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

ІГРОВИЙ ЗАСТОСУНОК У ЖАНРІ ACTION-RPG НА ОСНОВІ РУШІЯ UNITY

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22017113

Здобувачка _____ О. А. Шерстюк
підпис

«___»_____ 2024 р.

Керівник ст. викладач _____ С. Ю. Боровльова
підпис

«___»_____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
підпис

«___»_____ 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії програмного забезпечення, канд. техн. наук, доцент

_____ Є. О. Давиденко

«___» _____ 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано здобувачці групи 408 факультету комп'ютерних наук

Шерстюк Олені Андріївні

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Ігровий застосунок у жанрі Action-RPG на основі рушія Unity

Затверджена наказом Ректора ЧНУ ім. Петра Могили від «22» грудня 2023 р. № 269

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

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

Очікуваним результатом є розробка та реалізація ігрового застосунку у жанрі Action-RPG, шляхом використання рушія Unity

4. Перелік питань, що підлягають розробці:

проведення аналізу аналогічних систем та дослідження предметної області, дослідження особливостей роботи з рушієм Unity, проєктування ігрового середовища, створення відповідних моделей, розробка та реалізація ігрового середовища, тестування та відлагодження ігрового застосунку.

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 Медичного інституту	Спеціальна частина з охорони праці

Керівник роботи ст. викладач Боровльова Світлана Юріївна
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання Шерстюк Олена Андріївна
(прізвище, ім'я, по батькові)

_____
(підпис)

Дата видачі завдання «16» січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Ігровий застосунок у жанрі Action-RPG на основі рушія Unity»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	16.01.2024	16.01.2024	Виконано
2	Ознайомлення з літературою та аналіз теми роботи	17.01.2024	21.01.2024	Виконано
3	Складання календарного плану КРБ	22.01.2024	22.01.2024	Виконано
4	Аналіз предметної області	23.01.2024	10.02.2024	Виконано
5	Розробка проєктних рішень	11.02.2024	25.02.2024	Виконано
6	Моделювання та конструювання ПЗ	26.02.2024	15.03.2024	Виконано
7	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування	16.03.2024	28.04.2024	Виконано
8	Розробка спеціальної частини з охорони праці	14.05.2024	20.05.2024	Виконано
9	Відгук керівника КРБ	22.05.2024	22.05.2024	Виконано
10	Оформлення КРБ та презентації	23.05.2024	01.06.2024	Виконано
11	Попередній захист	03.06.2024	05.06.2024	Виконано
12	Рецензування	13.06.2024	15.06.2024	Виконано
13	Завершення оформлення КРБ та презентації	15.06.2024	16.06.2024	Виконано
14	Захист кваліфікаційної роботи	24.06.2024	25.06.2024	Виконано

Розробила здобувачка Шерстюк Олена Андріївна _____
(прізвище, ім'я, по батькові) *(підпис)*

«22» січня 2024 р.

Керівник роботи ст. викладач Боровльова Світлана Юріївна _____
(посада, прізвище, ім'я, по батькові) *(підпис)*

«22» січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Ігровий застосунок у жанрі Action-RPG на основі рушія Unity»

Здобувачка 408 гр.: Шерстюк Олена Андріївна

Керівник: ст. викладач Боровльова Світлана Юріївна

Кваліфікаційна робота бакалавра присвячена створенню ігрового застосунку у жанрі Action-RPG на основі рушія Unity.

Об’єктом роботи є процеси, які пов’язані з організацією та створенням ігрового застосунку в жанрі Action-RPG.

Предметом роботи є програмні засоби та інформаційні технології для розробки ігрового застосунку в жанрі Action-RPG.

Метою кваліфікаційної роботи є розробка гри на рушії Unity для популяризації жанру Action-RPG.

Кваліфікаційна робота бакалавра складається з вступу, чотирьох розділів, висновків, спеціальної частини з охорони праці та безпеки в надзвичайних ситуаціях та переліку джерел посилання.

У вступі визначається актуальність, науково-практичне значення обраної теми, об’єкт та предмет дослідження, а також мета та завдання роботи.

У першому розділі проводиться системний аналіз обраної предметної сфери та, на його основі, формується постановка задачі та специфікація вимог до програмного забезпечення.

У другому розділі розробляються проєктні рішення, що забезпечуються виконання специфікації вимог до програмного забезпечення. Цей процес включає в себе моделювання об’єкту та предмету дослідження, а також створення функціональних та інформаційних моделей програмного забезпечення.

У третьому розділі описується виконана робота з моделювання та конструювання програмного забезпечення. Вибір мови програмування, рушія, розробка концепту гри та її геймплею.

У четвертому розділі демонструється виконана робота з кодування розробленого програмного забезпечення.

У висновках оцінюються отримані результати відносно аналогів, практичне значення результатів та рекомендації щодо їх впровадження у певній сфері діяльності.

У спеціальній частині з охорони праці та безпеки в надзвичайних ситуаціях розглянуто питання охорони праці, які безпосередньо пов’язані з діяльністю розробника програмного забезпечення.

КРБ викладена на 66 сторінок, вона містить 4 розділи, 54 ілюстрації, 3 таблиці, 20 джерел в переліку посилань.

Ключові слова: Unity, C#, Action-RPG, розробка ігрових застосунків

ABSTRACT

of the Bachelor’s Thesis

"Action-RPG game application based on Unity engine"

Student of group 408: Sherstiuk Olena Andriivna

Supervisor: Senior Lecturer Borovleva Svitlana Yuriivna

The bachelor’s thesis is devoted to the creation of a game application in the genre of Action-RPG based on the Unity engine.

The object of work is the process of organization and creation of a game application in the genre of Action-RPG.

The subject of the work is software and information technology for the development of a game application in the genre of Action-RPG.

The purpose of the thesis is to develop a game on the Unity engine to popularize the Action-RPG genre.

A bachelor’s thesis consists of an introduction, four chapters, conclusions, a special part on occupational health and safety in emergencies, and a list of references.

The introduction defines the relevance, scientific, and practical significance of the chosen topic, the object and subject of the study, as well as the purpose and objectives of the work.

The first section conducts a systematic analysis of the selected subject area and, on its basis, formulates the problem statement and specification of software requirements.

In the second section, design solutions are developed to ensure that the software requirements specification is met. This process includes modeling the object and subject of research, as well as creating functional and information models of the software.

The third section describes the work done on software modeling and design. The choice of programming language, engine, development of the game concept, and its gameplay.

The fourth section demonstrates the work done on coding of the developed software.

The conclusions evaluate the obtained results concerning analogs, the practical significance of the results, and recommendations for their implementation in a particular field of activity.

The special part on occupational health and safety in emergencies considers occupational health and safety issues that are directly related to the activities of a software developer.

The bachelor’s thesis consists of 66 pages, it contains 4 sections, 54 illustrations, 3 tables, and 20 sources in the list of references.

Keywords: Unity, C#, Action-RPG, game application development

ЗМІСТ

ВСТУП

У наш час комп’ютерні ігри є одним із найбільш динамічних і перспективних сегментів розважальної індустрії. Їх вплив на культуру та економіку продовжує зростати, відкриваючи нові можливості для розвитку творчості, технологій та навіть способу життя. До того ж, комп’ютерні ігри стають не лише формою розваги, а й інструментом навчання, тренування навичок та розвитку креативності. Вони проникають у різні сфери нашого життя, від освіти та медицини до бізнесу та соціальної сфери. Крім того, ігрова індустрія продовжує активно впливати на формування молодіжної культури та громадської думки, надаючи значний вплив на свідомість та поведінку людей. Тому дослідження різних аспектів процесу розробки комп’ютерних ігор залишається актуальною і важливою темою.

Для розробки ігрових застосунків використовуються різні ігрові рушії такі як, Unity, Unreal Engine, Godot тощо. Unity – це потужний і популярний багатоплатформний двигун для розробки ігор та застосунків. Він надає можливість створювати інтерактивні проекти, підтримуючи майже 30 платформ, у тому числі персональні комп’ютери, мобільні пристрої, інтернет-програми, ігрові консолі, пристрої доповненої реальності (AR) або віртуальної реальності (VR) та інші платформи. Unity відомий своєю гнучкістю, широким набором інструментів та підтримкою різних технологій, роблячи його ідеальним вибором для початківців та досвідчених розробників.

Завдяки зручному Drag & Drop інтерфейсу та функціональному графічному редактору двигун Unity дозволяє створювати сцени та розташовувати об’єкти в реальному часі, надаючи можливість негайного тестування. Його інтерфейс поділено на кілька панелей, кожна з яких відповідає за певну функцію.

Unity може бути використана для створення базових моделей, однак для розробки більш деталізованих і унікальних 3D-моделей буде використано Blender. Blender – це безкоштовне та відкрите програмне забезпечення для 3D-моделювання та анімації. Blender включає в себе великий пакет програм, що надають інструменти

для всіх етапів створення 3D графіки: моделювання, анімації, створення реалістичних ефектів, рендерингу, об’єднання шарів зображення тощо.

Об’єктом роботи є процеси, які пов’язані із організацією та створенням ігрового застосунку в жанрі Action-RPG.

Предметом роботи є програмні засоби та інформаційні технології для розробки ігрового застосунку в жанрі Action-RPG.

Метою кваліфікаційної роботи є розробка гри на рушії Unity для популяризації жанру Action-RPG.

Для досягнення поставленої мети, потрібно виконати наступні завдання:

- описати та проаналізувати роботу Unity;
- дослідити та схарактеризувати особливості сучасних ігор у жанрі Action-RPG;
- розробити інтерфейс та графічний дизайн;
- спроєктувати ігровий застосунок;
- реалізувати ігровий застосунок на рушії Unity;
- перевірити якість роботи застосунку.

Область застосування: запропонований ігровий застосунок призначений для створення багатого та динамічного ігрового досвіду, що поєднує в собі захоплюючі дії з глибокими елементами рольової гри, дозволяючи гравцям поринути в епічні пригоди і формувати свою власну долю в ігровому світі. Він підтримує свободу дій гравця, надаючи безліч шляхів до успіху, сюжетні лінії, що розгалужуються, і осмислені рішення, які визначають результат гри.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність предметної області

Оскільки ігрова індустрія продовжує зростати і розвиватися, предметна область комп'ютерних ігор є міждисциплінарною, має елементи інформатики, мистецтва, психології, соціології тощо. Вона пропонує багату та динамічну сферу навчання для тих, хто цікавиться створенням та аналізом інтерактивного цифрового досвіду. Предметна область комп'ютерних ігор охоплює широкий спектр тем, пов'язаних із дизайном, розробкою, аналізом та культурним впливом відеоігор.

Комп'ютерні ігри – це один із напрямків комп'ютерної індустрії. До комп'ютерних ігор можна віднести спеціально створені програми з геймплеєм, тобто програми з певним ігровим сюжетом, ігровою механікою, що дозволяють взаємодіяти гравцю з ігровим процесом.

Термін «відеоігри» є більш широким, він охоплює ігри на різних платформах, включаючи консолі (наприклад, PlayStation, Xbox), персональні комп'ютери, мобільні пристрої та інші ігрові пристрої. Тоді як термін «комп'ютерна гра» частіше використовується в контексті, де основна увага приділяється іграм, в які грають на настільних або портативних комп'ютерах.

Комп'ютерні ігри беруть свій початок з 1940 року. Тоді американський фізик та засновник квантової механіки, що працював в електротехнічній компанії Westinghouse, створив першу в світі комп'ютерну гру – Nimatron [1]. Суть гри: на ігровому полі по купках розкладено певну кількість предметів (сірників, каменів, фішок, кісток тощо), гравці по черзі беруть предмети (можна взяти скільки завгодно, але тільки з однієї купки), програє той, кому дістанеться останній предмет.

Рисунок 1.1 – Перша в світі комп’ютерна гра Nimatron

У період з 1950-х по 1970-і роки індустрія відеоігор перебувала в зародковому стані, і вони були в основному спрощеними та зосереджені на базовій ігровій механіці через технологічні обмеження. Більшість ранніх ігор були побудовані на основі однієї концепції, наприклад, Pong (1972), в якій використовувалися дві ракетки та м’яч, або Spacewar! (1962) – космічна бойова гра для двох гравців [1].

На початку 80-х можна було, не дивлячись ні на назви, ні на обкладинку, сказати: «Це платформер», і майже напевно вгадати. Зробити перше припущення про жанр гри можна вже за її зовнішнім виглядом, дивлячись на скріншоти. Наприклад, якщо на них простежується вигляд від першої особи – це, напевно, шутер. А якщо в грі присутні дрібні фігурки, на які гравець дивиться звідкись зверхньо, це напевно або стратегія, або RPG. Але щоб сказати точніше, треба, звичайно ж, грати. Жанр визначають закладені у гру механіки. Які завдання в ній потрібно виконувати і як, чи є в грі головоломки, яка в ній зброя, вогнепальна чи холодна – все це підкаже напрямок. Однак помилитися та заплутатися все одно легко.

На сьогоднішній день кількість комп’ютерних ігор просто величезна, і часто вони класифікуються за характеристиками чи завданнями, а не за типом геймплея. Тому категорії ігор, або жанри, можуть поділятися на під жанри, а одна гра цілком може належати до кількох жанрів одразу. Жанри ігор можуть бути дуже різноманітними і іноді перетинатися, але ось загальна класифікація з деякими підкатегоріями [2]:

а) «Дія» (Action):

– платформер (Platformer): необхідно переміщатися по оточенню, стрибаючи чи іншим чином переміщаючись платформами;

– шутер (Shooter): орієнтований на стрілянину по ворогах або мішенях, поділяється на: шутер від першої особи (FPS), шутер від третьої особи (TPS);

– файтинг (Fighting): гравці борються один з одним або з противниками, створеними штучним інтелектом, віч-на-віч або в командних битвах;

– виживання (Survival): гравець повинен вижити в умовах небезпек навколишнього середовища, часто з обмеженими ресурсами;

– стелс (Stealth): гравець повинен уникнути виявлення під час виконання завдань;

– ритм (Rhythm): гравці повинні слідувати ритму, щоб прогресувати.

б) «Пригоди» (Adventure):

– екшн пригоди (Action-Adventure);

– обери та натисни (Point-and-Click): гравці взаємодіють із навколишнім середовищем, головним чином, натискаючи на об’єкти та переміщаючись за сюжетом;

– відкритий світ (Open World): дозволяє гравцям вільно досліджувати велике відкрите середовище;

– текстові (Text-Based): більшість ігрового процесу, розвитку сюжету відбувається у вигляді текстової взаємодії.

в) «Рольові ігри» (Role-Playing Games (RPG)):

– екшн рольові ігри (Action RPG);

– покрокова рольова гра (Turn-Based RPG);

– масова багатокористувацька гра (Massively Multiplayer Online (MMORPG));

– тактична гра (Tactical RPG (TRPG)).

г) «Стратегія» (Strategy):

– стратегія в реальному часі (Real-Time Strategy (RTS)): гравці керують арміями або цивілізаціями в режимі реального часу, беруть участь у боях та управлінні ресурсами;

– покрокова стратегія (Turn-Based Strategy (TBS)): гравці по черзі приймають стратегічні рішення та часто включають управління арміями чи цивілізаціями;

– захист вежі (Tower Defense): гравці повинні захищатись від хвиль ворогів, стратегічно розміщуючи захисні споруди;

– 4X (eXplore, eXpand, eXploit, eXterminate): орієнтовані на створення цивілізацій чи імперій і управління ними.

д) «Симулятор» (Simulation):

– симулятор життя (Life Simulation): моделює аспекти життя, такі як відносини, кар’єра та повсякденна діяльність;

– симулятор транспортних засобів (Vehicle Simulation): гравці керують транспортними засобами, такими як літаки, автомобілі тощо.

е) «Спорт» (Sports):

– командні види спорту (Team Sports);

– індивідуальні види спорту (Individual Sports).

ж) «Головоломки» (Puzzle):

– зіставлення плиток (Tile-Matching);

– ігри засновані на фізиці (Physics-Based): гравці вирішують головоломки, маніпулюючи об’єктами та використовуючи принципи фізики;

– ігри на логіку (Logic).

з) «Жахіття» (Horror):

– жахливе виживання (Survival Horror);

– хоррор дія (Action Horror): ігри, у яких елементи страху поєднуються з динамічним ігровим процесом.

и) «Гонки» (Racing);

к) «Освітні» (Educational).

Ці категорії та підкатегорії не є вичерпними, і деякі ігри можуть належати до кількох жанрів чи під жанрів. Крім того, у міру розвитку ігор можуть з’являтися нові жанри та під жанри.

1.2 Огляд аналогів

У процесі розробки ігрового застосунку було проведено детальне дослідження аналогів у жанрі Action-RPG, особливо тих, які розроблені з використанням рушія Unity. Цей аналіз охоплює різні аспекти, включаючи ігрову механіку, функції, недоліки та переваги використання, стратегії взаємодії. Вивчаючи існуючі ігри цього жанру та платформи, слід виявити успішні елементи та потенційні недоліки і зрештою використати ці знання для інформування та покращення зусиль з розробки ігрового застосунку.

