МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри
_____Світлана ВАЩЕНКО
____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,
освітньо-професійної програми «Інформаційні технології проектування»
на тему: «Ігровий додаток жанру 2D RPG у середовищі Unity»

_____

Здобувача групи_ІТ-92/1___Груздо Дмитра Руслановича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро ГРУЗДО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник_____к.т.н. доц.___Віктор НЕНЯ_____ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра інформаційних технологій

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В.о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО

«____» _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Груздо Дмитру Руслановичу

1 Тема роботи Ігровий додаток жанру 2D RPG у середовищі Unity

_____

керівник роботи__Неня Віктор Григорович, к.т.н., доцент_____,

затверджені наказом по університету № 0588-VI від «29 »травня 2023 р.

2 Строк подання студентом роботи «_20_ »__червня__ 2023 р.

3 Вхідні дані до роботи технічне завдання на розробку web-додатку робіт фахівця з цифрового дизайну_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, які **потрібно розробити)** вступ, аналіз предметної області, аналіз додатків аналогів проектування додатку, розробка додатку_____

_____

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) вступ, актуальність, постановка задачі, аналіз додатків-аналогів, постановка задачі, моделювання та проектування, програмна реалізація, архітектура додатку, висновок _____

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 8 лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	17.04.2023 – 18.04.2023	
2	Оформлення технічного завдання	18.04.2023 – 18.04.2023	
3	Аналіз предметної області	19.04.2023 – 19.04.2023	
4	Проектування додатку	20.04.2023 – 21.04.2023	
5	Розробка додатку	24.04.2023 – 16.05.2023	
6	Тестування додатку	17.05.2023 – 17.05.2023	
7	Завантаження додатку на сайт	18.05.2023 – 18.05.2023	
8	Оформлення пояснювальної записки	01.06.2023 – 06.06.2023	

Студент _____ Груздо Д.Р.
(підпис)

Керівник роботи _____ к.т.н.,доц. Віктор НЕНЯ
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Ігровий додаток жанру 2D RPG у середовищі Unity».

Кваліфікаційна робота містить 73 сторінки, 11 таблиць, 36 рисунків, список літератури 23 найменування, 3 додатки.

Створений 2D ігровий додаток жанру RPG – “Chronicles of Eldia”, використовуючи середовище розробки Unity. Реалізована бойова система, система прогресії, можливість купувати та знаходити спорядження, створені моделі та анімації, інтерфейс. Оформлено візуальний стиль та аудіо супровід.

Перший розділ складається з аналізу предметної області та додатків аналогів. Представлено мету та постановку задачі проекту.

Другий розділ містить результати структурно-функціонального моделювання MVC, що складається з контекстної діаграми IDEF0, діаграми декомпозиції першого рівня та діаграма варіантів використання додатка Use Case.

Третій розділ містить у собі опис архітектури додатка та опис його практичної реалізації.

Результатом є готовий ігровий додаток, який розміщений на сайті Itch.io.

Ключові слова: ігровий додаток, Unity, C#, 2D, RPG.

ЗМІСТ

ВСТУП 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7
1.1 Огляд останніх досліджень і публікацій 8
1.2 Аналіз програмних продуктів – аналогів 9
1.3 Постановка задачі 13
2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ 14
2.1 Структурно-функціональне моделювання процесу розробки 14
2.2 Моделювання варіантів використання додатку 17
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ДОДАТКУ 19
3.1 Архітектура ігрового додатку 19
3.2 Реалізація додатку 20
ВИСНОВКИ 33
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 34
ДОДАТОК А 37
ДОДАТОК Б 48
ДОДАТОК В 60

ВСТУП

Історія відео-ігор починається у 1950-х роках, коли компанія “Національний інститут стандартів та технологій” (NIST) створила першу гру під назвою “OXO” або “Noughts and Crosses”. Гра була реалізована на електронних лампах та платах. Згодом, в 1962 році компанія “Spacewar!” створила гру з тією ж назвою для мейнфреймів [9].

У 1970-х роках компанія Atari випустила перші домашні відеоігри, такі як Pong та Space Invaders. Згодом з’явилися ігрові консолі, такі як Nintendo Entertainment System (NES) та Sega Genesis. У 1990-х роках відео-ігри стали більш складними та деталізованими, з’явилися 3D графіка, розширені можливості геймплею та мультиплеєрні режими [9].

У 2000-х роках з’явилися онлайн-ігри, які дозволяють гравцям з усього світу грати один з одним в режимі реального часу. Також в цей час з’явилися мобільні ігри, що дозволяють грати на мобільних пристроях. У 2010-х роках відео-ігри стали дещо більш соціальними, з додаванням можливості спілкування та співпраці гравців у грі.

Сьогодні відео-ігри є однією з найбільш популярних розважальних форм, з різноманітними жанрами та рівнем складності. Історія відео-ігор продовжується, і з кожним роком гри стають більш технологічними, реалістичними та соціальними.

Unity – це одне з найбільш популярних середовищ розробки ігор, яка забезпечує розробникам широкі можливості створення ігор будь-якого жанру та рівня складності. Середовище розробки Unity включає в себе візуальний редактор, мови програмування C# та JavaScript, інтегрований фізичний рушій та готові бібліотеки для розробки мобільних, веб- та настільних ігор [7].

Unity постійно розвивається та оновлюється, додаючи нові функції та можливості, що дозволяє розробникам створювати більш складні та реалістичні ігри з кращою графікою та фізикою. В середовищі Unity також є

можливість використовувати готові модулі та інструменти, які дозволяють швидко та ефективно розробляти різні елементи гри.

Окрім того, Unity має велику спільноту розробників, яка надає підтримку та допомогу в розвитку проектів, а також дозволяє обмінюватись досвідом та знаннями з іншими розробниками. В цілому, середовище розробки Unity є потужним та ефективним інструментом для створення ігор, що дозволяє розробникам реалізовувати свої ідеї та творчі концепції. Unity забезпечує широкі можливості для створення 2D ігор, що дозволяє розробникам створювати проекти різного рівня складності та різних жанрів, включаючи платформери, рольові ігри, ігри-шутери та інші. Тому це середовище було обране для реалізації ігрового додатку, в рамках виконання дипломного проекту [8].

