умовах, наближених до реальних. Це дозволяє отримати відгуки від різних типів гравців і виявити додаткові баги та недоліки, які могли бути пропущені під час альфа-тестування;

3) релізне тестування: Завершальний етап тестування перед випуском гри. Проводиться остаточна перевірка всіх виправлень і змін, внесених на попередніх етапах, для забезпечення стабільної та безпомилкової роботи гри при її релізі.

Загальний процес тестування комп’ютерної гри може включати наступні кроки:

Планування тестування:

1) визначення цілей тестування;

2) розробка тестового плану, що включає сценарії, тестові випадки та критерії успіху;

3) підготовка тестового середовища та інструментів.

Проведення тестування:

1) модульне тестування: Перевірка окремих компонентів гри на наявність помилок;

2) інтеграційне тестування: Перевірка взаємодії між різними модулями гри;

3) системне тестування: Комплексна перевірка всієї гри на відповідність технічним та функціональним вимогам;

4) регресійне тестування: Перевірка того, що виправлення помилок не призвели до виникнення нових багів в інших частинах гри.

Оцінка продуктивності:

1) вимірювання швидкості завантаження гри, частоти кадрів (FPS), навантаження на процесор і графічну карту;

2) перевірка гри на різних апаратних конфігураціях та операційних системах.

Перевірка зручності використання:

1) оцінка інтуїтивності та логічності користувацького інтерфейсу;

2) збір відгуків від тестувальників щодо зручності управління та загального ігрового досвіду.

Збір та аналіз результатів тестування:

1) складання звітів про знайдені помилки та їх пріоритезація;

2) аналіз продуктивності та стабільності гри;

3) оцінка користувацького досвіду на основі зібраних відгуків.

Виправлення помилок та повторне тестування:

1) внесення змін та виправлень на основі отриманих результатів;

2) повторне тестування для перевірки виправлень та виявлення нових проблем.

Підготовка до релізу:

1) остання перевірка гри перед випуском;

2) підготовка документації для користувачів та інструкцій щодо використання гри.

Виконання цих кроків забезпечує високоякісний кінцевий продукт, який відповідає очікуванням користувачів та стандартам індустрії. Це сприяє підвищенню задоволеності користувачів та успішності гри на ринку.

Перед тим як зануритися в світ комп'ютерної гри, важливо ще зрозуміти основні аспекти, які дозволять гравцю максимально насолодитися ігровим процесом. Інструкції користувача мають на меті допомогти новим гравцям розпочати гру, ознайомити їх з базовими та розширеними функціями, а також надати чіткі вказівки щодо налаштувань гри для оптимальної продуктивності та комфорту. Правильне ознайомлення з інтерфейсом, елементами керування та ігровою механікою дозволить уникнути непорозумінь і підвищить ефективність гри.

Інструкції користувача є невід'ємною частиною будь-якої гри, оскільки вони забезпечують плавний перехід від реального світу до віртуального. Вони допомагають гравцям зосередитися на сюжеті гри та її завданнях, не відволікаючись на технічні нюанси. Крім того, детальні інструкції допомагають новачкам швидше освоїти гру, що є важливим аспектом, особливо для складних або багаторівневих ігор.

Розглянемо ключові розділи інструкцій користувача, які допоможуть гравцям розпочати гру, зрозуміти її основні механіки, а також налаштувати гру відповідно до своїх потреб та уподобань:

Початок роботи:

Головне меню

Після запуску гри гравець потрапляє до головного меню, яке є стартовою точкою для початку гри та налаштувань. Головне меню включає наступні опції:

1) нова гра: почати нову гру з самого початку;
2) завантажити: завантажити гру з збереженого моменту на вибір;
3) продовжити: продовжити гру з останнього збереженого моменту;
4) налаштування: перейти до налаштувань гри, де можна змінити параметри графіки, звуку та управління;
5) вихід: вийти з гри.

Елементи керування

Основні елементи керування включають:

1) W, A, S, D: переміщення персонажа;
2) ліва кнопка миші: основна дія (атака, вибір);
3) права кнопка миші: вторинна дія (прицілювання, контекстні дії);
4) E: взаємодія з об'єктами;
5) пробіл: стрибок;
6) shift: прискорення.