Рисунок 1.2 – Аналоги створюваного ігрового застосунку

Таблиця 1.1 – Основні характеристики Grim Dawn

Назва	Grim Dawn
Розробник	Crate Entertainment
Архітектура	Desktop application
Мова реалізації	C#
Функції	– система двох класів, що дозволяє створювати різноманітні збірки персонажів; – хардкорний режим для додаткових випробувань; – багатокористувацький кооперативний режим та режим PvP; – велика система видобутку з рідкісними та легендарними предметами; – глибока крафтова та зачарувальна механіка;
Переваги	– приваблива графіка та захоплюючий дизайн світу; – гнучке налаштування складності гри; – глибокі системи налаштування персонажа.
Недоліки	– обмежена різноманітність типів ворогів та їх поведінки; – деякі проблеми із продуктивністю на деяких системах; – відсутність підтримки платформ macOS та Linux.
Посилання	https://www.grimdawn.com/

Таблиця 1.2 – Основні характеристики Path of Exile

Назва	Path of Exile
Розробник	Grinding Gear Games
Архітектура	Desktop application
Мова реалізації	C#
Функції	– величезне дерево навичок із тисячами можливих білдів; – велика система крафту; – сезонні ліги з унікальними випробуваннями та нагородами; – хардкорний режим для додаткових випробувань; – PvP-арени та турніри; – динамічна економіка з торгівлею керована гравцями.
Переваги	– глибока ігрова механіка та розвиток персонажа; – можливість повторного проходження; – регулярні оновлення.
Недоліки	– періодичні проблеми з балансом певних навичок, предметів чи механік; – проблеми зі стабільністю сервера під час пікового навантаження; – обмежені внутрішньоігрові посібники та посібники для складних систем.
Посилання	https://www.pathofexile.com/

Таблиця 1.3 – Основні характеристики Torchlight II

Назва	Torchlight II
Розробник	Runic Games
Архітектура	Desktop application
Мова реалізації	C#
Функції	– чотири ігрові класи персонажів з унікальними здібностями; – система тварин-компаньйонів; – різноманітна кількість квестів та завдань; – велике дерево навичок з розгалуженими шляхами; – режим New Game Plus для підвищеної складності.
Переваги	– доступний ігровий процес, що підходить як для звичайних, так і для хардкорних гравців; – привабливий художній стиль та саундтрек; – офлайн-режим, що дозволяє грати без підключення до Інтернету.
Недоліки	– обмежена глибина налаштування персонажів; – багатокористувацька інфраструктура залежить від серверів, розміщених гравцями, що призводить до потенційних проблем з підключенням; – випадкові помилки та глюки, особливо в мультиплеєрі.
Посилання	https://www.torchlight2.com/en

Аналіз успішних аналогів допоможе визначити фактори, які сприяли їхньому успіху, такі як ігрова механіка, художній стиль, сюжет або маркетингові стратегії. Так само вивчення менш успішних ігор може дати уявлення про те, яких пасток слід уникати.

1.3 Специфікація вимог до програмного забезпечення

Призначення системи: створення багатого та динамічного ігрового досвіду, що поєднує в собі захоплюючі дії з глибокими елементами рольової гри, дозволяючи гравцям поринути в епічні пригоди і формувати свою власну долю в ігровому світі. Ігровий застосунок підтримує свободу дій гравця, надаючи безліч шляхів до успіху, сюжетні лінії, що розгалужуються, і осмислені рішення, які визначають результат гри.

Область застосування: область розваг, реклама та маркетинг, розвиток ігрової індустрії.

Характеристики користувачів: користувачі віком від 18 років з ПК та доступом до мережі Інтернет.

Функції системи:

- два головних героя;
- розгалужені шляхи та дослідження світу;
- режим спільної гри;
- велика крафтова система;
- різноманітні навички;
- випадкові квести та зустрічі;
- різноманітні головоломки;
- динамічна система діалогів;
- система зв’язків між персонажами;
- можливість купівлі та продажу зброї та ресурсів;
- різні класи та підкласи ворогів та босів;
- можливість повторного проходження гри;
- зміна складності гри;
- динамічна погода та зміна дня та ночі;
- взаємодія з навколишнім середовищем.

Вимоги до технічного забезпечення:

- операційна система: Windows 10 або вище;
- процесор: Intel Core i5-6600K або AMD Ryzen 5 1600X;
- відеокарта: NVIDIA GeForce GTX 970 або AMD Radeon R9 390X;
- оперативна пам’ять: не менше 8 ГБ;
- місце на жорсткому диску: не менше 50 ГБ.

Архітектура програмної системи: складається з клієнтської частини.

Системне програмне забезпечення: для розробки ігрового застосунку на основі рушія Unity необхідно встановити такі системні програмні засоби, як Windows 10 або вище, DirectX 12 або вище, Visual Studio 2019 або вище, а також Blender версії 2.60 або вище для створення 3D-моделей та анімацій.

Мова і технологія розробки ПЗ: Рекомендовано використовувати мову програмування C#.

Інтерфейс користувача: Інтерфейс користувача має бути інтуїтивно зрозумілим та зручним для використання. Необхідно створити добре продуманий інтерфейс, який дозволить користувачам легко керувати персонажем, взаємодіяти з ігровим світом та керувати настройками гри.

Висновки до розділу 1

У цьому розділі було проаналізовано предметну сферу комп’ютерних ігор. Розглянуто перші створені комп’ютерні ігри, їх принцип роботи. Виділено основні жанри та під жанри сучасних комп’ютерних ігор. Схарактеризовано ігри жанру Action-RPG на основі рушія Unity. Проведений аналіз аналогів дозволив виявити успішні елементи та потенційні недоліки для покращення створюваного ігрового застосунку. А також встановлено специфікації вимог до програмного забезпечення.

2 АНАЛІЗ СИСТЕМИ

2.1 Створення сценаріїв використання

Короткий usecase:

Користувач запускає ігровий застосунок на своєму пристрої, після чого він потрапляє до головного меню гри. У головному меню гри можна обрати різні опції, такі як: створення нової гри, продовження збереженої гри та інші доступні опції. Після вибору опції «Нова гра», користувач має обрати персонажа, яким він буде керувати у грі. Якщо користувач успішно вибрав персонажа, розпочинається генерація світу, після чого гравець може взаємодіяти з ігровим світом.

Поверхневий usecase:

Головний сценарій (успішний)

Користувач запускає ігровий застосунок на своєму пристрої. У головному меню гри він обирає «Продовжити гру». Відбувається генерація світу та завантаження усіх збережених даних гравця. Гравець може вибирати завдання та/або квести, які включатимуть боротьбу з ворогами, збір ресурсів, дослідження ігрового світу тощо. Різноманітні ресурси можуть бути використані для крафту предметів та розвитку персонажа. Після досягнення певного рівня розвитку персонажа, гравець може вибирати та покращувати певні навички персонажа та розвивати зв’язки з командою.

Альтернативні сценарії:

– гравець обирає опцію «Нова гра» та ігрового персонажа для початку нової гри. Він може зберегти прогрес свого персонажа у перший слот профілю, а у другий – прогрес іншого персонажа;

– гравець обирає опцію «Налаштування» для зміни графічний налаштувань, звукових ефектів, чутливість контролера тощо;

– гравець обирає опцію «Вийти з гри» після збереження ігрового процесу;

– гравець обирає опцію «Продовжити гру», але з’являється вікно з помилкою;

– гравець може спробувати перезавантажити гру або звернутись у технічну підтримку для вирішення проблеми.

Повний usecase

Use Case Name: Гравець розпочинає нову гру в жанрі Action-RPG

Scope: System

Level: User-goal

Primary Actor: Гравець

Stakeholders and interests:

– гравець: бажає розпочати нову гру в жанрі Action-RPG;

– розробники: бажають, щоб гравець успішно розпочав нову гру та насолоджувався геймплеєм.

Preconditions:

– користувач має доступ до комп’ютера;

– на комп’ютері встановлено ігровий застосунок на основі рушія Unity;

– гравець має створений обліковий запис у грі.

Success guarantee: Гравець успішно запустить нову гру в жанрі Action-RPG на пристрої.

Main Success Scenario:

– гравець запускає ігровий застосунок;

– гравець обирає у головному меню гри опцію «Нова гра»;

– гравець обирає головного персонажа, яким буде керувати;

– система створює нову гру з обраним головним персонажем.

Extensions:

– гравець не успішно розпочинає гру через технічні проблеми:

– система виводить повідомлення про проблему та пропонує шляхи її вирішення;

– гравець повторно запускає ігровий застосунок;

– якщо проблема не була вирішена, гравець звертається до технічної підтримки.

Special Requirements:

– гра повинна мати систему автоматичного збереження прогресу гравця;

– гра повинна мати систему навчання, що допоможе гравцю ознайомитись з основними елементами гри.

Technology and Data Variations List: Рекомендується використовувати Blender версії 2.60 та новіше для створення 3D-моделей та анімацій.

Frequency of Occurrence: Частота використання usecase при використанні системи залежить від кількості гравців, що нею користуються.

Miscellaneous: Можна додати багатокористувацький режим.

2.2 Діаграма варіантів використання

Діаграми варіантів використання (Use Case Diagram) описують високорівневі функції та область дії системи. Ці діаграми також визначають взаємодію між системою та її учасниками [3]. Варіанти використання та актори на цих діаграмах описують, що робить система і як актори її використовують, але не те, як система працює всередині.

Розглянемо діаграму використання для ігрового застосунку, що проєктується (рис. 2.1):

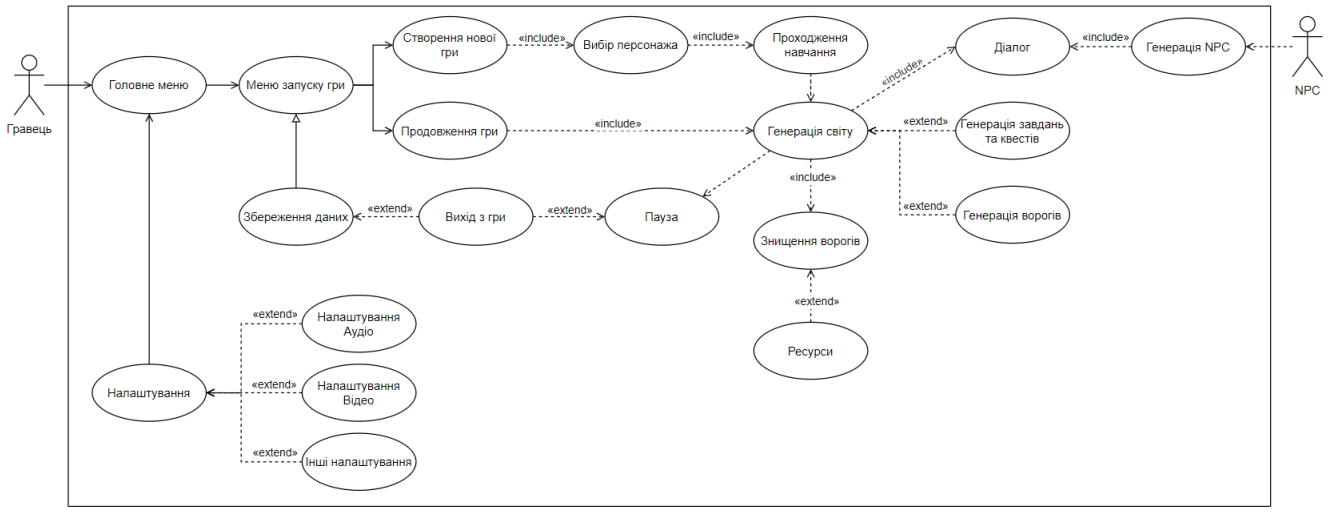

Рисунок 2.1 – Діаграма варіантів використання

Згідно вищенаведеної діаграми гравець може змінити налаштування звуку та відео, а також розпочати нову гру або увійти у збережену. У гравця є можливість вибору головного персонажа після чого він має пройти навчання. Також на діаграмі присутній ще один актор – NPC, до якого може звертатись гравець. Протягом гри гравець може виконувати різноманітні квести та завдання, знищувати ворогів та отримувати ресурси з них. В ігровому застосунку присутня можливість збереження ігрового процесу при виході з гри.

2.3 Побудова діаграм взаємодії (послідовності та кооперації)

У моделях UML взаємодія – це поведінка, яка є спілкуванням між одним або декількома учасниками. Діаграма послідовності (Sequence Diagram) – це діаграма взаємодії UML, яка моделює повідомлення, що передаються між учасниками, такими як об’єкти та ролі, а також керуючі та умовні структури, такі як об’єднані фрагменти [4]. Вона використовується для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання.

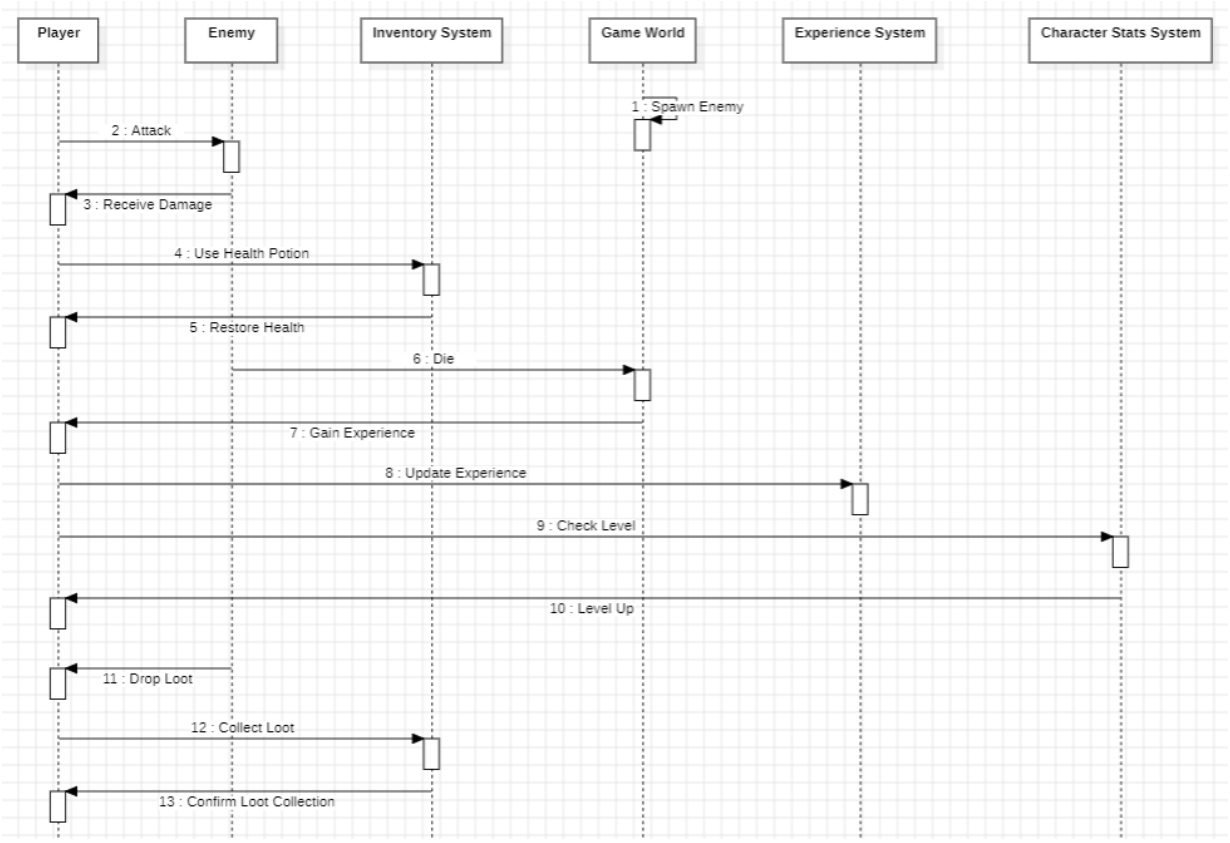

Рисунок 2.2 – Діаграма послідовності для варіанту використання «Combat»

Від «Game World» до «Game World» – Message: Spawn Enemy.

Від «Player» до «Enemy» – Message: Attack.

Від «Enemy» до «Player» – Message: Receive Damage.

Від «Player» до «Inventory System» – Message: Use Health Potion.

Від «Inventory System» до «Player» – Message: Restore Health.

Від «Enemy» до «Game World» – Message: Die.

Від «Game World» до «Player» – Message: Gain Experience.

Від «Player» до «Experience System» – Message: Update Experience.

Від «Player» до «Character Stats System» – Message: Check Level.

Від «Character Stats System» до «Player» – Message: Level Up.

Від «Enemy» до «Player» – Message: Drop Loot.

Від «Player» до «Inventory System» – Message: Collect Loot.

Від «Inventory System» до «Player» – Message: Confirm Loot Collection.

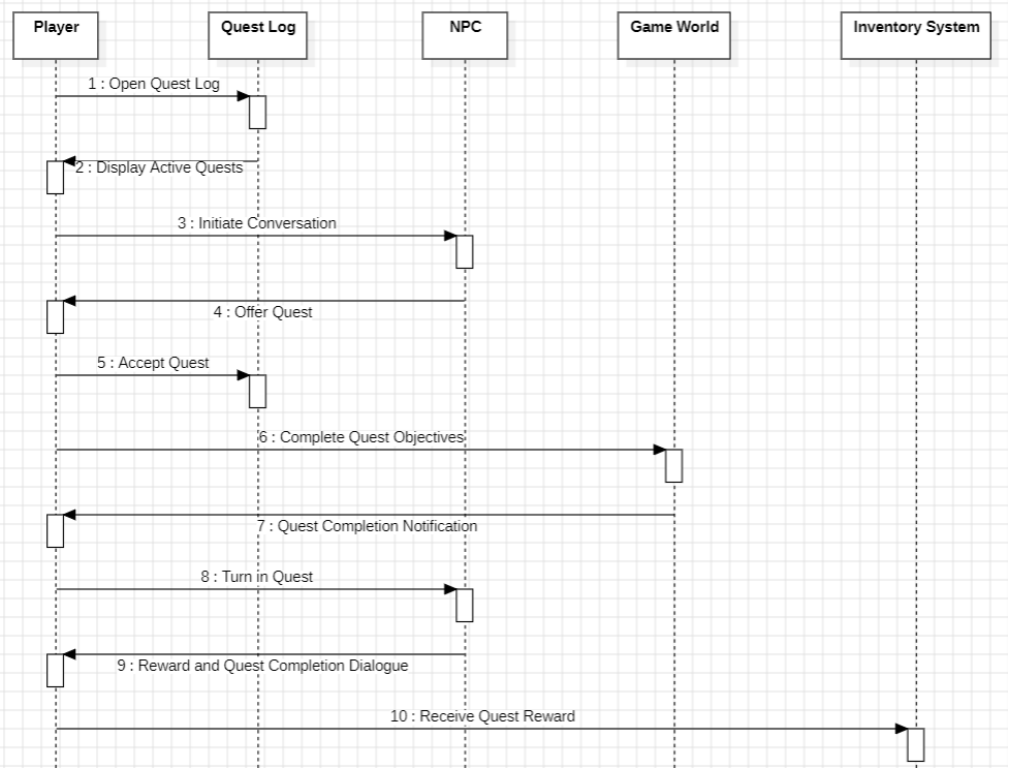

Рисунок 2.3 – Діаграма послідовності для варіанту використання «Quest Completion»

Від «Player» до «Quest Log» – Message: Open Quest Log.