Мета проекту - розробити ігровий додаток жанру 2D RPG.

Для того, щоб розробити додаток у середовищі Unity, потрібно виконати наступні задачі:

- визначити актуальність роботи, дослідити предметну область та провести аналіз аналогічних ігор;
- виконати функціональне моделювання та графічний опис процесу розробки, а також розробити діаграму варіантів використання ;
- розробити та реалізувати структуру та функціонал гри;
- виконати тестування гри.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Предметна область відеоігор є досить широкою та розмаїтою. Вона охоплює всі аспекти створення, розробки, тестування, реклами та продажу відеоігор.

Один з найважливіших аспектів створення відеоігор – це їх концептуалізація. Розробники повинні створювати ідеї для гри, які будуть цікавими та захоплюючими для гравців. Концептуалізація включає в себе – визначення жанру гри, розробку персонажів та історії, проектування ігрового світу та ігровий процес [17].

Після концептуалізації гри, розробники переходять до фази розробки, що включає в себе створення графіки, звуку та музики, програмування та тестування гри. Ці етапи можуть займати декілька років і вимагати значних витрат грошей та ресурсів [17].

Після завершення розробки гри, вона проходить фазу тестування та поліпшення. Тестування може проводитися як внутрішніми тестерами, так і залученням зовнішніх тестерів. Результати тестування допомагають розробникам виправляти помилки та удосконалювати гру [17].

Після успішного тестування, гра готова до реклами та продажу. Все починається з класичного рекламного брифу. У ньому детально описуються завдання, цільова аудиторія та основний посил, який необхідно донести. Якщо розробником стоїть маркетингова задача, цим повідомленням може бути заклик встановити гру, пограти або повернутися до неї. Для реклами своєї гри, розробники зазвичай використовують ігрові виставки, наприклад E3 (Electronic Entertainment Expo – Виставка електронних розваг), де велика кількість студій анонсують свої проекти, чи показують трейлери ігрового процесу, для привернення максимальної уваги до продукту.

1.1 Огляд останніх досліджень і публікацій

Останні дослідження та публікації про 2D RPG (Role-playing game – рольова комп'ютерна гра) ігри зосереджені на кількох основних аспектах, таких як ігровий процес, сюжет, візуальне та аудіо оформлення гри [10].

Одним з ключових досліджень є аналіз геймплею 2D RPG ігор, у якому досліджувалась взаємодія гравця з ігровим світом, включаючи розвиток персонажів, бойову систему, дослідження світу та завдання. Дослідження вказують на те, що ігровий процес може бути вдосконалений шляхом розвитку бойової системи та персонажів. Бойова система –це набір можливостей, які гра надає користувачу, для взаємодії з ворогами у бойових ситуаціях [19].

Щодо сюжету, були проведені дослідження, що зосереджуються на сюжетних наративах. Вони вказують на те, що сюжет має велике значення для гравців, оскільки він визначає їхні мотивації та інтереси в грі. Гарний сюжет повинен бути- емоційно зворушливим, цікавим та відповідати жанру гри.

Графіка також є важливим елементом ігор. Вона зазнає значного впливу від новітніх технологій, таких як підтримка високої розподільної здатності та анімація високої якості. Дослідження показують, що гарна графіка може підвищити інтерес гравців та зробити гру більш привабливою для них. Це підтверджують оцінки проектів на сайтах-агрегаторах, таких як – MetaCritic, GameStats, Rotten Tomatoes.

Сучасні розробники відеоігор використовують різноманітні технології для створення власних проектів [2]. Наприклад інтегровані середовища розробки, такі як Unity, Unreal Engine, Godot, являють собою основну технологію для створення ігор. Вони забезпечують розширені інструменти для створення графіки, програмування, фізики, анімації та інших аспектів гри.

Графічні двигуни: Графічні движки, такі як Unity, Unreal Engine, CryEngine, дозволяють створювати графічну складову ігор, включаючи

реалістичні 3D моделі, освітлення, частинки, ефекти, шейдери тощо. Вони надають інструменти для роботи з графічними ресурсами та їх візуалізацією [3].

Мови програмування - для розробки ігор широко використовуються мови програмування, такі як C#, C++, JavaScript/TypeScript, Python [4].

Фізичні двигуни, такі як к Box2D та PhysX, забезпечують симуляцію фізики у іграх. Вони дозволяють обробляти колізії, сили, рухи та інші аспекти, що додають реалізм до геймплею [3].

1.2 Аналіз програмних продуктів – аналогів

2D RPG – це жанр відеоігор, що поєднує в собі елементи рольових ігор та пригодницьких ігор. У цьому жанрі гравець керує персонажем, який відвідує різні локації. Локація – це віртуальне просторове середовище, в якому відбувається дія гри.), виконує завдання, збирає ресурси та підвищує рівень персонажа.

На сьогоднішній день на ринку є декілька програмних продуктів – аналогів 2D RPG, серед яких можна виділити такі:

Stardew Valley – це популярна 2D RPG, яка була розроблена у Канаді студією ConcernedApe. Додаток був написаний на мові C#, використовуючи Microsoft XNA Framework. Гравець керує фермером, який відвідує різні місця, збирає ресурси, вирощує рослини та тварин, і виконує завдання з метою розвитку свого господарства. Гра має приємну графіку та різноманітність завдань. На платформі Metacritic гра має – 87 балів від журналістів та 8.7 від гравців [1].

Рисунок 1.1 – Скріншот гри Stardew Valley

Undertale – ще одна 2D RPG, створена американським розробником Тобі Фоксом. Гравець керує людиною, яка потрапляє у підземне царство, де зустрічається з різними монстрами та є учасником різних конфліктів. Розробник приділяв більшу увагу сюжету та можливості гравців обирати власний шлях проходження гри. Користувач має можливість обрати один з 3 можливих розвитків сюжетної лінії:

- Нейтральний шлях – якщо гравець, знищує лише частину монстрів, а іншу частину не чіпає.
- Шлях пацифіста – розвивається, якщо гравець не знищує жодного противника.
- Шлях знищення – розвивається, якщо гравець знищує кожного противника на своєму шляху.