Режим гри

Гра має тільки один режим гри – одиночна гра. В цьому режимі гравець проходить сюжет, виконуючи завдання та місії, досліджуючи світ гри та взаємодіє з персонажами.

Ігровий процес:

Мета

Основна мета гри залежить від сюжетної лінії. Це може бути досягнення певних цілей, перемога над ворогами, вирішення головоломок або виживання в складних умовах. Гравцеві необхідно виконувати завдання та місії, які поступово відкривають нові рівні та можливості гри.

Управління гравцем

Гравець керує персонажем, використовуючи різні елементи управління для переміщення, атаки, взаємодії з об'єктами та виконання спеціальних дій.

Ігрова механіка

Ігрова механіка включає всі дії та взаємодії, які гравець може виконувати в грі:

1) бойова система: включає різні типи атак, захисту та ухилянь. Також включає використання магії або спеціальних навичок;

2) дослідження: гравець може досліджувати навколишній світ, взаємодіяти з об'єктами та персонажами, шукати схованки та секрети;

3) головоломки: у грі можуть бути різні головоломки, які гравець повинен вирішити для просування сюжету.

Налаштування гри:

Графіка

Налаштування графіки дозволяють оптимізувати візуальну якість гри під апаратне забезпечення гравця:

1) роздільна здатність: вибір роздільної здатності екрана;

2) рівень деталізації: налаштування якості текстур, тіней, ефектів освітлення та інших графічних елементів;

3) режим вікна: вибір між повноекранним режимом, віконним режимом або віконним режимом без рамок;

4) антиаліайзинг: налаштування згладжування країв об'єктів.

Звук

Налаштування звуку забезпечують оптимальне аудіосередовище для гри:

1) гучність музики: регулювання рівня гучності музичного супроводу;

2) гучність звукових ефектів: регулювання рівня гучності звукових ефектів;

3) гучність діалогів: регулювання рівня гучності діалогів персонажів;

4) вибір аудіо виходу: налаштування пристроїв для відтворення звуку (динаміки, навушники).

Інструкції користувача надає гравцям необхідну інформацію для початку гри, розуміння ігрового процесу та налаштування параметрів відповідно до їхніх потреб. Детально описане головне меню, елементи керування та режими гри

дозволяють гравцям швидко орієнтуватися в інтерфейсі та розпочати гру без зайвих труднощів.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи було детально розглянуто організацію робочого середовища, структуру проєкту та реалізацію ключових аспектів геймплею в Godot 4. Правильна організація проєкту, включаючи розподіл ресурсів, таких як спрайти, музика, анімації, та їх інтеграція, є критичною для ефективного розроблення гри.

Реалізація анімації персонажів та об'єктів за допомогою спрайтів додає візуальної привабливості та динаміки. Кожна анімація, така як атака, смерть, покой, стрибок, біг, сидіння, безсвідомість, ходьба та здібності персонажа, була налаштована з урахуванням швидкості та напрямку, що забезпечує реалістичність ігрового процесу.

Система колізій відіграє ключову роль у взаємодії об'єктів гри. Колізії гравця відповідають за рух по землі та отримання урону від ворогів, колізії ворогів – за отримання шкоди та рух по землі, а колізії атак персонажа – за влучання у ворога та нанесення шкоди.

Штучний інтелект ворогів забезпечує реалістичну поведінку, що робить ворогів складними та цікавими суперниками. Методи руху та реакції на дії гравця додають глибини до геймплею.

Механіки гри, пов'язані з взаємодією гравця з ворогами, включають систему бою, ворожі поведінкові шаблони та стратегії виживання, що робить гру захоплюючою та викликає емоційні відгуки у гравця.