Від «Quest Log» до «Player» – Message: Display Active Quests.

Від «Player» до «NPC» – Message: Initiate Conversation.

Від «NPC» до «Player» – Message: Offer Quest.

Від «Player» до «Quest Log» – Message: Accept Quest.

Від «Player» до «Game World» – Message: Complete Quest Objectives.

Від «Game World» до «Player» – Message: Quest Completion Notification.

Від «Player» до «NPC» – Message: Turn in Quest.

Від «NPC» до «Player» – Message: Reward and Quest Completion Dialogue.

Від «Player» до «Inventory System» – Message: Receive Quest Reward.

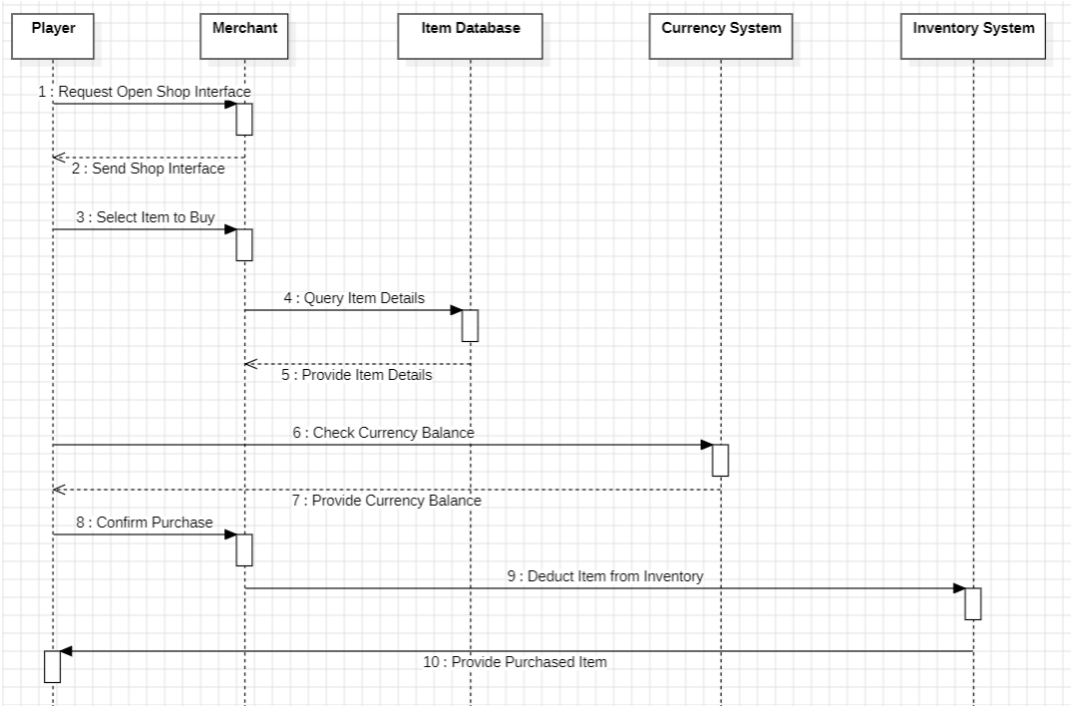

Рисунок 2.4 – Діаграма послідовності для варіанту використання «Buying Items from a Merchant»

Від «Player» до «Merchant» – Message: Request Open Shop Interface.

Від «Merchant» до «Player» – Message: Send Shop Interface.

Від «Player» до «Merchant» – Message: Select Item to Buy.

Від «Merchant» до «Item Database» – Message: Query Item Details.

Від «Item Database» до «Merchant» – Reply Message: Provide Item Information.

Від «Player» до «Currency System» – Message: Check Currency Balance.

Від «Currency System» до «Player» – Reply Message: Provide Currency Balance.

Від «Player» до «Merchant» – Message: Confirm Purchase.

Від «Merchant» до «Inventory System» – Message: Deduct Item from Inventory.

Від «Inventory System» до «Player» – Message: Provide Purchased Item.

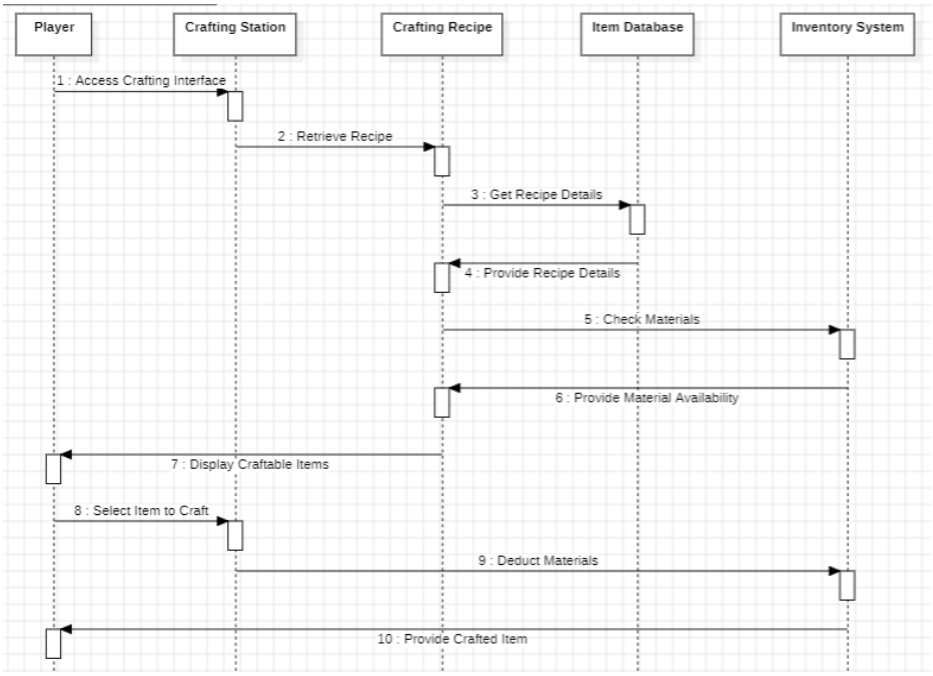

Рисунок 2.5 – Діаграма послідовності для варіанту використання «Crafting an Item»

Від «Player» до «Crafting Station» – Message: Access Crafting Interface.

Від «Crafting Station» до «Crafting Recipe» – Message: Retrieve Recipe.

Від «Crafting Recipe» до «Item Database» – Message: Get Recipe Details.

Від «Item Database» до «Crafting Recipe» – Message: Provide Recipe Details.

Від «Crafting Recipe» до «Inventory System» – Message: Check Materials.

Від «Inventory System» до «Crafting Recipe» – Message: Provide Material Availability.

Від «Crafting Recipe» до «Player» – Message: Display Craftable Items.

Від «Player» до «Crafting Station» – Message: Select Item to Craft.

Від «Crafting Station» до «Inventory System» – Message: Deduct Materials.

Від «Inventory System» до «Player» – Message: Provide Crafted Item.

Діаграма кооперації (Collaboration Diagram) описує поведінку системи лише на рівні окремих об’єктів, які обмінюються між собою повідомленнями, щоб досягти потрібної мети чи реалізувати певний варіант використання. Таке уявлення структури моделі як сукупності взаємодіючих об’єктів забезпечує діаграма кооперації [5].

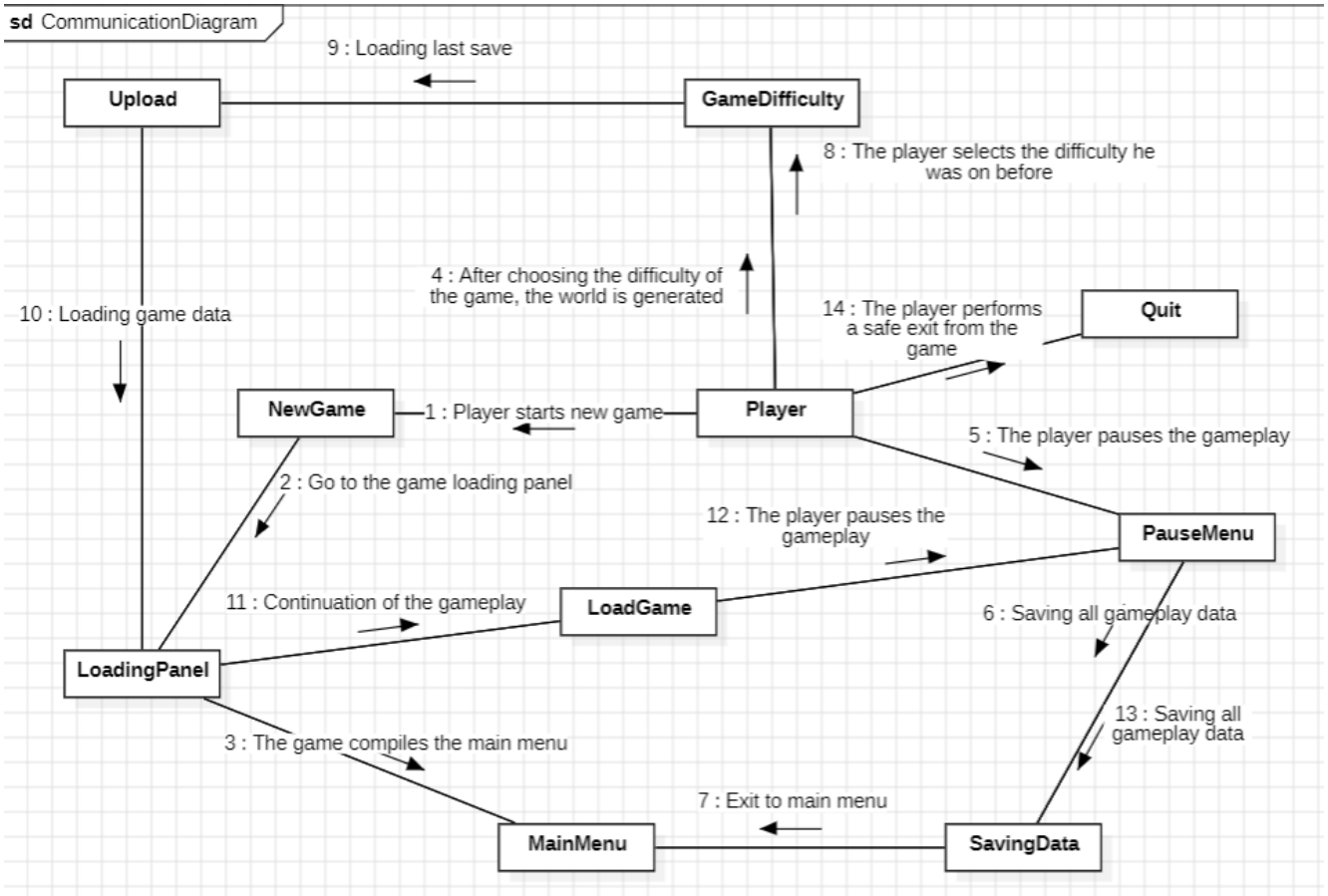

Рисунок 2.6 – Діаграма кооперації

На діаграмі кооперації розміщуються об’єкти, які є екземпляри класів, зв’язку з-поміж них, які у своє чергу є екземплярами асоціацій і повідомлення. Зв’язки доповнюються стрілками повідомлень, при цьому показуються тільки ті об'єкти, які беруть участь у реалізації кооперації, що моделюється.

2.4 Діаграми станів

Діаграма станів (Statechart Diagram) показує перехід об’єкта з одного стану в інший. Вона використовується для моделювання динамічних аспектів системи. Ця

діаграма корисна при моделюванні життєвого циклу об’єкта [6]. Від інших діаграм діаграма станів відрізняється тим, що описує процес зміни станів лише одного екземпляра певного класу – одного об’єкта, причому об’єкта реактивного, тобто об’єкта, поведінка якого характеризується його реакцією на зовнішні події.

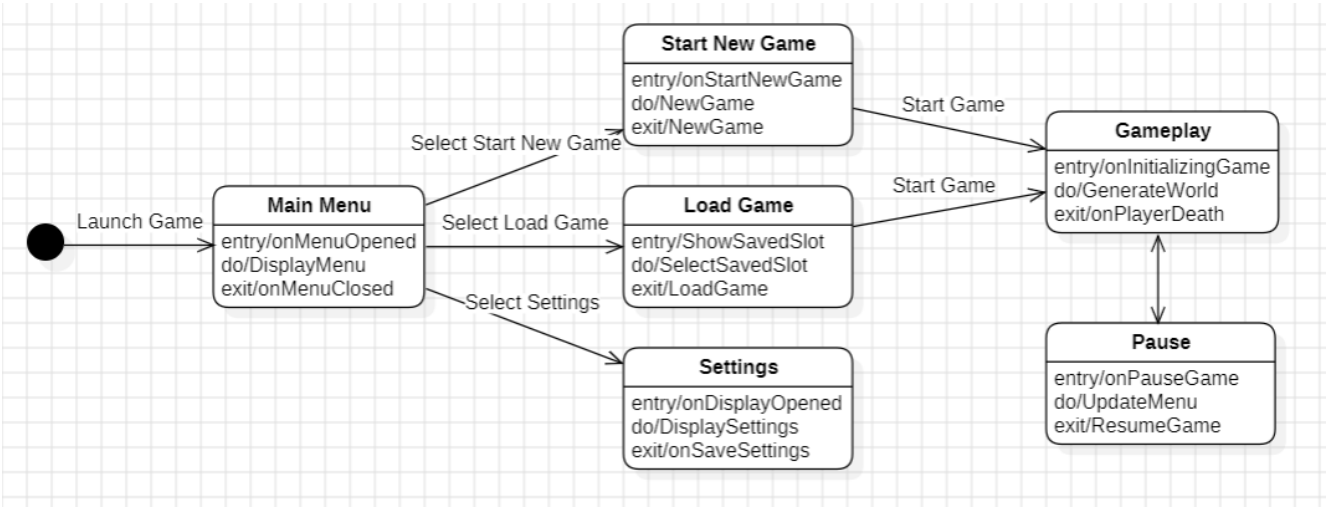

Рисунок 2.7 – Загальна діаграма станів

На загальній діаграмі стану (рис. 2.7) для головного меню відображено вибір режиму гри, де гравець може вибрати нову гру чи продовжити попередньо збережену. У налаштуваннях є можливість зміни звуку та відео. Також в головному меню гри є можливість вийти з гри, яка також доступна і з меню паузи чи смерті гравця.

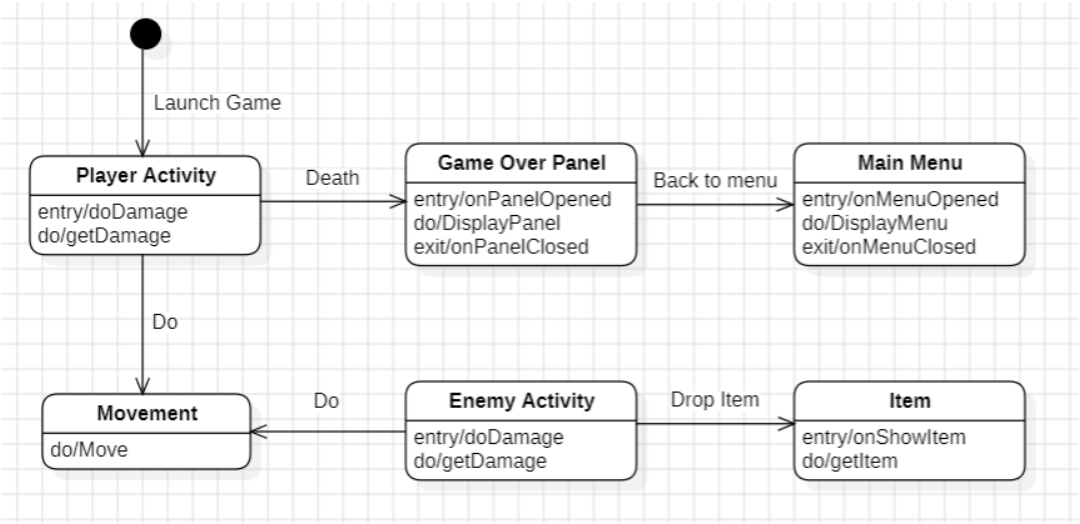

Рисунок 2.8 – Діаграма станів гравця

На рис. 2.8 відображено можливість пересування гравця, нанесення та отримання пошкоджень. Для ворога є можливість застосовувати ті ж самі можливості. Якщо здоров’я закінчиться, то з’явиться вікно повідомлення про поразку.