Оцінка на Metacritik – 92 бали від журналістів, 8.5 від гравців[1].

Рисунок 1.2 – Скріншот гри Undertale

Axiom Verge – це 2D RPG, що була розроблена американським програмістом Томом Хеппом. Гравець керує науковцем, який потрапляє на загадкову планету, де він зустрічається з різними небезпечними створіннями та збирає різні ресурси, щоб вивчити та розблокувати нові технології. Гра має унікальну графіку та захоплюючу історію. 80 балів від журналістів та 7.7 бали від користувачів [1].

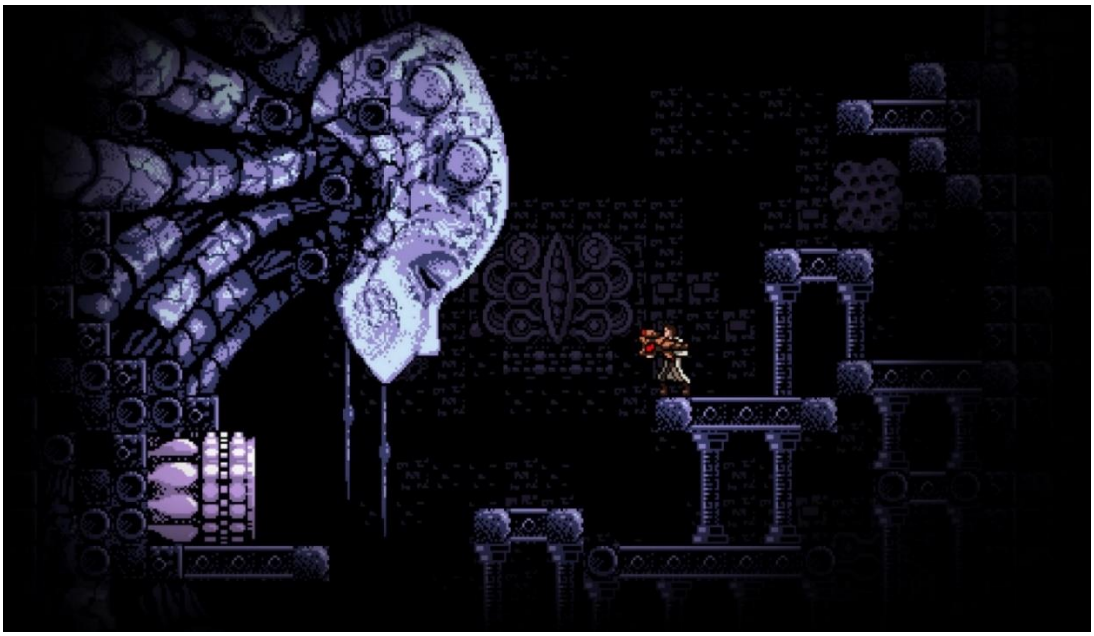

Рисунок 1.3 – Скріншот гри Axiom Verge

Таблиця 1.1 – Порівняння додатків-аналогів

Характеристика		Stardew Valey	Undertale	Chronicles of Eldia
Унікальний візуальний стиль		-	+	+
Унікальний саунд дизайн		+	+	+
Висока складність		-	+	-
Велика кількість контенту		+	-	+
Сюжет		+	+	+
Цікавий ігровий процес		+	+	+

1.3 Постановка задачі

Мета проекту – розробка ігрового додатку жанру 2D RPG. Для реалізації даного додатку було обрано середовище розробки Unity, дане програмне забезпечення має дуже великий набір інструментарію та є дуже зручним для розробників-початківців. При створенні скриптів буде використано мову C#. Користувач, після завершення та здачі проекту, отримує доступ для безкоштовного завантаження ігрового додатку “Chronicles of Eldia”. Основні вимоги для розробки:

- забезпечити достатню кількість інформації для ознайомлення з продуктом;
- дослідити предметну область та цільову аудиторію;
- проаналізувати продукти-аналоги;
- обрати інструменти реалізації;
- створити прототип додатку;
- продумати структуру додатку;
- розробити функціонал додатку;
- виконати тестування додатку.

Для досягнення поставленої мети необхідно виконати такі задачі:

- проектування концепції гри;
- створення ассетів;
- розробка геометрії рівнів;
- програмування та скриптування;
- реалізація функціональності.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Структурно-функціональне моделювання процесу розробки

Моделювання процесу розробки ігрового додатку може бути здійснено за допомогою структурно-функціонального підходу. На данному етапі будуть описуватись методології які дозволять спланувати створення проекту, визначити послідовність робіт та розподілити час на кажен етап, що сприяє ефективному використанню ресурсів і зменшенню часу, потрібного для завершення проекту.

IDEF0 (Integration Definition for Function Modeling) - це методологія функціонального моделювання, яка дозволяє аналізувати та описувати функції системи, процеси, взаємозв'язки та інші аспекти комплексних проектів [13].

IDEF0 використовує графічні символи для створення функціональних моделей, які можуть бути використані для аналізу, проектування та управління процесами. Основна ідея IDEF0 полягає в поданні системи як набору функцій, що виконуються за певними правилами та взаємозв'язками.

Основні елементи IDEF0 включають [13]:

- Функції: Представлені у вигляді боксів та описують дії, які виконуються системою або її компонентами.
- Фактори керування: Показують вхідні та вихідні параметри для функцій та визначають умови, за яких функції виконуються.
- Об'єкти Представлені у вигляді стрілок та показують потоки даних, матеріалів або інформації між функціями.
- Ресурси Вказують ресурси, необхідні для виконання функцій, такі як обладнання, персонал, програмне забезпечення тощо.

- Керування: Показують механізми або правила, за якими відбувається взаємодія між функціями та об'єктами.

Застосування IDEF0 дозволяє визначити ієрархічну структуру функцій, встановити послідовність виконання завдань, ідентифікувати взаємозв'язки між елементами системи та забезпечити зрозуміле представлення процесів для аналізу та вдосконалення. IDEF0 може бути корисним інструментом при проектуванні розробки ігрових додатків для розуміння та візуалізації функціональності системи.