Усі розглянуті аспекти організації робочого середовища, структури проєкту, анімації та систем взаємодії формують цілісний ігровий досвід, роблячи гру захоплюючою, реалістичною та привабливою для гравців, що є основою для створення успішного ігрового проєкту.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра здійснено всебічний аналіз, розробку та тестування 2D гри на ігровому рушію Godot. Основні етапи роботи включали аналіз аналогічних ігрових продуктів, визначення вимог до розробки гри, моделювання алгоритмів та логіки гри, а також створення прототипу гри. Робота демонструє здатність використовувати сучасні технології та методики у геймдеві, забезпечуючи якісний ігровий досвід. Для досягнення поставленої мети виконано наступні завдання:

1) проаналізовано аналогічні ігрові продукти на платформі Godot та інших двигунах: проаналізовано існуючі ігрові проєкти, що дозволило ідентифікувати сильні та слабкі сторони конкурентів, врахувати сучасні тенденції та визначити унікальні елементи для власного проєкту;

2) визначено основні вимоги до розроблюваної гри на основі аналізу сучасних тенденцій у геймдеві: Визначено функціональні та нефункціональні вимоги, що включають необхідні ігрові механіки, графічні та звукові ресурси, а також очікування від продуктивності та користувацького інтерфейсу;

3) розроблено алгоритми та логіку гри з використанням побудови UML-діаграм та блок-схем: Виконано моделювання ігрових процесів та взаємодій між об’єктами гри за допомогою UML-діаграм. Це забезпечило чітке уявлення про структуру та функціонал гри, полегшило розробку і тестування;

4) розроблено прототип гри та проведено його тестування на відповідність вимогам: Розроблено прототип гри з реалізацією базових механік, анімацій та інтерфейсу. Проведено тестування прототипу для виявлення та усунення помилок, забезпечуючи відповідність початковим вимогам.

Практичне значення отриманих результатів полягає у створенні основи для подальшого розвитку та вдосконалення ігрового проєкту. Проєкт продемонстрував можливості ігрового рушія Godot для розробки 2D ігор, а також підкреслив важливість чіткого планування та моделювання на етапі проєктування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Arlow J., Neustadt A. UML 2 And the Unifield Process: Practical Object-Oriented Analysis And Design. Addison-Wesley Professional, 2005. 592 p.
2. PC Gamer: Action. URL: http://surl.li/udpaf (last accessed 26.04.2024)
3. PC Gamer: Strategy video game. URL: http://surl.li/udozy (last accessed 26.04.2024)
4. PC Gamer: Puzzle. URL: http://surl.li/udpay (last accessed 26.04.2024)
5. PC Gamer: Role playing video game. URL: http://surl.li/udpbd (last accessed 26.04.2024)
6. Steam: Dig or Die. URL: http://surl.li/tznrj (last accessed 28.05.2024)
7. Steam: Blood Harvest 3. URL: http://surl.li/tznte (last accessed 28.05.2024)
8. Steam: Dome Keeper. URL: http://surl.li/tznuc (last accessed 28.05.2024)
9. Itch.io: Hooded Protagonist. URL: http://surl.li/udpbt (last accessed 29.05.2024)
10. Itch.io: Demon Woods Parallax Background. URL: http://surl.li/udpbu (last accessed 29.05.2024)
11. Itch.io: Free Effect Pixel. 16×16 URL: http://surl.li/udpcl (last accessed 29.05.2024)
12. Mixkit: Free Game Sound Effects. URL: http://surl.li/udpco (last accessed 30.05.2024)
13. Pixabay: Free Game Sound Effects Library. URL: http://surl.li/udpdl (last accessed 30.05.2024)
14. Storyblocks: Pixel Royalty Free Music. URL: http://surl.li/udpdz (last accessed 30.05.2024)
15. Google Drive: Music. URL: http://surl.li/udpeh (last accessed 30.05.2024)
16. Itch.io: Night Borne Warrior. URL: http://surl.li/udpgn (last accessed 29.05.2024)
17. Itch.io: Enemies Animated. URL: http://surl.li/udpio (last accessed 30.05.2024)