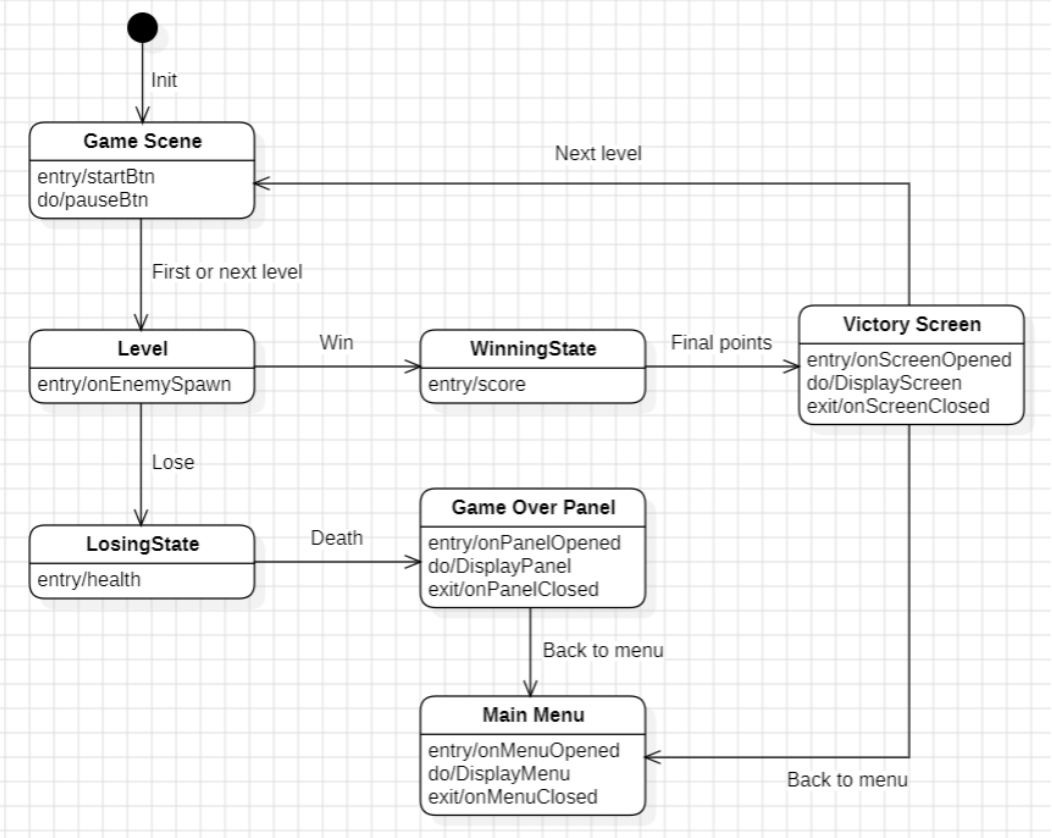

Рисунок 2.9 – Діаграма станів геймплею

Відповідно до діаграми станів геймплею (рис. 2.9), після початку гри відбувається генерація певного рівня. Якщо гравець успішно пройде рівень, перед ним з’явиться вікно перемоги з підрахунком очок, інакше – вікно програшу з можливість повернутись до головного меню.

2.5 Створення мокапів

Для створення мокапів для ігрового застосунку (див. рис. 2.10-2.14) було використано онлайн інструмент Moqups [7]. Moqups – це просте та потужне рішення для створення професійних каркасів для веб-сайтів, програм та інформаційних панелей. Редактор, де створюються дизайни, містить дві основні

панелі – ліву з інструментами та праву із функціями для форматування. На панелі інструментів розташовані основні елементи, які користувач може додати до проекту, керування сторінками, завантаження зображень та іконок. Права панель дозволяє редагувати додані елементи, змінювати їх розмір, колір, форми, шрифт та стилі тексту, додавати ефекти. До елементів належать: кнопки, форми, рамки для тексту, посилання, слайдери, таблиці, геометричні фігури та інші компоненти створення інтерфейсів.

Рисунок 2.10 – Мокап головного меню ігрового застосунку

У головному меню гри присутні чотири кнопки: «New Game» (для початку нової гри), «Load Game» (для завантаження збереженої гри), «Settings» (налаштування гри) та «Exit» (вихід з гри), кожна з яких відповідає за певну дію.

Рисунок 2.11 – Панель виходу з гри

На рис. 2.11 зображено мокап діалогового вікна про вихід з гри, яке використовується для взаємодії з гравцем під час ігрового процесу. Це вікно містить елементи інтерфейсу, що дозволяють гравцеві зробити вибір щодо завершення гри, такі як кнопки «Так» та «Ні».

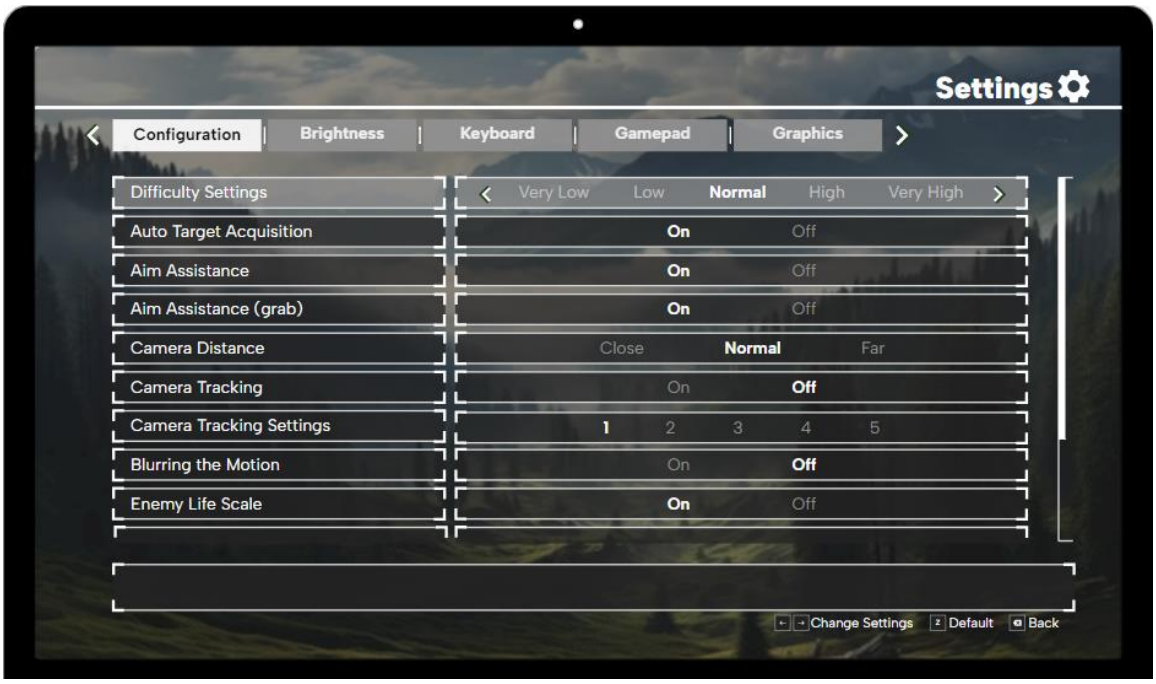

Рисунок 2.12 – Панель налаштувань гри

У панелі налаштування гри присутні різноманітні розділи такі, як: конфігурація, яскравість, клавіатура, геймпад та графіка. В кожному розділі присутні відповідні підрозділи, які може змінювати користувач.

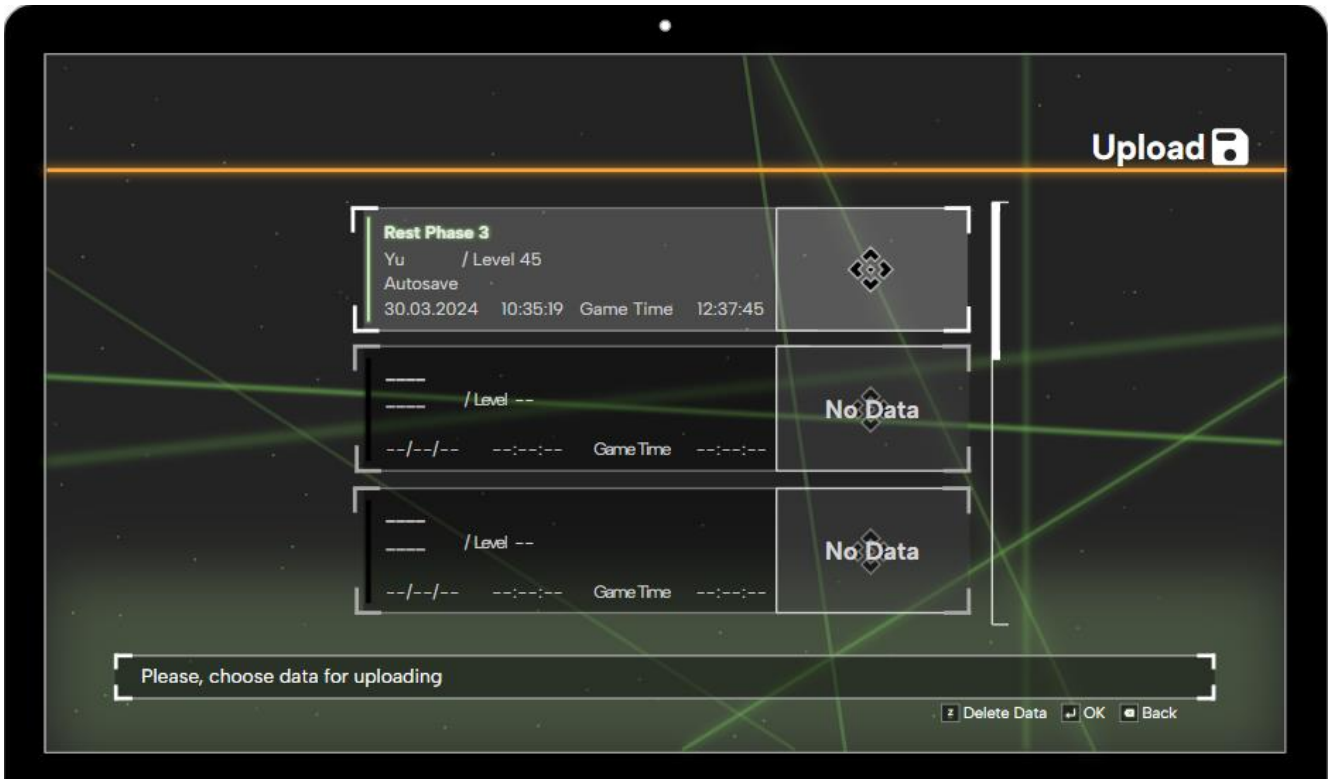

Рисунок 2.13 – Мокап панелі зі збереженими ігровими процесами

В ігровому застосунку реалізована можливість зберігання даних гри у різні комірки, що дозволяє збільшити розмаїття команд, їх навичок та ресурсів. Цей підхід дозволяє гравцям ефективно керувати та налаштовувати свої гри, зберігаючи поточний прогрес і вибрані налаштування. Мокап інтерфейсу для збереження даних включає в себе відповідні елементи для введення та вибору інформації, зручно розташовані на екрані, що сприяє легкій та інтерактивній взаємодії користувачів з системою збереження. Це надає гравцям можливість зручно керувати своїми налаштуваннями та збереженим прогресом в грі, забезпечуючи позитивний імерсійний досвід гри.

Рисунок 2.14 – Мокап панелі спорядження персонажів

За допомогою панелі спорядження персонажів можна обирати для кожного з них певні навички, зброю, а також броню. Тут також відображаються такі дані, як максимальна кількість здоров’я персонажа, його рівень, кількість сили, атаки та захисту, кількість мани та золота, а також поточний час.

Висновки до розділу 2

У цьому розділі було створено сценарії використання з застосуванням наступних форм написання: коротка, поверхнева та повна форма. Побудовано діаграму варіантів використання, на якій було описано що робить система і як актори її використовують. За допомогою діаграми послідовності було більш детально описано логіку створених сценаріїв використання. А діаграма кооперації відображає уявлення структури моделі як сукупності взаємодіючих об’єктів. Для створюваного ігрового застосунку було розроблено п’ять мокапів з відображенням інтерфейсу гри та її основними складовими.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІГРОВОГО ЗАСТОСУНКУ

3.1 Розробка UML-діаграм

Діаграма UML – це спосіб візуалізації систем та програмного забезпечення з використанням уніфікованої мови моделювання (Unified Modeling Language). Інженери-програмісти створюють діаграми UML, щоб зрозуміти дизайн, архітектуру коду та запропоновану реалізацію складних програмних систем. Діаграми UML також використовуються для моделювання робочих процесів та бізнес-процесів.

Існують дві основні категорії: структурні діаграми та поведінкові діаграми. Структурні діаграми відображають елементи системи, що моделюється. Говорячи більш технічною мовою, вони показують різні об’єкти у системі. Поведінкові діаграми показують, що має відбуватися у системі. Вони описують, як об’єкти взаємодіють один з одним, створюючи систему, що функціонує.

3.1.1 Діаграма класів

Діаграма класів UML – це основний будівельний блок будь-якого об’єктно-орієнтованого рішення, що визначає типи класів в системі та різного роду зв’язки, які існують між ними. На діаграмах класів зображуються також атрибути класів, операції класів та обмеження, що накладаються на зв’язки між класами. Атрибути описують властивості об’єктів класу, а операції – це функції чи перетворення, що можуть мати параметри та повертати значення [8].

Клас представлений прямокутником, який містить три відсіки, розташованих вертикально: верхній відсік містить ім’я класу і є обов’язковим, а два нижні відсіки містять докладну інформацію про атрибути класу, а також операції або поведінку класу.

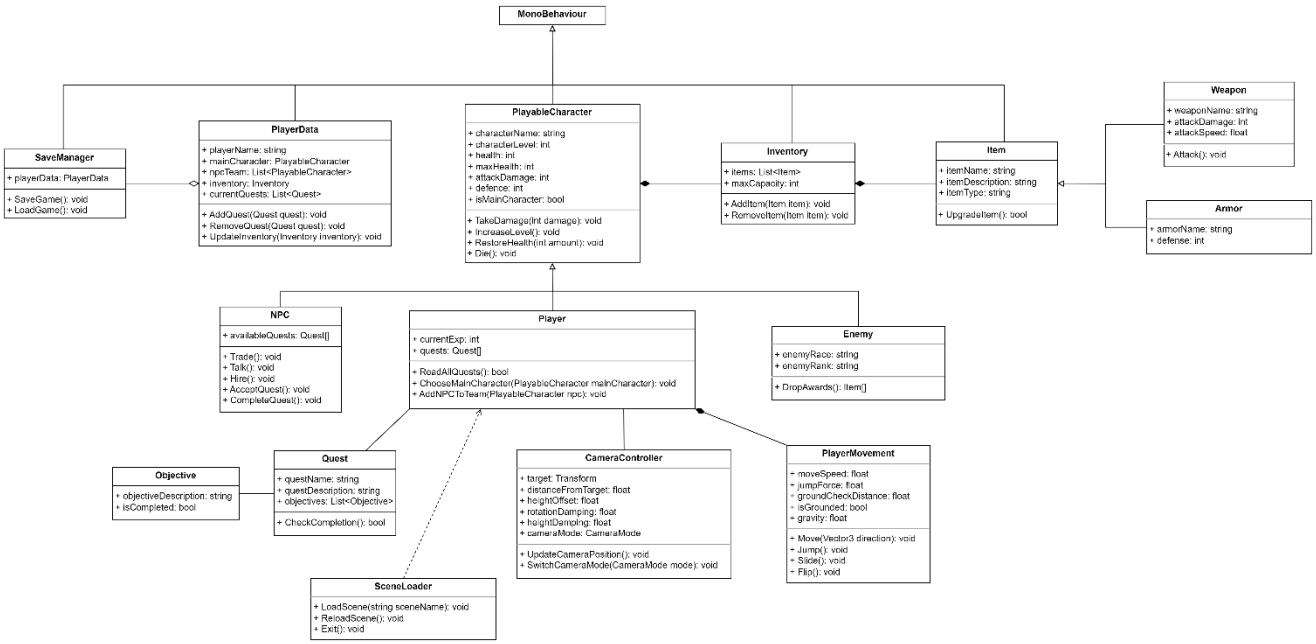

Рисунок 3.1 – Діаграма класів

Клас PlayableCharacter представляє персонажа, яким може керувати гравець.

Клас Player відображає гравця у грі.

Клас PlayerMovement керує механікою руху гравця, включаючи швидкість, стрибки та ковзання.

Клас CameraController управляє ігровою камерою, дозволяючи їй слідувати за гравцем.

Клас SceneLoader керує завантаженням та перезавантаженням ігрових сцен.

Клас Enemy відображає ворожого персонажа у грі.

Клас NPC представляє у грі неігрового персонажа.

Клас Weapon демонструє предмет зброї в грі.

Клас Armor є предметом броні в грі.

Клас Inventory керує інвентарем гравця, включаючи додавання та видалення предметів.

Клас Item представляє загальний предмет у грі з такими атрибутами, як ім’я, опис та тип.

Клас Quest являє собою квест у грі з назвою, описом та цілями, які необхідно виконати.

Клас Objective є цілю квесту, вказуючи на завдання, яке необхідно виконати.

Клас SaveManager керує збереженням та завантаженням ігрових даних, включаючи інформацію про гравця та ігровий прогрес.

Клас PlayerData містить дані, пов’язані з гравцем, включаючи ім’я, персонажів, інвентар та квести.

3.1.2 Діаграма компонентів та розгортання

Діаграма компонентів (Component Diagram) розбиває складну систему на більш дрібні компоненти і візуалізує взаємозв’язок між цими компонентами.

Діаграма компонентів дозволяє визначити архітектуру системи, що розробляється, встановивши залежності між програмними компонентами. Основний тип сутностей на ній – це компоненти, інтерфейси та залежності між ними [9].

Компонент є окремою частиною системи, яка виконує певну функцію або має певну роль. Він може бути програмним модулем, класом, бібліотекою, сервісом, фізичним пристроєм тощо. Він зазвичай відображається у вигляді прямокутника з його ім’ям або позначенням.

Інтерфейси є місцями, де групи класів компонента взаємодіють з іншими компонентами системи. Альтернативний спосіб представлення інтерфейсів – розширення символів із поля компонента. Інтерфейс, що надається на діаграмі компонентів, вказує на те, які можливості або функції надає конкретний компонент для взаємодії з іншими компонентами або системами. Необхідний інтерфейс діаграми компонентів вказує на інтерфейси, які компонент очікує з інших компонентів.

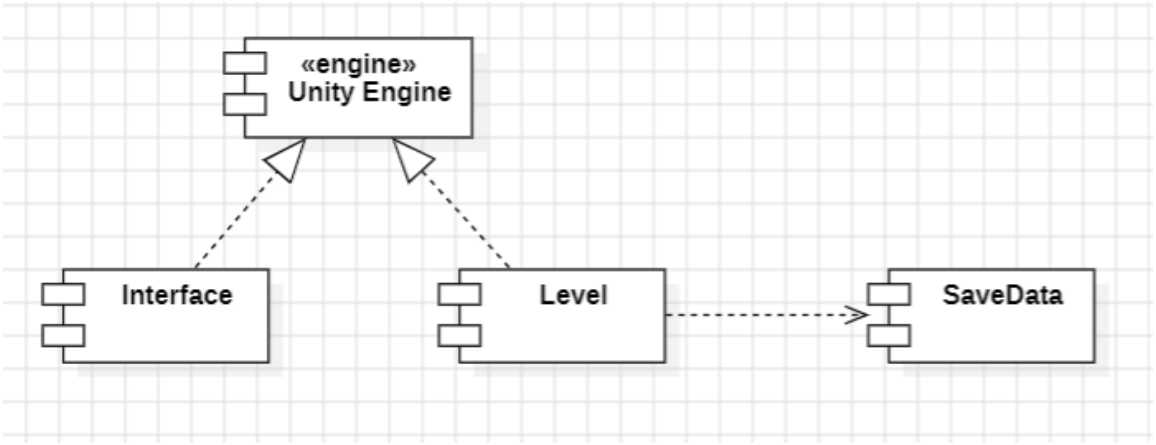

Рисунок 3.2 – Діаграма компонентів

На діаграмі компонентів (рис. 3.2) відображені інтерфейси, які вказують на можливості та функції кожного компонента для взаємодії з іншими частинами системи.