Концептуальна діаграма бізнес-процесу розробки додатку наведена на рисунку 2.1.

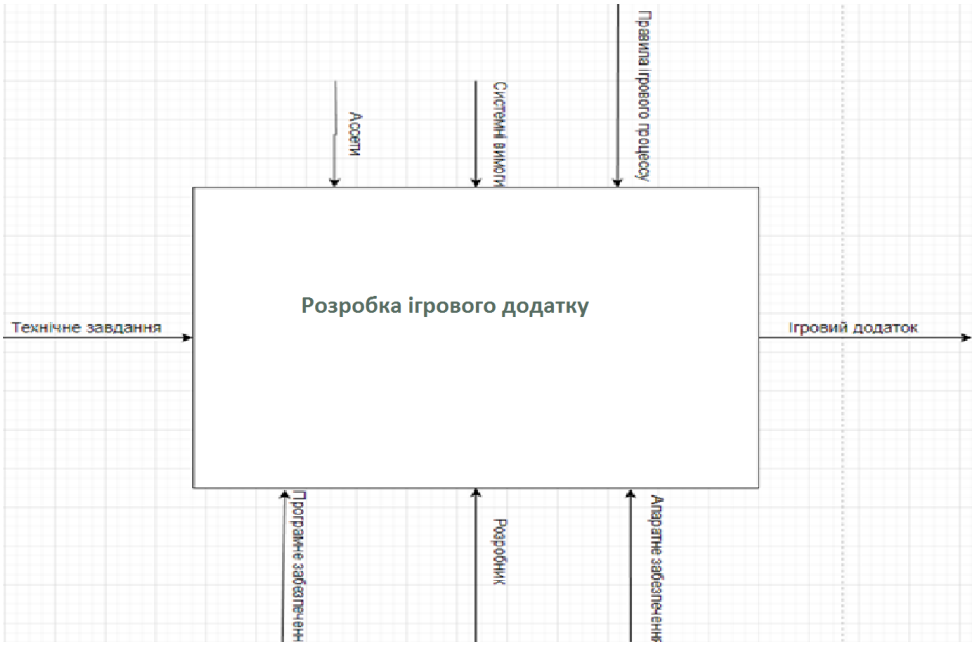

Рисунок 2.1 – Контекстна діаграма IDEF0

Після створення діаграми декомпозиції, виділено такі підпроцеси: створення структури додатку, розробка моделей та анімацій, реалізація ігрових механік, створення меню та інтерфейсу та тестування.

Таблиця 2.1 – Дані для діаграми декомпозиції

Підпроцес	Вхід	Управління	Механізм	Вихід
Створення структури додатка	Технічне завдання	ассети, середовище, системні вимоги, правила ігрового процесу	Розробник, програмне забезпечення, апаратне забезпечення	структура
Розробка моделей та анімацій	структура	ассети, середовище, системні вимоги, правила ігрового процесу	Розробник, програмне забезпечення, апаратне забезпечення	Програма з моделями та анімаціями
Реалізація ігрових механік	Программа з моделями та анімаціями	ассети, середовище, системні вимоги, правила ігрового процесу	Розробник, програмне забезпечення, апаратне забезпечення	Програма з ігровими функціями
Створення меню та інтерфейсу	Программа з ігровими функціями	ассети, середовище, системні вимоги, правила ігрового процесу	Розробник, програмне забезпечення, апаратне забезпечення	Демо-версія додатку
Тестування	Демо-версія додатку	ассети, середовище, системні вимоги, правила ігрового процесу	Розробник, програмне забезпечення, апаратне забезпечення	Ігровий додаток

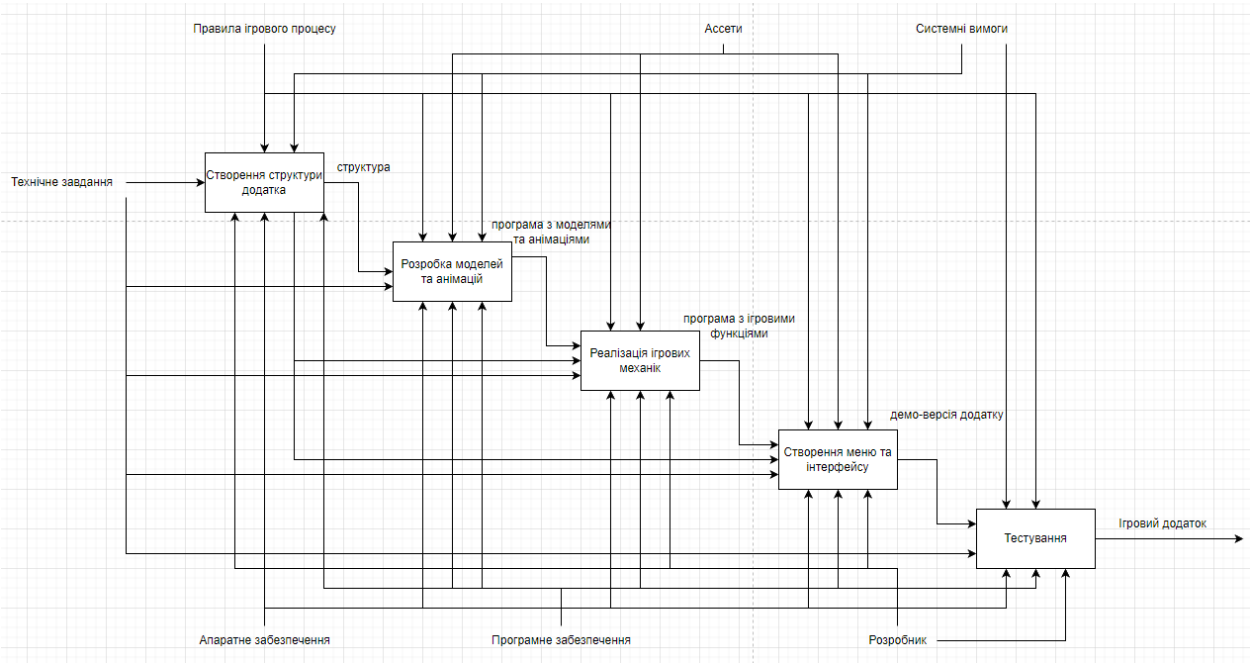

Рисунок 2.2 – Діаграма першого рівня декомпозиції IDEF0

2.2 Моделювання варіантів використання додатку

Моделювання варіантів використання допомагає ідентифікувати та описати різні сценарії використання, які користувачі можуть здійснювати під час взаємодії з додатком[13]. Це дає розробникам можливість краще розуміти потреби та очікування користувачів, а також спрямовує процес розробки на досягнення конкретних цілей.