18. Itch.io: Pixel Mud Hand Enemy Sprite. URL: http://surl.li/udpks (last accessed 30.05.2024)

19. Itch.io: Pixel Mud Hand Enemy Sprite. URL: http://surl.li/udpks (last accessed 30.05.2024)

20. Itch.io: Pixel Final Boss Enemy. URL: http://surl.li/udpld (last accessed 30.05.2024)

21. Freepick: Sprites Buttons. URL: http://surl.li/udpqi (last accessed 30.05.2024)

ДОДАТОК А

Лістинг програмних модулів

Скрипт Player

```
extends CharacterBody2D

@export var maxSpeed: int = 300
@export var acceleration: int = 2000
@export var sprintSpeedMultiplier: float = 2
@export var friction: int = 10000
@export var jumpForce: int = 1000
@export var gravity: int = 2000

var axis: Vector2 = Vector2.ZERO
@onready var animation = $AnimatedSprite2D
var is_attacking: bool = false

func _ready():
	pass

func _physics_process(delta: float) -> void:
	get_input_axis()

	if Input.is_action_just_pressed("left_click") and is_on_floor() and not is_attacking:
		attack()

	if is_attacking:
		# Skip other movements and animations if attacking
		apply_gravity(delta)
		move_and_slide()
		return
	if axis.x == 0:
		apply_friction(friction * delta)
		if is_on_floor():
			animation.play("idle")
	else:
		if Input.is_action_pressed("sprint"):
			apply_motion(acceleration * delta, maxSpeed * sprintSpeedMultiplier)
		else:
			apply_motion(acceleration * delta, maxSpeed)

		if is_on_floor():
			if abs(velocity.x) > maxSpeed * 0.5:
				if Input.is_action_pressed("sprint"):
					animation.play("run")
					if $Run/SoundRateRun.time_left == 0 and is_on_floor():
						$Run.play()
						$Run/SoundRateRun.start()
				else:
```

```
                    animation.play("walk")
                    if $Steps/SoundRateSteps.time_left == 0 and is_on_floor():
                        $Steps.play()
                        $Steps/SoundRateSteps.start()
            else:
                animation.play("walk")

        apply_gravity(delta)

        if Input.is_action_just_pressed("jump") and is_on_floor():
            jump()
            animation.play("jump")

        move_and_slide()

    func get_input_axis() -> void:
        axis.x = int(Input.is_action_pressed("move_right")) - int(Input.is_action_pressed("move_left"))
        if Input.is_action_pressed("move_left"):
            $AnimatedSprite2D.scale.x = -1
        elif Input.is_action_pressed("move_right"):
            $AnimatedSprite2D.scale.x = 1


    func apply_friction(friction_factor: float) -> void:
        if velocity.x != 0:
            if abs(velocity.x) > friction_factor:
                velocity.x -= sign(velocity.x) * friction_factor
            else:
                velocity.x = 0

    func apply_motion(acc_factor: float, max_speed: float) -> void:
        velocity.x += axis.x * acc_factor
        if abs(velocity.x) > max_speed:
            velocity.x = sign(velocity.x) * max_speed

    func apply_gravity(delta: float) -> void:
        if not is_on_floor():
            velocity.y += gravity * delta
        else:
            velocity.y = 0
    func jump() -> void:
        velocity.y = -jumpForce
    func attack() -> void:
        is_attacking = true
        animation.play("attack")
        await get_tree().create_timer(0.7).timeout
        is_attacking = false
```

Скрипт In_game_menu

```
extends CanvasLayer
@onready var buttons = $buttons
func _on_button_pressed():
    buttons.play()
    await buttons.finished
    get_tree().change_scene_to_file("res://Scenes/Main_menu.tscn")
```

Скрипт Main_menu

```
extends CanvasLayer

@onready var buttons = $buttons

func _on_play_pressed():
    buttons.play()
    await buttons.finished
    get_tree().change_scene_to_file("res://Scenes/world.tscn")

func _on_levels_pressed():
    buttons.play()
    await buttons.finished
    pass # Replace with function body.


func _on_options_pressed():
    buttons.play()
    await buttons.finished
    pass # Replace with function body.

func _on_saves_pressed():
    buttons.play()
    await buttons.finished
    pass # Replace with function body.

func _on_exit_pressed():
    buttons.play()
    await buttons.finished
    get_tree().quit()
```

Скрипт Soul

```
extends AnimatedSprite2D

func _ready():
    play("movement_soul")
```