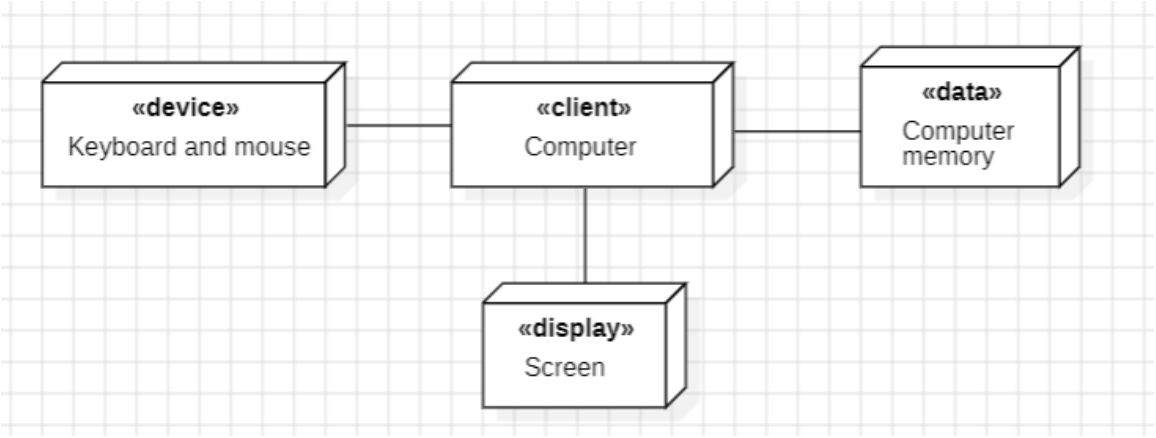

Рисунок 3.3 – Діаграма впровадження

Подання засобів впровадження (deployment view) відображає програмні засоби на вузли обчислювальних систем (processing nodes). На ньому можна побачити конфігурацію елементів обробки (processing element) і програмних процесів, що працюють на них. Подання засобів впровадження враховує такі потреби, як доступність системи, надійність, швидкодія та масштабованість. Щоб продемонструвати різні вузли обчислювальних систем та зв’язок між ними, створюються діаграми впровадження (deployment diagram).

Така діаграма дозволяє розробникам архітектури зрозуміти топологію системи та відобразити компоненти на виконувані процеси. Тут враховуються такі

аспекти: процесорна архітектура, швидкість, ємність, пропускна спроможність каналів взаємодії процесів, фізичне розташування апаратних ресурсів, технологія розподіленої обробки.

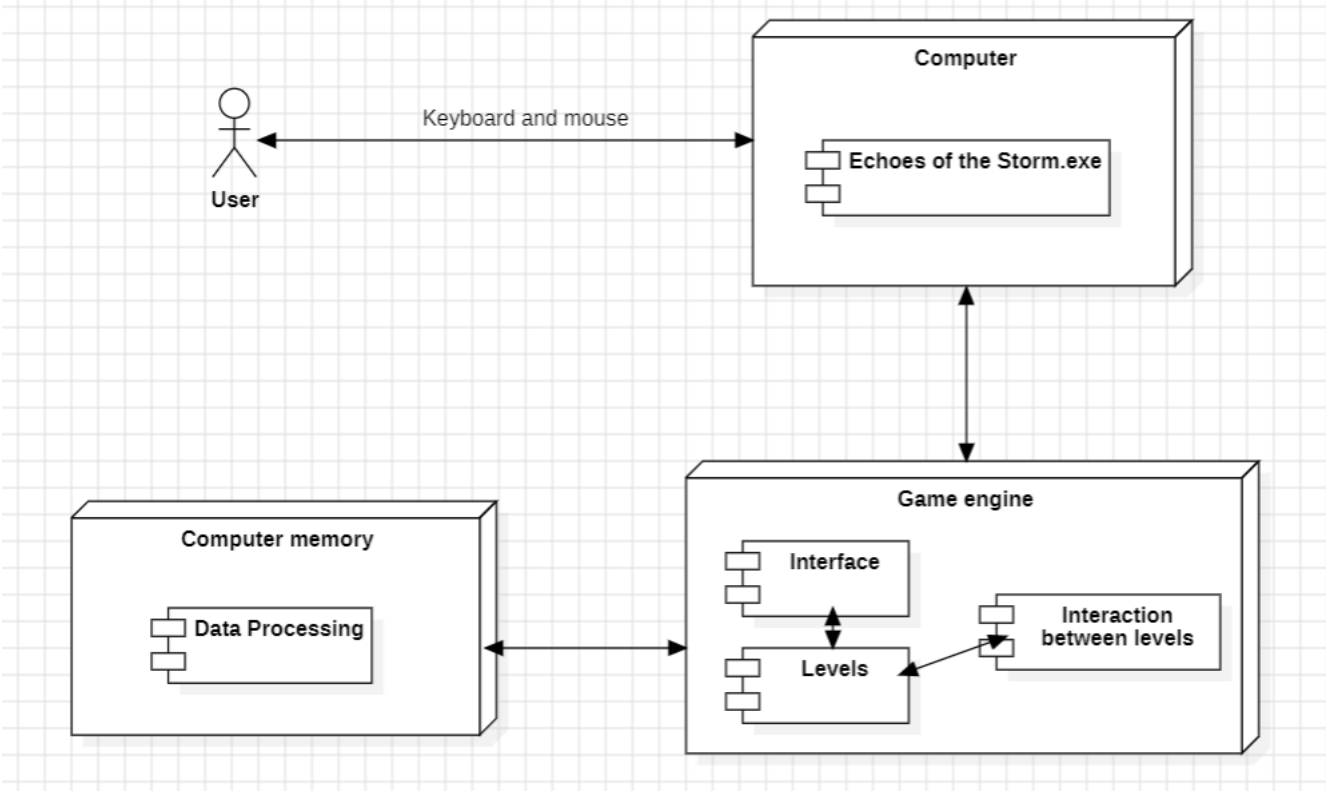

Рисунок 3.4 – Діаграма розгортання

Діаграма розгортання описує один із аспектів самої системи, а саме – фізичне розгортання інформації, виробленої програмою на апаратних компонентах. Іншими словами – це структурна схема, яка відображає архітектуру системи як розгортання (дистрибуції) програмних артефактів. Артефактами є конкретні елементи у фізичному світі, які є результатом процесу розробки.

До складу діаграми розгортання входить кілька типів фігур UML. Тривимірні блоки або вузли символізують базові програмні або апаратні елементи системи. Лінії, що йдуть від одного вузла до іншого, застосовуються для позначення зв’язків, а менші фігури, розташовані всередині блоків, – для програмних артефактів, які розгортаються на вузлах.

3.1.3 Діаграма пакетів

Діаграми пакетів наочно демонструють залежність між пакетами у складі системи. Пакет зображується у вигляді папки і застосовується для організації елементів моделі (наприклад, класів або сценаріїв використання) групи [10].

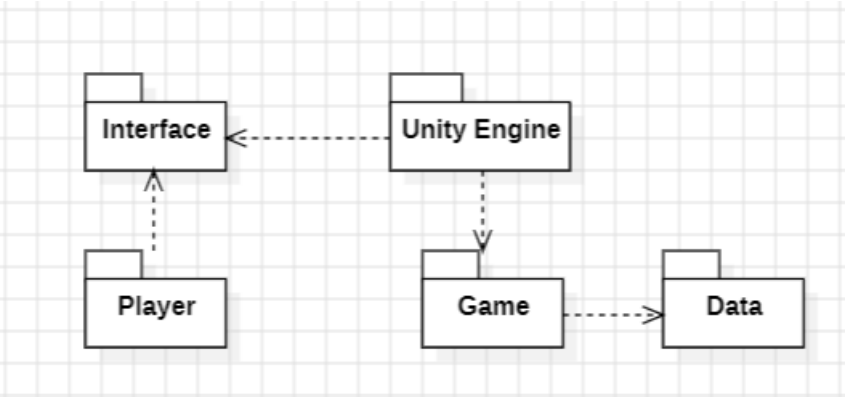

Рисунок 3.5 – Діаграма пакетів

Пакет Player (класи PlayableCharacter, Player, PlayerMovement, NPC, Enemy, Weapon, Armor, Inventory та Item), що відповідає за дії гравця, та Unity Engine (SceneLoader, CameraController), що виступає рушієм всього проекту, не залежать від інших пакетів, натомість вони впливають на пакети Interface (клас UIManager), який реагує на взаємодію гравця з інтерфейсом та тим, що відбувається у грі, та Game. У свою чергу пакет Game (Quest, Objective), що відповідає за геймплей, впливає на пакет Data (клас SaveManager та клас PlayerData), який містить у собі логіку збереження необхідної для гри інформації.

3.2 Огляд фреймворків та засобів розробки

Під час розробки ігрового застосунку було використано наступний стек технологій:

- ігровий рушій: Unity;
- мова програмування: C#;
- 3D-моделювання та анімація: Blender;
- графічний редактор: Photoshop.

Вибір правильних програмних інструментів має вирішальне значення для ефективної та успішної розробки ігрового застосунку.

3.2.1 Рушій Unity

Відеоігри перетворилися з простих піксельних екранів на неймовірно деталізовані віртуальні світи завдяки потужності ігрових двигунів. Ігровий двигун – це програмне забезпечення, що надає розробникам інструменти та засоби для створення ігор. Його функції можуть сильно відрізнятися, але зазвичай включають підтримку рендерингу графіки, фізичного моделювання, введення, звуку, штучного інтелекту та анімації [11].

Популярні ігрові двигуни, такі як Unity, Unreal Engine та Godot, надають розробникам потужні та гнучкі інструменти для втілення своїх ігрових ідей у життя.

Рисунок 3.6 – Логотипи ігрових двигунів Unity, Unreal Engine та Godot

Unity виділяється на ринку розробки інді-ігор своєю винятковою простотою використання та надійною підтримкою спільноти. Його універсальність проявляється як при розробці 2D-, так і 3D-ігор, що робить його ідеальним вибором для широкого спектру ігрових проєктів. Привабливість Unity полягає в тому, як він адаптується до унікальних навичок та потреб кожного розробника [11].

Інтерфейс користувача приємний у використанні, інтуїтивно зрозумілий і вимагає мало часу для освоєння, незважаючи на його багату функціональність. Зручність використання Unity забезпечує високу швидкість прототипування, що досягається за рахунок підтримки інтерактивного налаштування та налагодження, що дозволяє розробнику вносити зміни під час гри.

Завдяки Unity можна створювати контент на більш ніж 20 платформах, що відображено на рис. 3.7.

Рисунок 3.7 – Платформи, що підтримує Unity

Всі вищенаведені платформи можна умовно поділити на [12]:

Desktop:

– Windows (PC);
– Mac;
– Universal Windows Platform (UWP);
– Linux Standalone.

Mobile:

– iOS;
– Android.

Extended Reality (XR):

– ARKit;
– ARCore;
– Microsoft HoloLens;

– Windows Mixed Reality;
– Magic Leap (Lumin);
– Oculus;
– PlayStation VR.

Consoles:

– PS5;
– PS4;
– Xbox One;
– Xbox X|S;
– Nintendo Switch;
– Google Stadia.

WebGL.

Embedded:

– Embedded Linux;
– QNX.

Наочний приклад гри, створеної на Unity, яка підтримує різні платформи – Genshin Impact, успішний проєкт китайської студії miHoYo Limited. Більш популярною стала її мобільна версія, але користувачі можуть увійти до облікового запису, наприклад, з комп’ютера та продовжити грати з того ж моменту, на якому зупинилися в мобільній версії. Крім Genshin Impact, на Unity створено такі відомі проєкти, як Hearthstone, Outlast, Cuphead, Pokemon GO та багато інших.

Unity чудово взаємодіє з різними сторонніми інструментами та програмами. Наприклад, при інтеграції з Photoshop, Blender та інших програм розробники можуть легко експортувати графічні та 3D ресурси напряму в Unity без втрати якості або додаткових конвертацій. Це прискорює процес створення контенту та забезпечує безшовну співпрацю між дизайнерами та розробниками. З іншого боку Unity також пропонує використання хмарних сервісів для спільної роботи. Ці сервіси, такі як Unity Collaborate, Unity Teams, Photon Unity Networking, дозволяють

командам працювати над одним проєктом у реальному часі, забезпечуючи синхронізацію змін та зручне керування версіями.

3.2.2 Мова програмування C#

C# (C-Sharp) – це об’єктно-орієнтована мова програмування загального призначення, що використовується для розробки широкого спектру програм, включаючи корпоративне програмне забезпечення, відеоігри та мобільні програми.

Представлений Microsoft у 2002 році разом із випуском Visual Studio .NET, C# є частиною сімейства мов C, до якого також входять C та C++. C# є мовою із Сі-подібним синтаксисом і близький у цьому відношенні до C++ та Java.

C# надає базову, читабельну мову для побудови логіки застосунку, приховуючи при цьому більшу частину складності, що лежить в основі властивих мові можливостей. На сьогоднішній час мова стандартизована згідно специфікації ISO/IEC 23270: Інформаційні технології. Мови програмування. C#.

Рисунок 3.8 – Логотип C#

Мова C# була створена спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше. Фреймворк .NET представляє потужну платформу створення додатків. Можна виділити такі основні риси:

- підтримка декількох мов програмування;
- кросплатформність;
- міцна бібліотека класів;
- різноманітність технологій;
- продуктивність.

C# легко інтегрується з двигуном Unity. Розробники мобільних застосунків можуть використовувати його практично на будь-якому сучасному мобільному пристрої або консолі з використанням кросплатформної технології, такої як Xamarin.

Програми C# можна зберігати у кількох вихідних файлах. Під час компіляції програми C# всі вихідні файли обробляються разом, при цьому вони можуть вільно посилатися один на одного.

Отже, C# – це не тільки відмінний вибір для програмістів-початківців, але і багато захоплюючих можливостей для тих, хто вже знайомий з мовою, що поставляються в кожній новій версії. Завдяки своїй універсальності, сучасним функціям, сильній спільноті та підтримці Microsoft, C# є дуже простим у вивчені.

3.2.3 Програмне забезпечення Blender

Blender – це програмне забезпечення для 3D-анімації, яке використовується для створення анімаційних фільмів, візуальних ефектів, інтерактивних 3D-додатків та відеоігор. Він обслуговує широке коло підприємств: від кіностудій до розробників ігор.

Рисунок 3.9 – Логотип Blender

Програма надає розширені функції, які зазвичай зустрічаються в інших, більш дорогих додатках, таких як Maya або Autodesk. Проте Blender може працювати практично на будь-якому комп’ютері завдяки своїй відкритій архітектурі.

Ключові функції Blender включають моделювання, оснащення, анімацію, рендеринг, композитинг та відстеження руху. Він дозволяє користувачам створювати 3D-об’єкти, використовуючи різні методи, такі як видавлювання, розподіл поверхонь або інструменти редагування сітки.

Найбільшою перевагою користувальницького інтерфейсу Blender є його візуальний характер: під час роботи з вашою 3D-моделлю буде показано лише те, що необхідно для конкретної дії. Це робить навігацію сценою швидше, ніж в інших програмах, де доступ до кожної опції здійснюється через кілька меню.

Blender підходить як для 3D-анімації, так і для 2D-анімації. Завдяки його інструментам можна одразу займатися кількома напрямками роботи без використання додаткового ПЗ.

Blender надає можливість створювати 3D-об’єкти у повнофункціональному пакеті для створення та використовувати їх у дружньому, гнучкому, кросплатформному ігровому рушії такому як Unity. Ця функція дозволяє використовувати кожен інструмент там, де він найкраще підходить для повного робочого процесу розробки 3D-ігор.

Імпортувати об’єкти з Blender у Unity двома способами. Один передбачає використання файлу .blend, а інший файлу .fbx.

Кожна нова версія програми поповнюється новими інструментами та утилітами для роботи з різними проєктами – симулятор течії, спеціальні фільтри для рендерингу, фізичні двигуни, що повністю настроюються, і системи згоряння – все це перетворює Blender на багатоцільовий і повнофункціональний інструмент.

3.2.4 Графічний редактор Adobe Photoshop

Photoshop – це програмне забезпечення для створення зображень, графічного дизайну та редагування фотографій, розроблене Adobe у 1988 році. Створений Томасом і Джоном Нолламі, він спочатку був розроблений для комп’ютерів Macintosh, але тепер доступний для платформ Windows і MacOS.

Рисунок 3.10 – Логотип Adobe Photoshop

Photoshop є частиною Adobe Creative Cloud, в яку входять інші популярні інструменти, такі як Adobe Illustrator, Photoshop Lightroom та Adobe Dreamweaver. Photoshop Creative Cloud дозволяє користувачам працювати із зображеннями та графічним контентом де завгодно.

Photoshop спеціально розроблений, щоб дозволити користувачам створювати та редагувати растрові зображення у кількох шарах. Ці накладання або шари можуть підтримувати прозорість, а також діяти як маски або фільтри, які можуть змінювати базові зображення шарів під ними. Можна застосовувати тіні та інші ефекти, такі як альфа-композитинг. До цих шарів також можна застосувати декілька колірних моделей – CMYK, RGB, Spot Color, колірний простір Duotone та Lap.