Для моделювання варіантів використання ігрового додатку ‘Chronicles of Eldia” буде використана діаграма Use Case. Кожен варіант використання (use case) описує конкретний сценарій, який описує послідовність дій користувача та системи.

Таблиця 2.2 – Варіанти використання

№	Назва	Опис
1	Початок нової гри	Користувач може почати нову ігрову сесію
2	Продовження гри	Користувач може продовжити попередню ігрову сесію після виходу з неї
3	Вихід з гри	Користувач має можливість вийти з додатку
4	Відображення меню	Дає можливість призупинити гру та отримати доступ до певних функцій
5	Доступ до інветаря	Гравець може використовувати ігрові предмети
6	Закриття меню	Можливість відновити гру після паузи
7	Зберігання прогресу	Дає можливість гравцю зберегти прогрес своєї ігрової сесії
8	Вихід до головного меню	Можливість перейти з меню паузи до головного меню гри

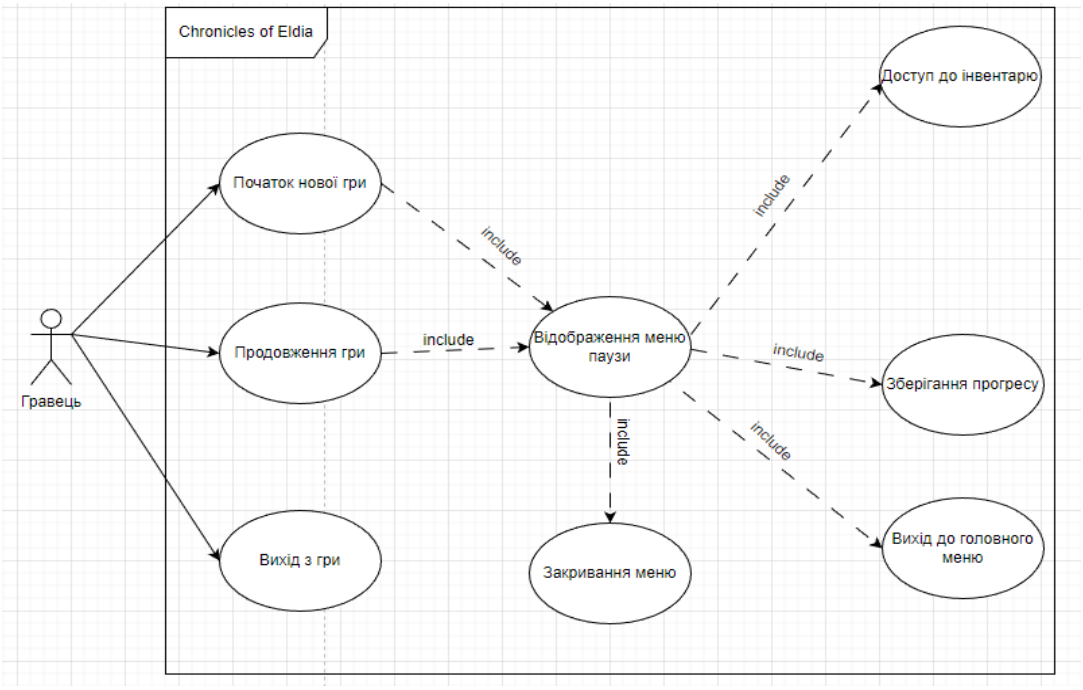

Рисунок 2.3 – Діаграма варіантів використання додатку

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ДОДАТКУ

3.1 Архітектура ігрового додатку

Архітектура ігрового додатку включає організацію його компонентів, модулів та взаємозв'язків між ними. Основна мета архітектури полягає у створенні структури, яка забезпечує ефективну розробку, підтримку та розширення додатку. Однією з поширених архітектур для ігрових додатків є Model-View-Controller (MVC), яка зображена на рисунку 3.1 [13].

Модель (Model) − включає в себе дані та логіку гри. Вона відповідає за зберігання і обробку ігрових об'єктів, станів, правил та логіки взаємодії з ними. Модель може містити компоненти для управління фізикою, штучним інтелектом, управліннями керуванням гравця та іншими логічними елементами гри.

Вид (View) − відповідає за візуалізацію гри та інтерфейсу користувача. Він включає графічні компоненти, анімацію, звукові ефекти та інші елементи, які створюють візуальний досвід гравця. Вид також може включати інтерфейс користувача для взаємодії з грою, наприклад, кнопки, показники стану гравця тощо.

Контролер (Controller) − відповідає за керування взаємодією між моделлю та видом. Він обробляє вхідні події, такі як натискання клавіш, керуваннями гравця та інші дії, і взаємодіє з моделлю та видом відповідно до цих подій. Контролер також забезпечує синхронізацію та оновлення стану моделі та виду.