Photoshop чудово підходить для створення графіки та макетів для друкованих проєктів, таких як газети, журнали та плакати. Програмне забезпечення також може створювати дизайн веб-сайтів, логотипи та інше цифрове мистецтво.

Зображення, відредаговані у Photoshop, можна зберегти в різних форматах файлів, включаючи ті, які підходять для Інтернету (JPEG, PNG, GIF) або друку (TIFF).

Висновки до розділу 3

Цей розділ було присвячено моделюванню та проєктуванню ігрового застосунку, а також огляду стеку технологій, що було використано. Для відображення елементів системи, що моделюється, було застосовано такі структурні діаграми, як діаграма класів, пакетів, компонентів та розгортання. В якості стеку технологій для створення ігрового застосунку було використано рушій Unity, мову програмування C#, ПЗ Blender та графічний редактор Adobe Photoshop.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

4.1 Налаштування інтерфейсу та графічного дизайну

Зараз відеоігри стають все більш складними, із заплутаними сюжетними лініями, величезними віртуальними світами та численними ігровими механіками. Тому розробникам дуже важливо розставити пріоритети в дизайні інтерфейсу користувача, щоб гравці могли легко зрозуміти і використовувати ці функції. Дизайн інтерфейсу гри – це процес розробки інтерфейсу відеоігри. Інтерфейс користувача в іграх – це частина гри, з якою безпосередньо взаємодіє гравець. Він включає все: від меню і кнопок до значків, проєкційних дисплеїв (HUD), карт та інших інтерактивних компонентів. Ігровий користувацький інтерфейс можна класифікувати на чотири категорії.

Дієгетичний (diegetic) – це інтерфейс, який включено до ігрового світу. Іншими словами, це тип інтерфейсу, який персонажі гри можуть бачити, чути та чіпати. Наприклад, у Dead Space індикатор здоров’я на спині костюма ІКС, а у Alone in the Dark інвентар у вигляді розкладених усередині піджака предметів.

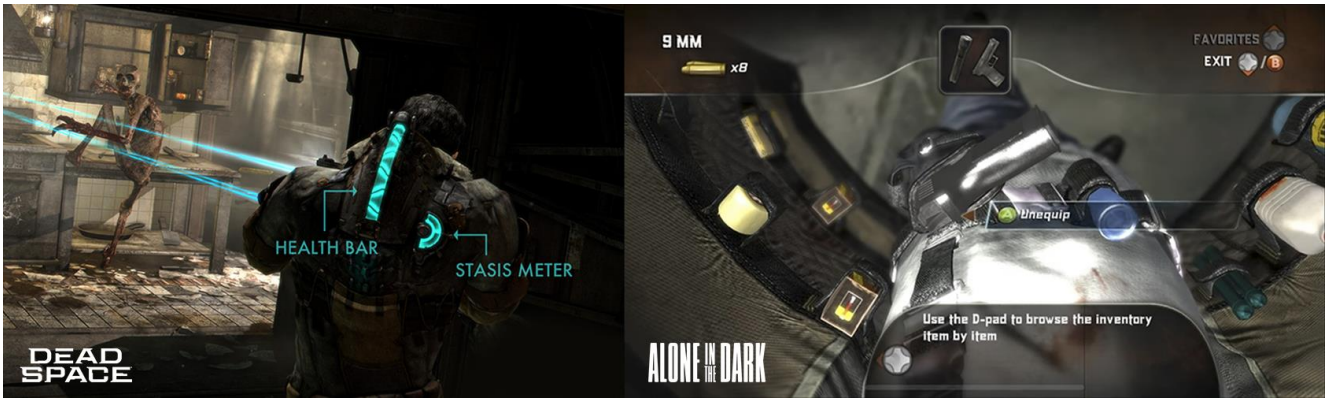

Рисунок 4.1 – Інтерфейс Dead Space та Alone in the Dark

Недієгетичний (non-diegetic) інтерфейс – це зворотний підхід, коли UI просто накладається на екран та існує тільки для гравця. Наприклад, у Doom є відображення показників здоров’я, запасу патронів та обраної зброї, а у Call of Duty Black Ops 2 – HUD та міні-карти.

Рисунок 4.2 – Інтерфейс Doom та Call of Duty Black Ops 2

Просторовий (spatial) інтерфейс має на увазі присутність безпосередньо в ігровому світі елементів інтерфейсу, наприклад маркерів квестів або підсвічених супротивників, проте в реальності ігрового світу вони також залишаються несуттєвими і служать лише потребам гравця. Наприклад, у Legend of Zelda: Breath of the Wild над головами ворогів з’являються стрілки, які вказують на кого нападатиме Лінк. А у World of Warcraft: Wrath of the Lich King можна побачити знаки оклику над головами NPC.

Рисунок 4.3 – Інтерфейс Legend of Zelda: Breath of the Wild та World of Warcraft: Wrath of the Lich King

Мета (meta) інтерфейс включає об’єкти, які начебто є частиною світу гри, але в той же час не існують у ньому фізично. До нього відносяться камера, що йде за персонажем гравця та змінює положення залежно від того, що відбувається на екрані, або звук биття серця та бризки крові, що покривають цю камеру в момент небезпеки для гравця на знак близькості загибелі протагоніста. Інше поширене використання цього типу – мобільний телефон, який з’являється на екрані, але

мається на увазі, що з ним взаємодіє ігровий персонаж. Persona 5, Catherine, Watchdogs та Grand Theft Auto 5 використовують цей тип інтерфейсу для взаємодії з мобільним телефоном.

Рисунок 4.4 – Інтерфейс Grand Theft Auto 5 та Watchdogs

Ефективний дизайн інтерфейсу користувача забезпечує чітке представлення важливої інформації, такої як індикатори здоров’я, бали, кількість боєприпасів і цілі. Цей візуальний зворотний зв’язок інформує гравців та допомагає їм брати участь у ігровому процесі.

Рисунок 4.5 – Інтерфейс ігрового застосунку

Для ігрового застосунку у жанрі Action-RPG на основі рушія Unity використано недієгенетичний та просторовий інтерфейс.

4.1.1 Сцени

Сцени (Scenes) в Unity містять ігрові об’єкти (Game Objects). Їх можна використовувати для створення навколишнього середовища, персонажів, перешкод і декорацій та користувацького інтерфейсу [12].

У новоствореному проєкті в Unity, буде доступний зразок сцени, що містить основну камеру та направлене джерело світла (рис. 4.6).

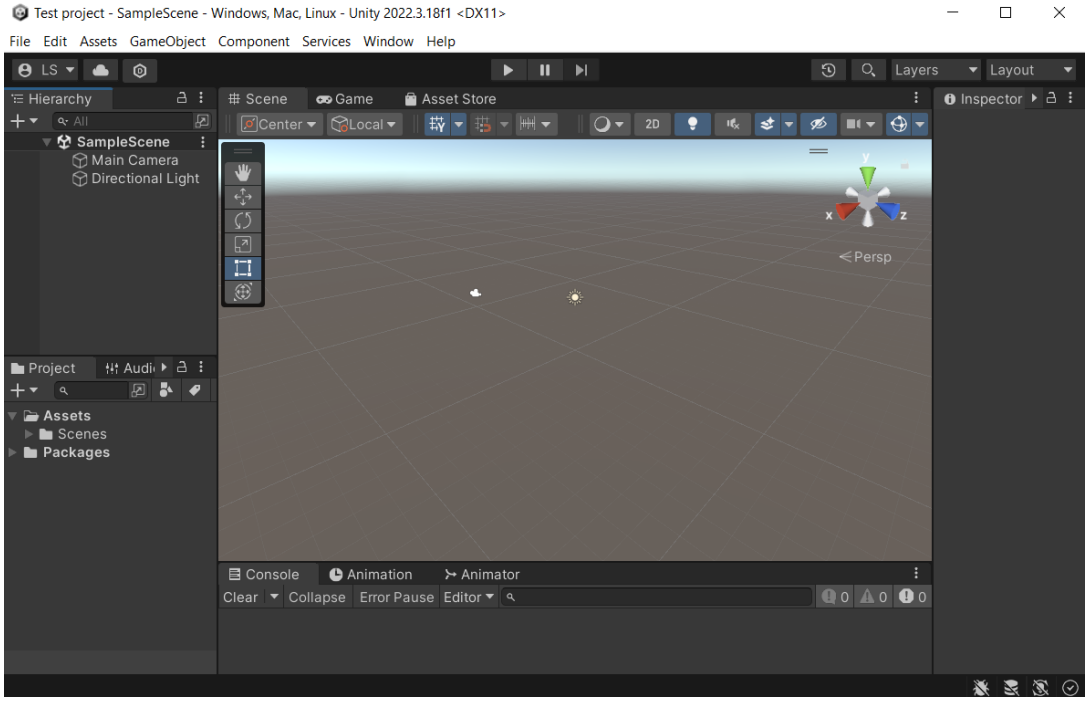

Рисунок 4.6 – Приклад сцени в Unity

У Unity існують різні способи створення нових сцен [12]:

– з діалогового вікна нової сцени (Файл (File) → Нова сцена (New Scene) або за допомогою клавіш Ctrl / Cmd + n);

– з меню (Ресурси (Assets) → Створити (Create) → Сцена (Scene));

– з вікна проєкту (всередині папки Scenes натиснути правою кнопкою миші та обрати Створити (Create) → Сцена (Scene));

– зі скрипту C# (за допомогою методу Instantiate).

Щоб уникнути плутанини між ігровими об’єктами та скриптами, слід змінити назву відповідно до їх призначення.

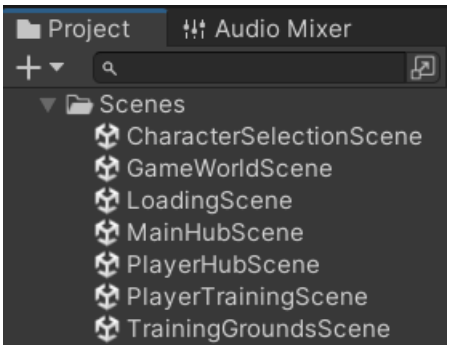

Рисунок 4.7 – Створені сцени в Unity

Усередині сцен будуть відбуватись основні дії зі створеними ігровими об’єктами (Game Objects). У вигляді ігрових об’єктів можна представляти як фізичні сутності, що спостерігаються у грі (персонаж, ґрунт, дерево, ландшафт, світло, зброя, куля, вибух тощо), так і метафізичні (менеджер спорядження, контролер багатокористувацького режиму тощо).

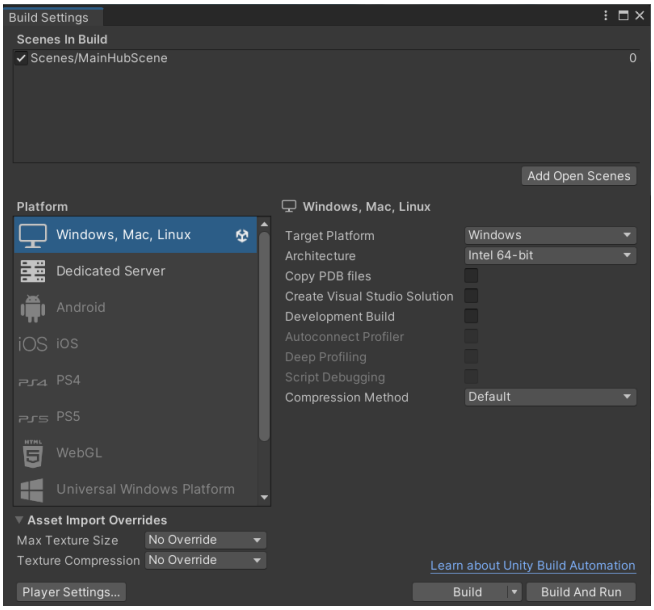

Рисунок 4.8 – Вікно налаштування збірки

Коли всі необхідні елементи було додано на полотно головного меню можна завантажувати сцену задля перевірки на помилки. Щоб завантажити сцену, її необхідно додати в налаштування збірки, інакше – повернеться помилка. Це можна зробити натиснувши на «Файл» → «Налаштування збірки», після чого відкриється вікно налаштування збірки (рис. 4.8), де треба обрати платформу та додати сцени, що слід завантажити.

4.1.2 Іконки та кнопки для меню та ігрових елементів

Для створення інтерфейсів, у тому числі і меню, у Unity використовуються UI-об’єкти. До них відносяться [12]:

– кнопки;
– зображення;
– списки;
– слайдери;
– чекбокси;
– списки та інші елементи.

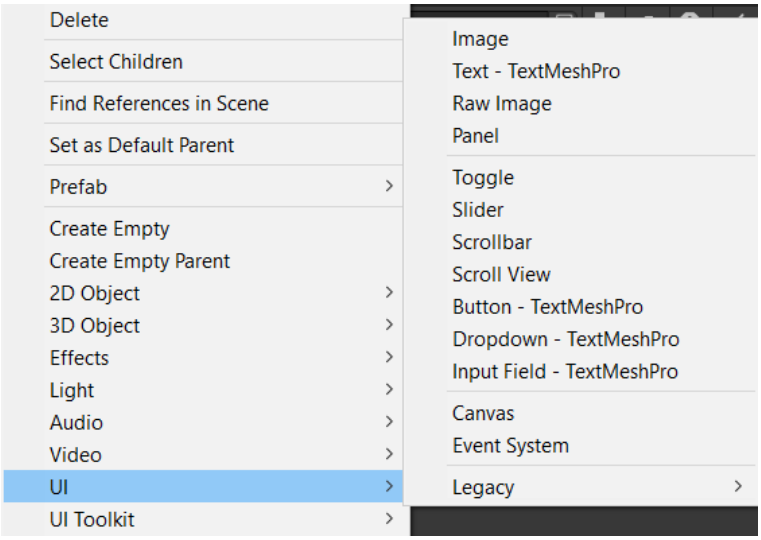

Рисунок 4.9 – UI-об’єкти в Unity

Відповідно до ігрових об’єктів, що перераховані в меню інтерфейсу користувача в Unity (рис. 4.9), є в основному чотири, які неодноразово використовуються для створення будь-якого типу інтерфейсу користувача. Це зображення, кнопки, текст і панелі. Більшість інших, таких як повзунки та поля введення, є в основному комбінаціями цих чотирьох із додатковими скриптами.

Щоб працювати з ними, потрібно створити об’єкт Canvas. Область полотна відображається у вигляді прямокутника у поданні сцени. Це дозволяє легко розміщувати елементи інтерфейсу користувача без необхідності постійно бачити уявлення гри.

Використання лише одного Canvas для всього інтерфейсу користувача можливо, але важливо враховувати наслідки. Unity оновлює та перемальовує Canvas щоразу, коли оновлюється один із його елементів (наприклад, підсвічування кнопки чи переміщення повзунка).

Для уникнення цього допоможе наявність кількох Canvas для різних обов'язків і оновлення стилів (наприклад, зберігаючи часто анімовані об'єкти в окремому Canvas від статичних об'єктів). Щоб захистити ієрархічну структуру, також можна вкладати полотна (Panels).

Панель (Panel) – це 2D-прямокутник для групування елементів інтерфейсу користувача. Можна змінити її колір або додати фонове зображення, натиснувши на панель в ієрархії, а потім у вікні інспектора в розділі «Image» обрати колір або фонове зображення.

Зображення – це неінтерактивний графічний елемент інтерфейсу користувача. Зображення широко використовуються іншими компонентами для візуального представлення. Unity також містить компонент Raw Image, який головним чином відрізняється від звичайних зображень тим, що приймає будь-які текстури, тоді як зображення приймають лише ресурси Sprite.

Всередині панелі головного меню буде додано наступні елементи: текст та кнопки. Щоб створити об'єкт «Текст» (TMP) треба натиснути правою кнопкою миші на фонову панель в ієрархії та вибрати UI → Text-TextMeshPro.

TextMeshPro (TMP) дозволяє додавати високоякісний текст інтерфейсу користувача, який можна легко стилізувати та редагувати. Для редагування створеного об'єкту «Текст» (TMP), а також інших ігрових об'єктів, слід натиснути на цей об'єкт в ієрархії та перейти до його компонентів у вікні інспектора.

Для створення кнопки потрібно натиснути правою кнопкою миші на фоновій панелі та обрати UI → Button-TextMeshPro.

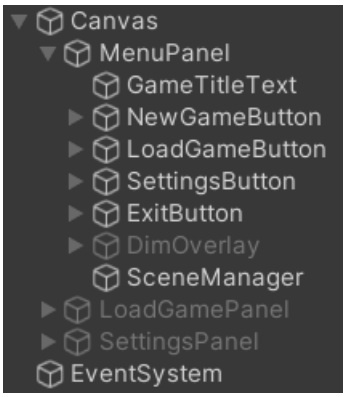

Рисунок 4.10 – Canvas з UI елементами

У Canvas на рис. 4.10 є три панелі: MenuPanel, LoadGamePanel і SettingsPanel. MenuPanel є головним меню ігрового застосунку, за допомогою якого гравець може розпочинати нову гру, завантажувати попередньо збережену гру, а також змінювати налаштування гри. Панелі LoadGamePanel і SettingsPanel відображаються після натиснення на кнопки LoadGameButton і SettingsButton відповідно.

4.1.3 3D-моделі об’єктів

3D-модель – це цифрове уявлення тривимірного об’єкта. У Blender 3D-модель складається з вершин, ребер та граней, які визначають її форму та структуру. Ці моделі можуть змінюватись від простих геометричних форм (наприклад, кубів та сфер) до дуже складних і деталізованих персонажів, навколишнього середовища та об’єктів [13].

Модифікатори дозволяють автоматично змінювати та перетворювати геометрію об’єктів, що значно спрощує та прискорює процес моделювання.

Перш за все, було знайдено якісні референси об’єктів та додано їх до Blender за допомогою Add → Image → Reference. Далі щоб відкрити меню додавання об’єктів потрібно натиснути Shift + A. Інструменти для маніпулювання об’єктами в 3D-просторі – переміщення (клавіша G), масштабування (клавіша S) та обертання (клавіша R) допомагають при створені будь-якого об’єкта.