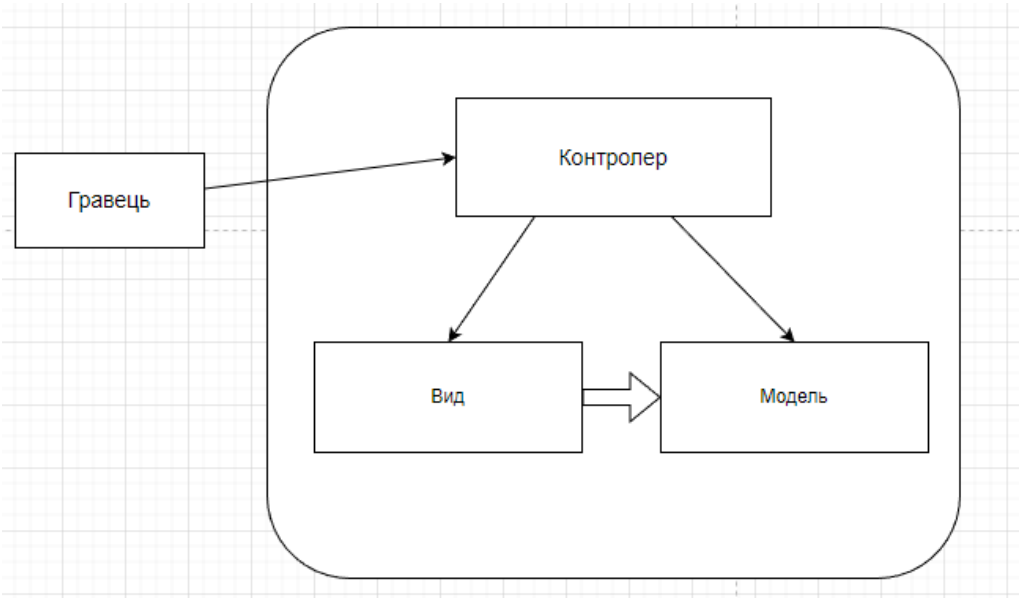

Рисунок 3.1 – Схема архітектури MVC [13]

3.2 Реалізація додатку

Для прграмної реалізації ігрового додатку використовувалось середовище розробки Unity та мова програмування C#, яка є базовою мовою створення скриптів у даному середовищі.

Перший етап розробки це налаштування проекту, рисунок 3.2, в Unity та завантаження потрібних матеріалів (ассетів), рисунок 3.3, для створення зовнішнього вигляду додатка. Unity має власну бібліотеку ассетів для створення проектів, у якій користувачі можуть ділитись власними матеріалами безкоштовно або за кошти.

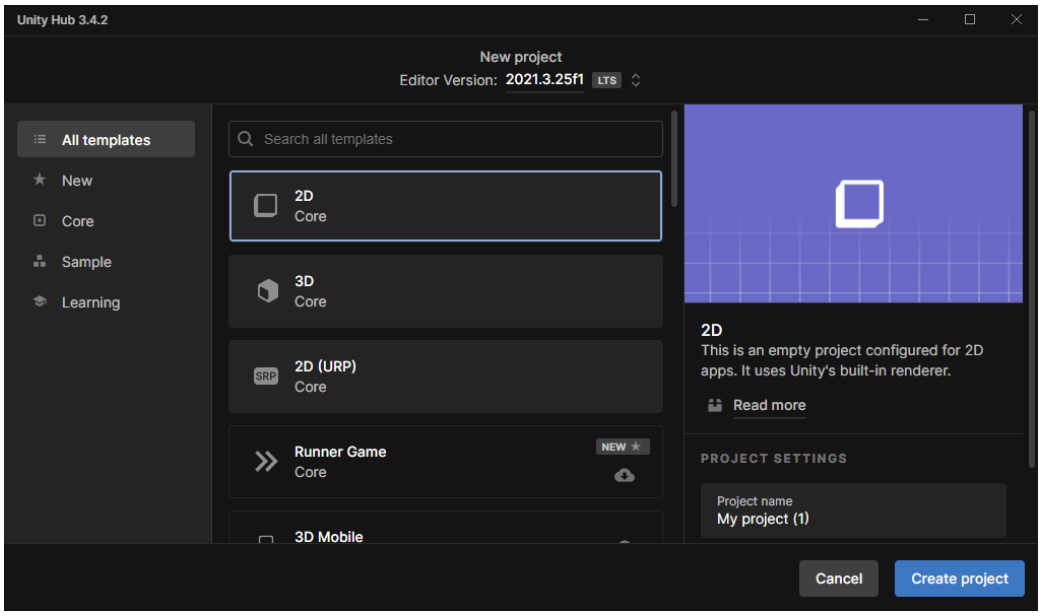

Рисунок 3.2 – Налаштування проекту

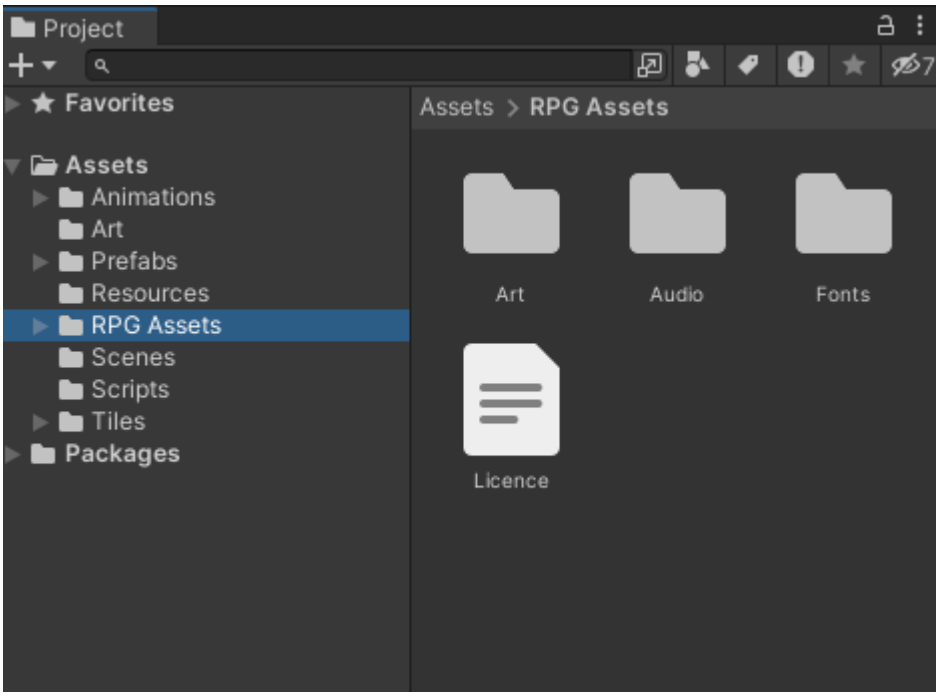

Рисунок 3.3 – Завантажений архів ассетів

Після первинного налаштування, проект готовий до подальшої розробки. За допомогою завантажених зображень створюються ігрові області, які поділені на декілька сцен, між якими буде пересуватись персонаж, що зображено на рисунку 3.4.