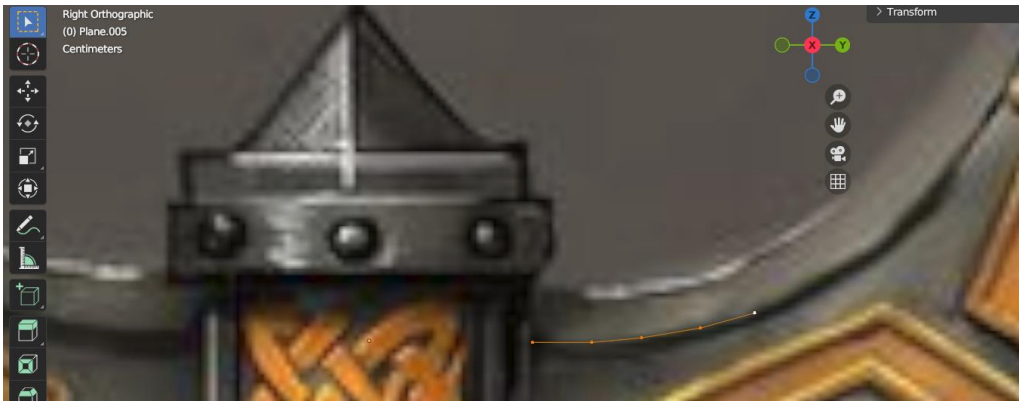

Рисунок 4.11 – Процес розробки 3D-моделі сокири

У режимі редагування (клавіша Tab) доданої фігури можна керувати її вершинами, гранями та ребрами. Вершини площини дозволяють легко обводити контур різних частин об’єктів на референсі (рис. 4.11). Щоб їх додати необхідно натиснути на потрібну та на клавішу E. Потім можна використати модифікатор Mirror для віддзеркалення створеного з іншої сторони або ж пророблюючи ті ж самі маніпуляції з додаванням вершин (залежить від обраного об’єкта).

Рисунок 4.12 – Збільшення об’єму створеного об’єкта 3D-моделі

Вершини, що було додано необхідно поєднати, обравши дві паралельно створені вершини та натиснувши клавішу F. Теж саме треба зробити обравши ребра. Після чого натиснувши Ctrl + r для створення нових розрізів у сітці об’єкта, треба натиснути клавішу S для збільшення об’єму створеного об’єкта (рис. 4.12).

Після створення леза сокири, слід додати топорище та інші необхідні частини. У випадку топорища буде використано циліндр поверх якого буде накладено площини з додаванням модифікаторів Subdivision, Shrinkwrap, Displace та Solidify для формування обмотки топорища.

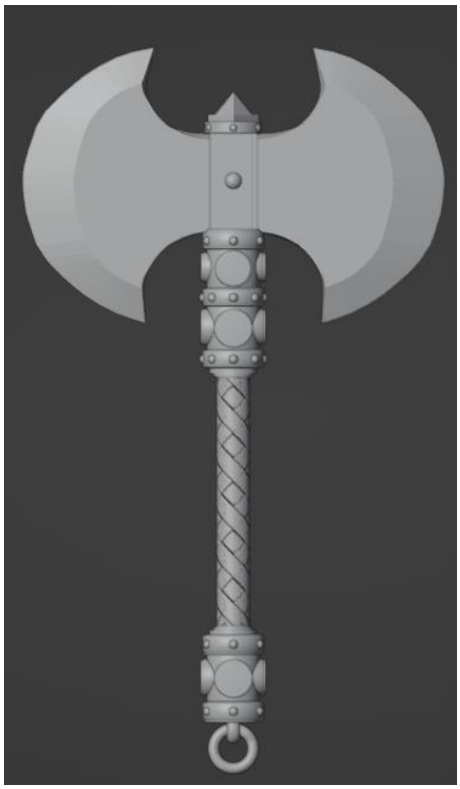

Рисунок 4.13 – 3D-модель сокири

Після виконання усіх вищеописаних кроків отримується 3D-модель сокири, яку тепер можна використовувати в якості зброї персонажа чи ворога (рис. 4.13).

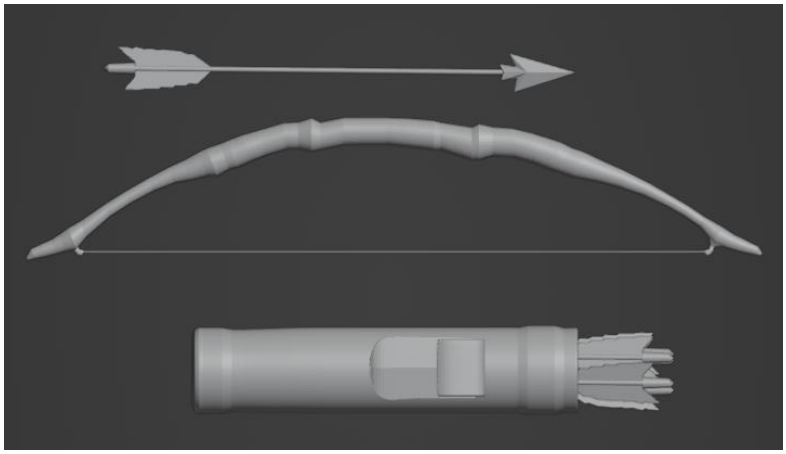

Рисунок 4.14 – 3D-модель зброї лучника

У процесі створення 3D моделей зброї лучника (рис. 4.14) в Blender спочатку було створено базову форму лука за допомогою циліндра. Потім, використовуючи інструменти Extrude, Rotation, Move та Scale, поступово модифікувався циліндр задля досягнення вигнутої форми, характерної для класичного лука. Стріли були створені аналогічним чином, починаючи з циліндра для древка. Для створення вістря використовувалась площина, яку за допомогою модифікатору Mirror було перетворено на симетричний та гострий наконечник. Це дозволило досягти точної та реалістичної форми стріли. Сагайдак був змодельований з такого ж циліндра, що і лук.

Незважаючи на те, що оточення також важливе для сприйняття гри – воно задає атмосферу і відчуття віртуального світу – саме герої залишаються головним фактором, що запам’ятовується. Візуальна привабливість персонажів, їх основні риси, звички, мотивація та моральні принципи – під час гри гравці свідомо чи ні ототожнюють себе зі своїми персонажами та на якийсь час стають з ними єдиним цілим.

Перш за все, необхідно знайти референси персонажа з різних боків. Далі у Blender створюється куб, який треба розмістити на референсі персонажа з переднього плану. Як і у випадку з сокирою та луком, слід використовувати режим редагування та клавішу E для точного відтворення форми персонажа. На цьому етапі слід не моделювати кисті, стопи та голову, оскільки для них буде застосовано окремий метод створення. Після завершення формування основної геометрії куба на передньому вигляді, переходимо до бічного вигляду персонажа, де коригуємо модель відповідно до референсів.

Починаючи із простих форм та поступово додаючи деталі, були використані функції Extrude, Sculpt , а також інші інструменти, такі як Bevel для створення згладжених країв, Loop Cut and Slide для додавання нових ліній і деталей, і Knife для створення точних розрізів. Для досягнення більшої плавності поверхонь у Blender можна використовувати подвійне натискання клавіші G для згладжування. Таким чином, обравши частину вершин, де необхідно згладити, можна досягти більш природного вигляду моделі (рис. 4.15).

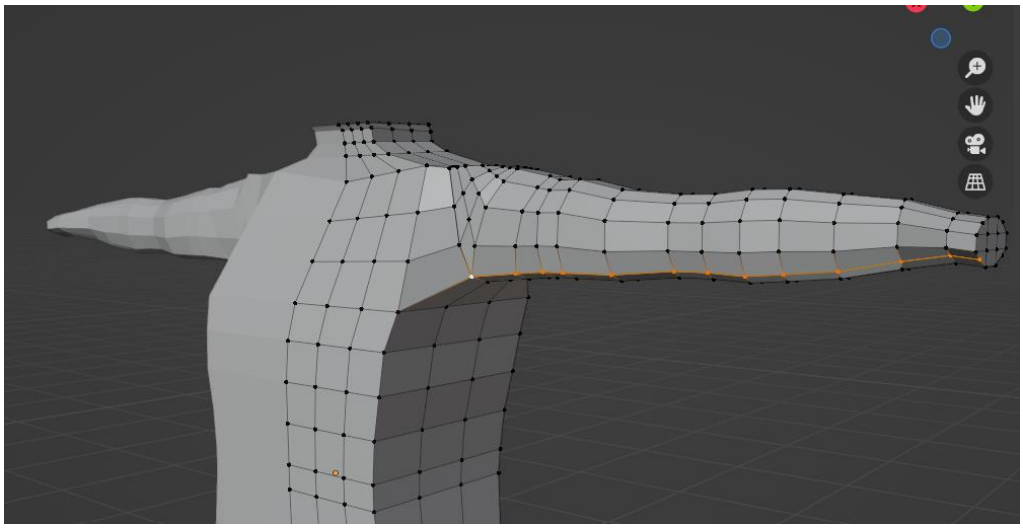

Рисунок 4.15 – Процес моделювання персонажа

Наступним етапом є створення кистей і стоп. Для кистей створюємо базову форму руки з використанням куба і поступово додаємо деталі пальців, застосовуючи інструменти Extrude і Rotate. Стопи створюються аналогічним чином, починаючи з простого куба і модифікуючи його для отримання більш деталізованої форми.

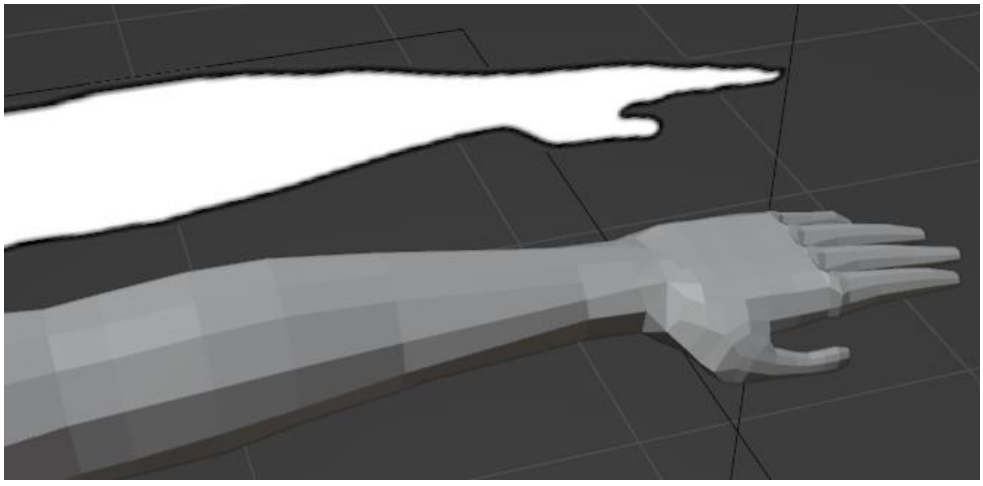

Рисунок 4.16 – Створення кінцівок персонажа

До ігрової сцени в Unity, де ландшафт використовується як основа, додаються ігрові елементи, такі як персонажі, об’єкти взаємодії тощо. Базова форма ландшафту може бути створена у Blender за допомогою площини, яка потім модифікується за допомогою інструментів, таких як Grab, Inflate, Smooth та інших.

Ці інструменти дозволяють створити реалістичні височини, пагорби, долини та інші елементи рельєфу [14].

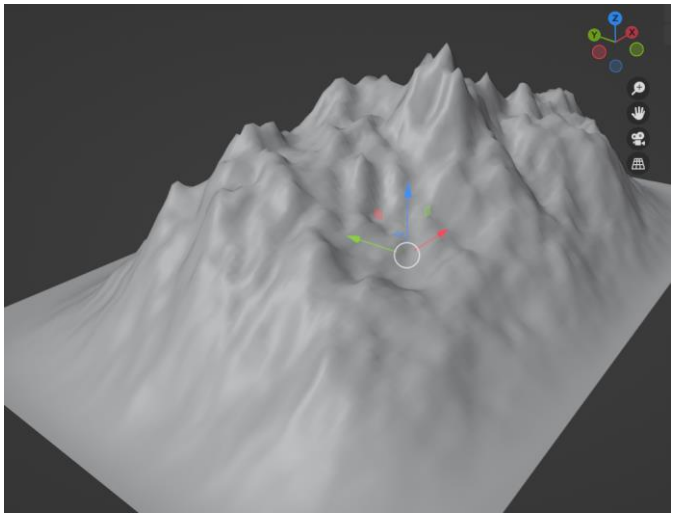

Рисунок 4.17 – Створення височин у Blender

Наступним кроком, після створення 3D-моделей об’єктів є їх експорт з Blender та імпорт до Unity. Для цього у Blender слід обрати File → Export → FBX (.fbx) та додати завантажений файл до папки Assets у Unity.

4.1.4 Текстури

Текстури у Blender відображають зовнішній вигляд об’єкта незалежно від його форми. У Blender матеріали складаються з текстур зображень. З технічного погляду текстура – це лише зображення чи візерунок, тоді як матеріал – це набір інформації, що визначає колір, блиск, прозорість та нерівності об’єкта. Один матеріал може містити кілька текстур, які змінюють зовнішній вигляд матеріалу в цілому, а об’єкт може мати декілька матеріалів, призначених для різних частин [15].

Основною перевагою використання текстур є можливість зробити простий об’єкт (наприклад, геометричну площину) схожим на складніший об’єкт (наприклад, дерев’яну підлогу).

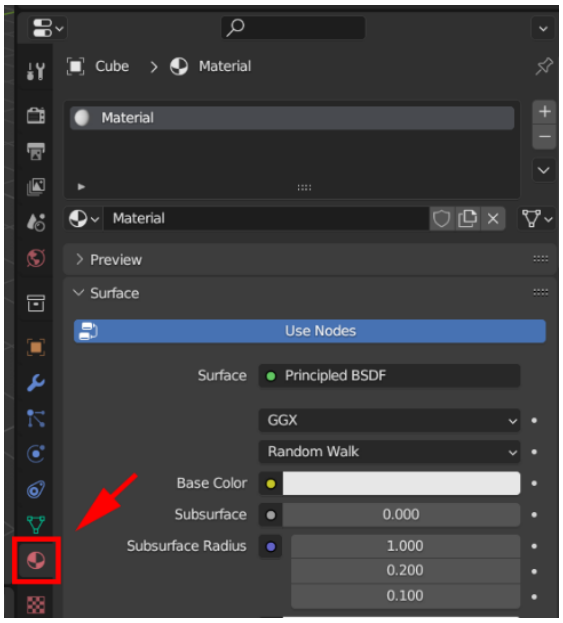

Рисунок 4.18 – Панель Material у редакторі властивостей

На панелі Material (рис. 4.18) можна додавати різні матеріали об’єкту, де за замовчанням додано звичайний матеріал з назвою Material. Для подальшого використання текстур зображень слід змінити назву матеріалу на відповідний та, якщо необхідно, додати один або декілька додаткових матеріалів.

Щоб підібрати необхідну текстуру можна або створити власну, або завантажити її з таких ресурсів, як BlenderKit [16], Poly Haven [17] чи Free PBR [18]. Після того як потрібна текстура буде створена або завантажена, її слід імпортувати в Blender для подальшого використання.

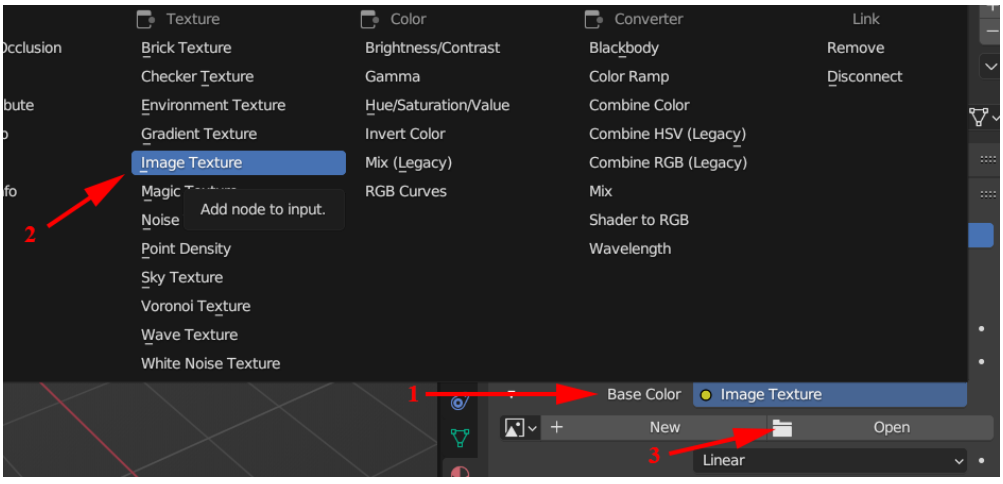

Рисунок 4.19 – Додавання текстури

На рис. 4.19 відображено процес додавання попередньо завантаженої текстури до моделі в Blender. Для цього необхідно відкрити вкладку Shader Editor (рис. 4.21), де можна налаштувати матеріали та застосувати текстури до моделі. Імпортовану текстуру можна перетягнути з вікна файлів прямо на вузлове дерево матеріалу або використовувати вузол Image Texture для її підключення. Це дозволить створити реалістичний вигляд моделі, надаючи їй додаткових деталей і глибини.

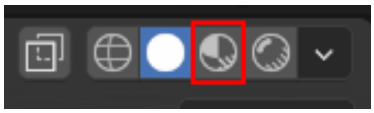

Рисунок 4.20 – Viewport Shading

У Viewport Shading можна обирати різні режими відображення моделі. Одним із них є режим для попереднього перегляду матеріалів та малювання текстур – Material Preview (рис. 4.20).