Рисунок 3.4 – Приклад ігрової локації

Рисунок 3.5 – Матеріали для створення локації

Гравець керує персонажем, який має унікальну модель на анімації пересування, рисунок 3.6. Зображення для моделі було взято з архіву ассетів. За допомогою вікна Animator ми створюємо анімації переміщення моделі головного героя, рисунок 3.7.

Рисунок 3.6 – Модель персонажа

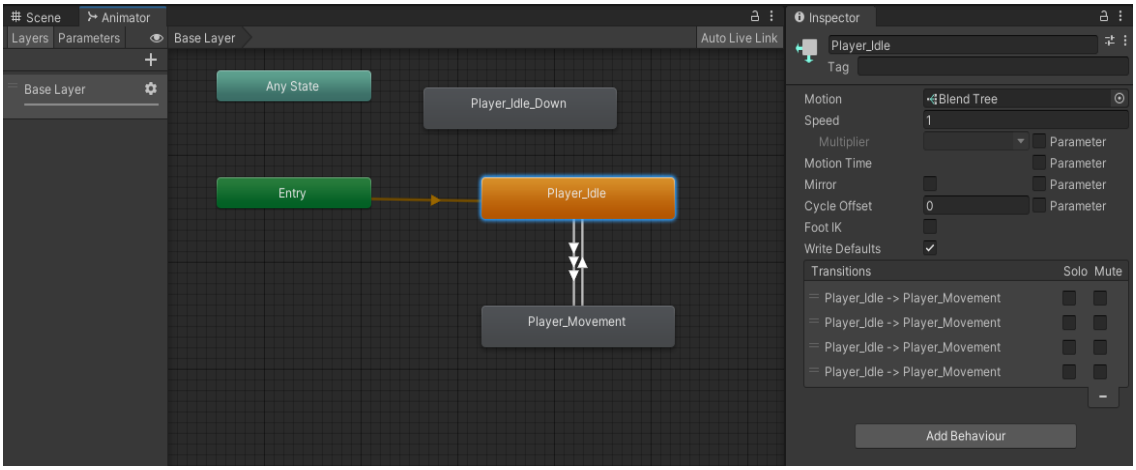

Рисунок 3.7 – Вікно створення анімацій

Контроль персонажа реалізований за допомогою скрипта Player Controller, рисунок 3.8. Кнопки керування персонажем задані у налаштуваннях проекту у вікні Input Manager, рисунок 3.9. Щоб персонаж не виходив за межі ігрової області, обмежуємо його пересування за допомогою функції у файлі скрипта Player Controller.

Рисунок 3.8 – Скрипт Player Controller

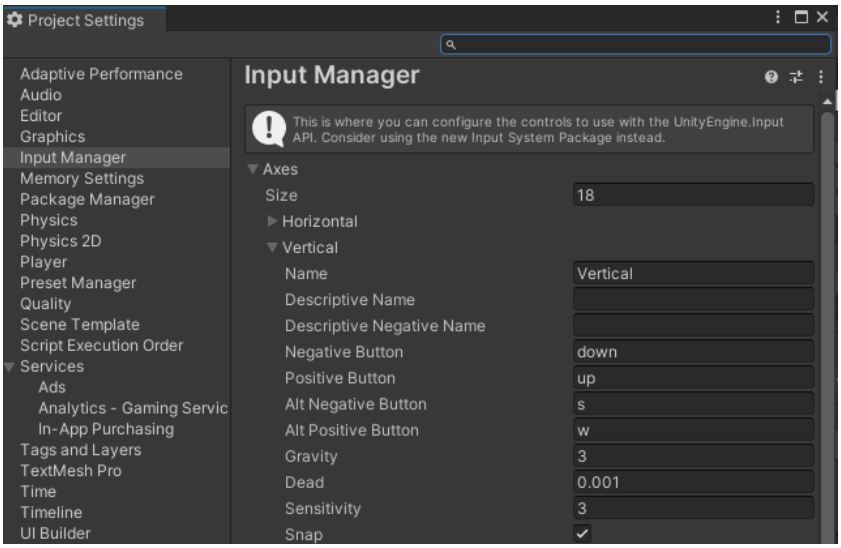

Рисунок 3.9 – Кнопки контролера

Для того, щоб камера слідкувала за персонажем та не виходила за межі ігрової області, був створений ігровий об’єкт Custom Camera з відповідними налаштуваннями та під’єднаний скрипт Camera Controller, рисунок 3.10.