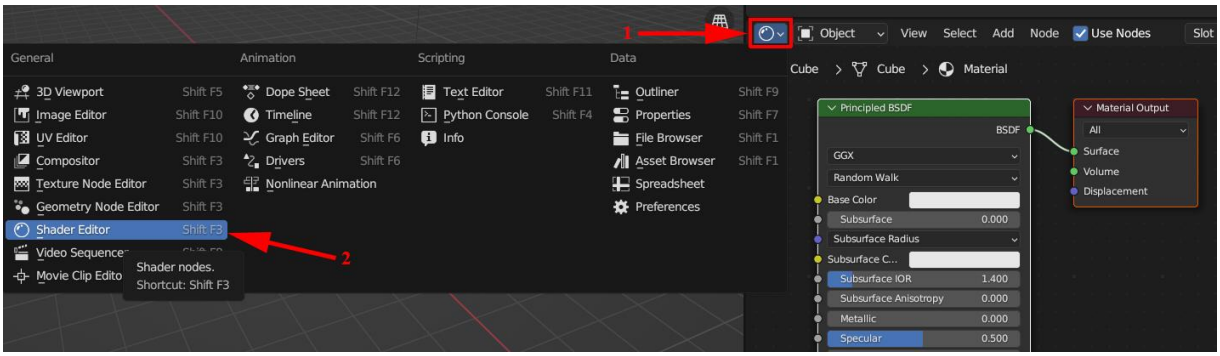

Рисунок 4.21 – Shader Editor

Shader Editor (рис. 4.21) у Blender відкриває широкі можливості для налаштування та створення матеріалів, надаючи користувачам повний контроль над виглядом своїх 3D моделей. Використовуючи вузлову систему, Shader Editor дозволяє комбінувати різні текстури, шейдери і математичні функції для досягнення бажаного результату.

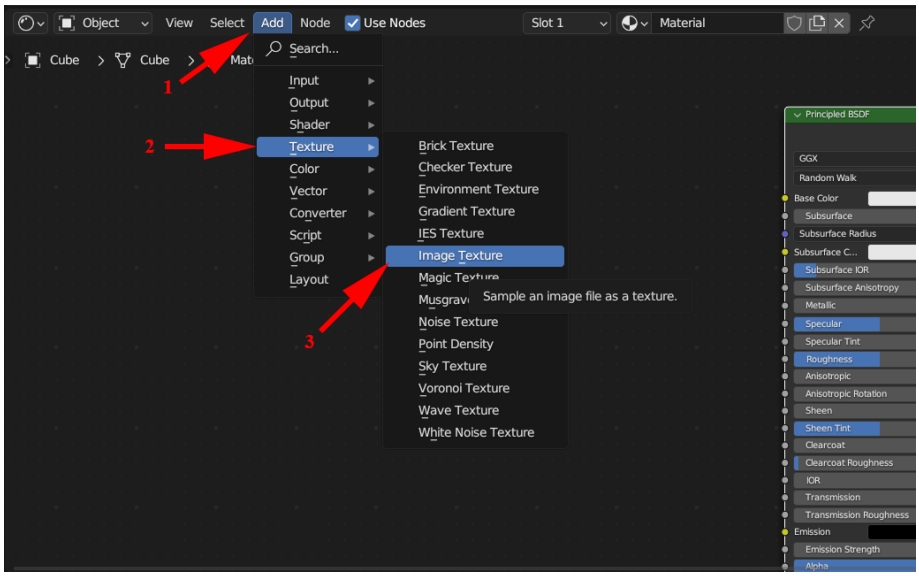

Рисунок 4.23 – Додавання текстури в Shader Editor

Додати текстуру в редакторі шейдерів можна натиснувши Add → Texture → Image Texture у шапці (рис. 4.22) або за допомогою клавіш Shaft + A. При додаванні нового матеріалу до об’єкта, Blender за замовчанням використовує для цього матеріалу шейдерний вузол Принциповий (Principled) BSDF.

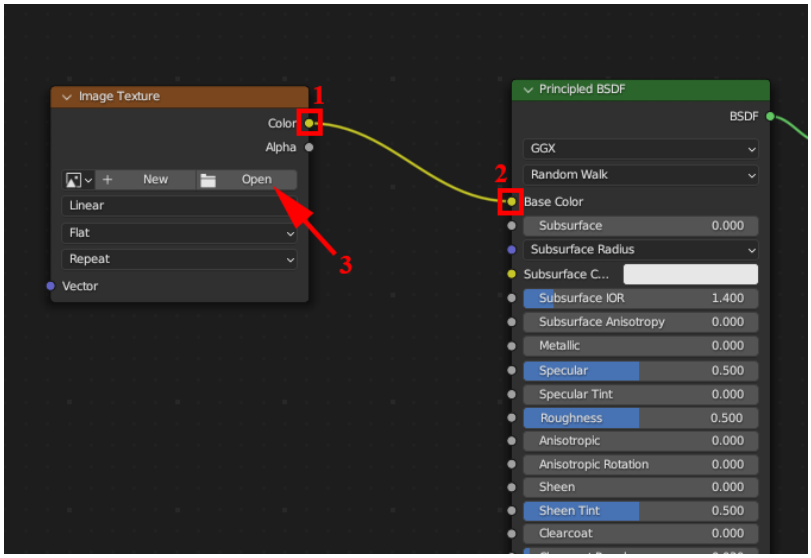

Рисунок 4.24 – Підключення матеріалу до Principles BSDF

Властивість Color блоку Image Texture треба з’єднати з властивістю Base Color блоку Principled BSDF, після чого можна додавати текстуру за допомогою Open (рис. 4.23).

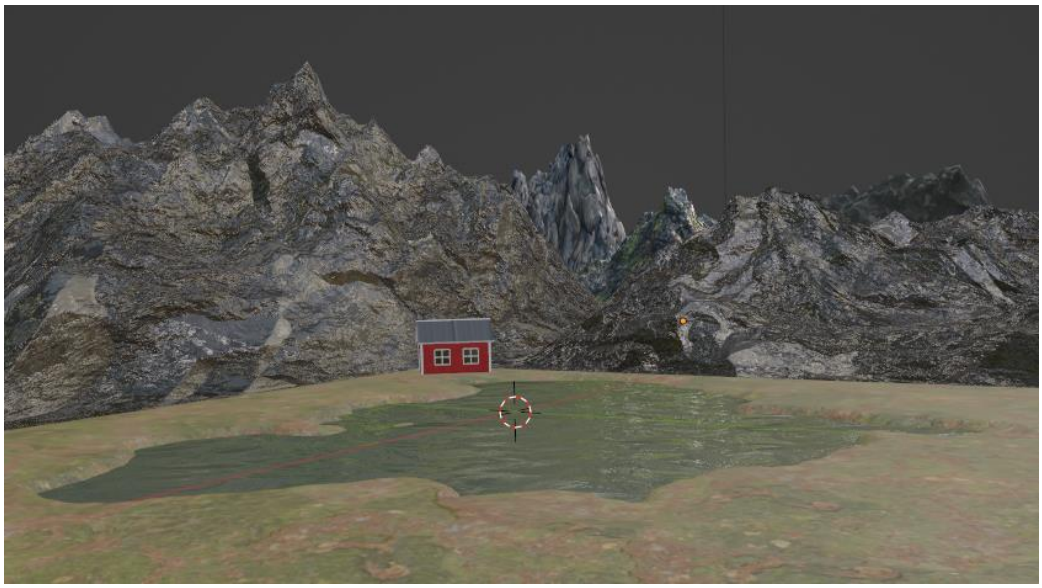

Рисунок 4.25 – Ландшафт ігрової сцени

Ландшафт ігрової сцени (рис. 4.25) відображає створені моделі з накладеними на них матеріалами, що включають невелике озеро, будинок та гори. Озеро відображає реалістичність водних поверхонь, віддзеркалюючи навколишній пейзаж. Будинок має деталізовану архітектуру, яка додає структуру сцені, а гори створюють відчуття глибини та простору.

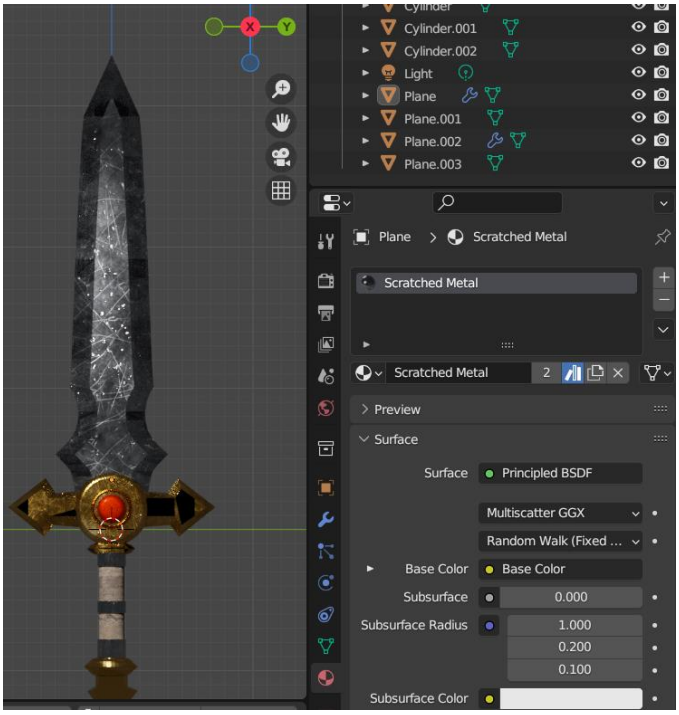

Рисунок 4.26 – Накладання матеріалів на меч

Процес накладання матеріалів на меч (рис. 4.26) додає йому деталізацію та реалістичний вигляд. Текстури включають зламки світла, тіні та відбиття, які підкреслюють форму та матеріали меча, роблячи його більш виразним у грі.

Рисунок 4.27 – Накладання матеріалів на кинджали

Кинджали (рис. 4.27) виготовлені з матеріалів, що включають деталізовані текстури, які підкреслюють їхню форму та функціональність у грі.

Рисунок 4.28 – Накладання матеріалів на щит

На щиті застосовано матеріал, що включає в себе дерев’яні зернини та металеві елементи, що додають йому структуру та захисний вигляд (рис. 4.28).

4.2 Написання скриптів

У Unity поведінка ігрових об’єктів контролюється компонентами, які приєднуються до них. Ці компоненти можуть бути вбудованими, наприклад, Transform для визначення позиції об’єкта у просторі, Sprite Renderer для відображення об’єкта та UI Text для виведення тексту в інтерфейсі. Також можна створювати власні компоненти за допомогою скриптів, які дозволяють обробляти ігрові події, змінювати параметри та реагувати на введення користувача.

Для створення скрипта в Unity можна використати меню Create у лівому верхньому куті панелі Project або вибрати Assets → Create → C# Script (або JavaScript / Boo скрипт) у головному меню. При створенні скрипта Unity автоматично генерує необхідні залежності та базові функції, такі як Start і Update. Start викликається один раз при створенні об’єкта із першим кадром, тоді як Update викликається кожен кадр. Скрипти взаємодіють із внутрішніми механізмами Unity через успадкування класу від MonoBehaviour [19].

Щоб керувати камерою, її положенням та обертанням додано наступний код:

```
public Transform target;
public Vector3 offset = new Vector3(0, 2, -5);
public float rotationSpeed = 1.0f;

void Update()
{
    transform.position = target.position + offset;

    transform.LookAt(target);

    float horizontal = Input.GetAxis("Mouse X") * rotationSpeed;
    target.Rotate(0, horizontal, 0);
}
```

Реалізація руху гравця:

```
public float speed = 6.0f;
public float jumpSpeed = 8.0f;
public float gravity = 20.0f;

private Vector3 moveDirection = Vector3.zero;
private CharacterController controller;
```

```
void Start()
{
    controller = GetComponent<CharacterController>();
}

void Update()
{
    if (controller.isGrounded)
    {
        float horizontal = Input.GetAxis("Horizontal");
        float vertical = Input.GetAxis("Vertical");

        transform.Rotate(0, horizontal * speed * Time.deltaTime, 0);

        moveDirection = transform.forward * vertical * speed;

        if (Input.GetButton("Jump"))
        {
            moveDirection.y = jumpSpeed;
        }
    }

    moveDirection.y -= gravity * Time.deltaTime;

    controller.Move(moveDirection * Time.deltaTime);
}
```

Логіка атаки ворога:

```
private void Attack()
{

    RaycastHit hit;
    if (Physics.Raycast(transform.position, transform.forward, out hit, attackRange))
    {
        PlayerHealth playerHealth = hit.collider.GetComponent<PlayerHealth>();
        if (playerHealth != null)
        {

            playerHealth.TakeDamage(attackDamage);
        }
    }
}
```

Метод для взаємодії гравця з NPC:

```
void InteractWithNPC()
{
    Collider[] npcs = Physics.OverlapSphere(transform.position, 3.0f, npcLayer);
    if (npcs.Length > 0)
    {
        NPCDialogue npcDialogue = npcs[0].GetComponent<NPCDialogue>();
        if (npcDialogue != null)
        {
            npcDialogue.StartDialogue();
        }
    }
}
```

Для збереження даних гри у форматі JSON потрібно спочатку створити структуру даних для зберігання інформації. Після цього можна використовувати бібліотеку серіалізації JSON, наприклад, JsonUtility у Unity, для перетворення цієї структури даних у JSON-рядок і запису його у файл [20].

Структура ігрових даних:

```
[System.Serializable]
public class GameData
{
    public string playerName;
    public int level;
    public int health;
    public SerializableVector3 position;
    public SerializableVector3 rotation;
    public Inventory inventory;
    public int coins;
    public int attack;
    public int armor;
    public string weapon;
    public int completedLevels;
    public int completedQuests;
}
```

Скрипт для керування збереження та завантаження даних:

```
private string savePath;

private void Awake()
{
    savePath = Application.persistentDataPath + "/save.json";
```

```
}

public void SaveGame(GameData data)
{
    string jsonData = JsonUtility.ToJson(data);
    File.WriteAllText(savePath, jsonData);
}

public GameData LoadGame()
{
    if (File.Exists(savePath))
    {
        string jsonData = File.ReadAllText(savePath);
        return JsonUtility.FromJson<GameData>(jsonData);
    }
    else
    {
        Debug.LogWarning("File not found.");
        return null;
    }
}
```

Після цього у PlayerController можна використовувати методи для збереження та завантаження даних гри.

Висновки до розділу 4

Застосовуючи ефективні стратегії, такі як забезпечення послідовності та ясності, оптимізація елементів управління, мінімалістичний дизайн, використання візуальної ієрархії та забезпечення контекстного зворотного зв’язку, було створено інтерфейс користувача, який є одночасно візуально привабливий та інтуїтивно зрозумілий. У ПЗ Blender було розроблено 3D-моделі об’єктів такі як: ландшафт, зброя, броня та персонажі. Також з його допомогою було додано матеріали та текстури для об’єктів. Всі створені моделі було імпортовано до Unity. Написано скрипти для взаємодії з ігровим світом та гравцем. А також виконано тестування ігрового застосунку, у результаті якого – помилок не було знайдено.

ВИСНОВКИ

У ході створення кваліфікаційної роботи бакалавра було розроблено ігровий застосунок у жанрі Action-RPG на основі рушія Unity. Проведено аналіз обраної предметної області та, на його основі, сформовано задачі та специфікації вимог до програмного забезпечення. Розроблено проєктні рішення, що забезпечують виконання специфікації вимог до програмного забезпечення. Цей процес включає в себе моделювання об’єкту та предмету дослідження, а також створення функціональних та інформаційних моделей програмного забезпечення. Описано виконану роботу з моделювання та конструювання програмного забезпечення. Обрано в якості мови програмування C#, рушій Unity, розроблено ігровий застосунок. Продемонстровано виконану роботу з кодування та тестування розробленого програмного забезпечення. При тестуванні помилок виявлено не було.

Виконано наступні **завдання**:

– опис та аналіз роботи Unity;

– дослідження та характеристика особливостей сучасних ігор у жанрі Action-RPG;

– розробка інтерфейсу та графічний дизайн;

– проєктування ігрового застосунку;

– реалізація ігрового застосунку на рушії Unity;

– перевірка якості роботи застосунку.

Слід зауважити, що дана програма є досить зручною та багатофункціональною для створення таких або схожих проєктів. При тестуванні програми помилок виявлено не було.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Game History. URL: https://www.computerhope.com/history/game.htm (date of access: 22.04.2024).

2. Video Game Genres: Everything You Need To Know. URL: https://www.hp.com/us-en/shop/tech-takes/video-game-genres (date of access: 22.04.2024).

3. Use Case Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/use-case-diagram (date of access: 23.03.2024).

4. Sequence Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/sequence-diagram (date of access: 24.03.2024).

5. Communication Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/communication-diagram (date of access: 25.03.2024).

6. Statechart Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/statechart-diagram (date of access: 25.03.2024).

7. Online Moqup. Wireframe & UI Prototyping Tool. Moqups. URL: https://moqups.com/ (date of access: 26.03.2024).

8. Class Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/class-diagram (date of access: 27.03.2024).

9. Component Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/component-diagram (date of access: 29.03.2024).

10. Package Diagram. URL: https://docs.staruml.io/working-with-uml-diagrams/package-diagram (date of access: 29.03.2024).

11. Eric Lengyel. Foundations of Game Engine Development. Volume 2: Rendering. – Terathon Software LLC, 2019. 412 p.

12. Unity – Manual: Unity User Manual 2022.3 (LTS). URL: https://docs.unity3d.com/Manual/index.html (date of access: 01.04.2024).

13. Rob Tarte. 3D Modeling in Blender – Tools, Tips and Tricks. – CRC Press, 2016. 482 p.

14. Arijan Belec. Blender 3D Incredible Models. First Edition. – Packt Publishing, 2022. 386 p.

15. Abdelilah Hamdani. 3D Environment Design with Blender: Enhance your modeling, texturing, and lighting skills to create realistic 3D scenes. – Packt Publishing, 2023. 344 p.

16. BlenderKit. URL: https://www.blenderkit.com/ (date of access: 05.04.2024).

17. Free PBR Materials. URL: https://freepbr.com/ (date of access: 05.04.2024).

18. Poly Haven. URL: https://polyhaven.com/ (date of access: 07.04.2024).

19. Paris Buttfield-Addison, Jonathon Manning, Tim Nugent. Unity Game Development Cookbook: Essentials for Every Game. First Edition. – O'Reilly Media, Inc., 2019. 405 p.

20. Kelvin Sung, Gregory Smith. Basic Math for Game Development with Unity 3D: A Beginner's Guide to Mathematical Foundations. Second Edition. – A K Peters/CRC Press, 2023. 575p.