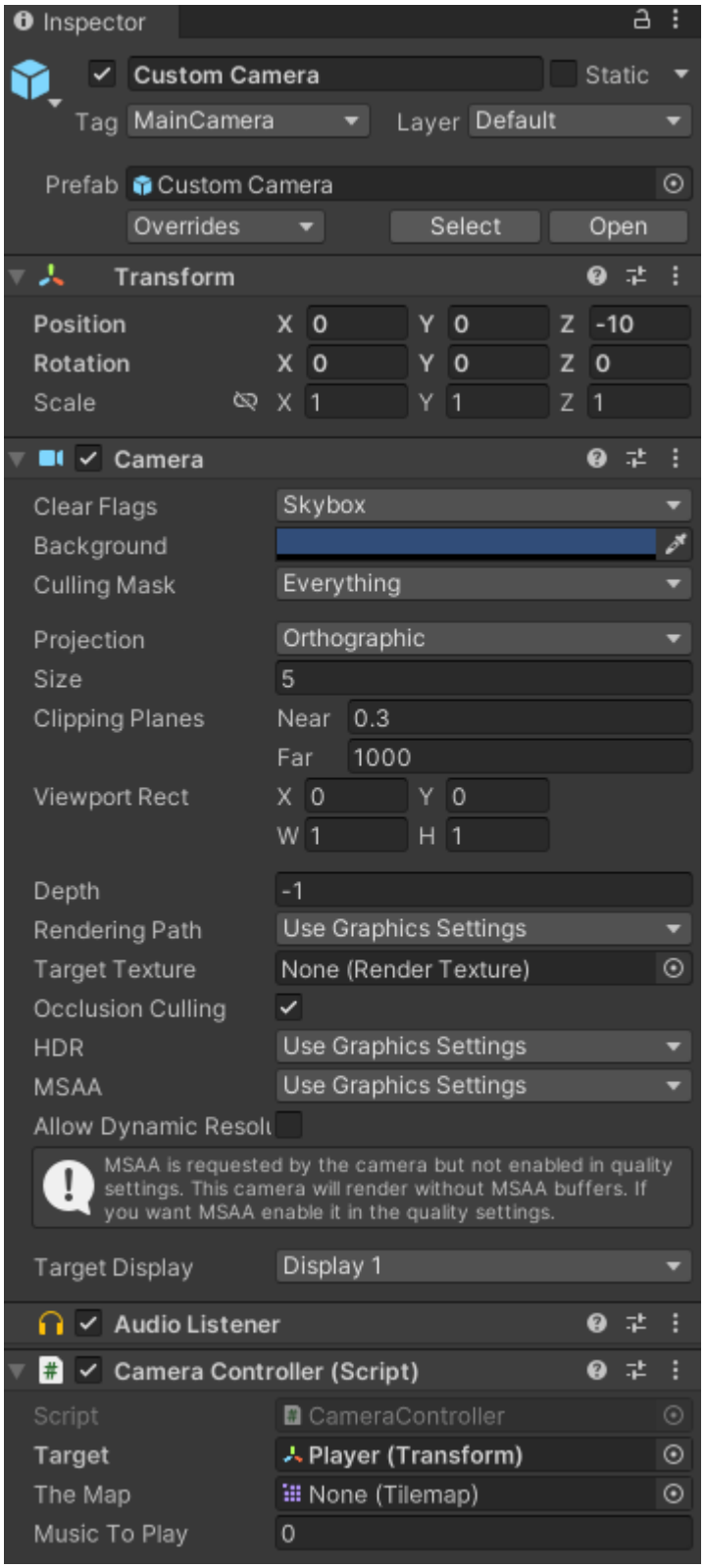

Рисунок 3.10 – Налаштування камери

Для того щоб персонаж мав можливість переміщатись з однієї локації до іншої, створено декілька ігрових об’єктів Area Exit(місце виходу з локації), рисунок 3.11, та Area Entrance (місце переходу до локації), рисунок 3.12. При

взаємодії персонажа з цими об’єктами виконується відповідний скрипт, та персонаж переміщується в іншу сцену.

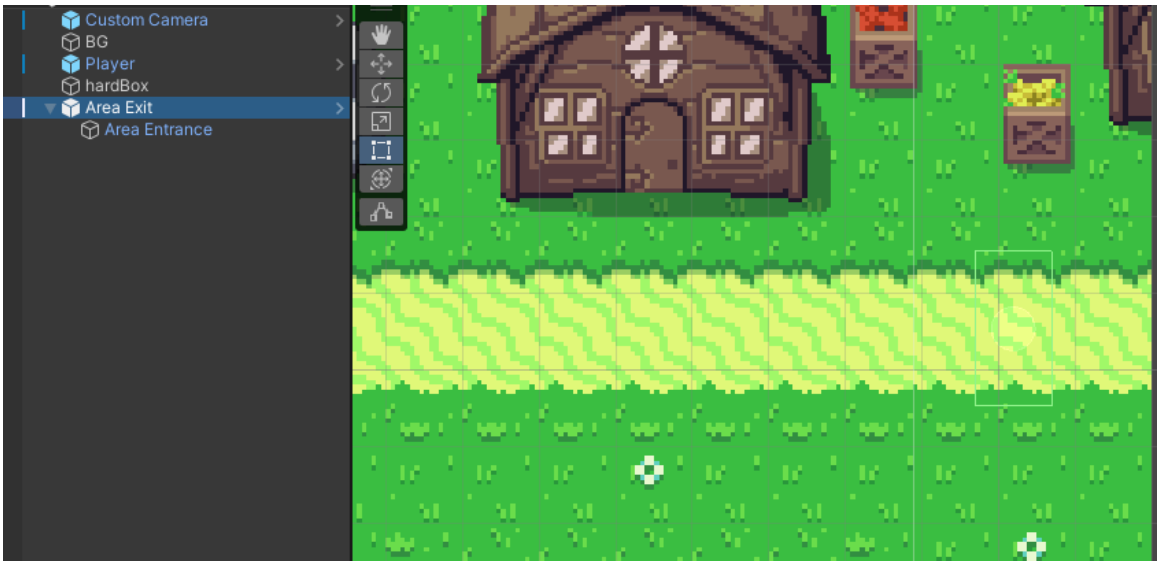

Рисунок 3.11 – Area Exit та Area Entrance

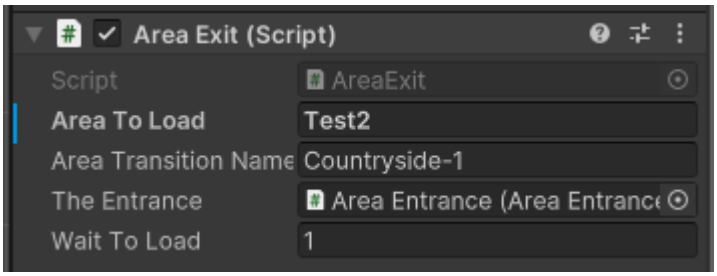

Рисунок 3.12 – Скрипт Area Exit

Користувач має можливість взаємодіяти з іншими персонажами, якими керує комп’ютер. Одні персонажі можуть вести з головним героєм діалог та давати завдання, інші персонажі це вороги з якими гравець вступає в бій. Діалогова система реалізована файлами скриптів DialogActivator, рисунок 3.13, та DialogManager, рисунок 3.14. Бойова система реалізована скриптами BattleManager, BattleMove, BattleStart, BattleNotification, BattleReward, BattleStart, BattleTarget, BattleType. Бойова сцена відображена на рисунку 3.16.

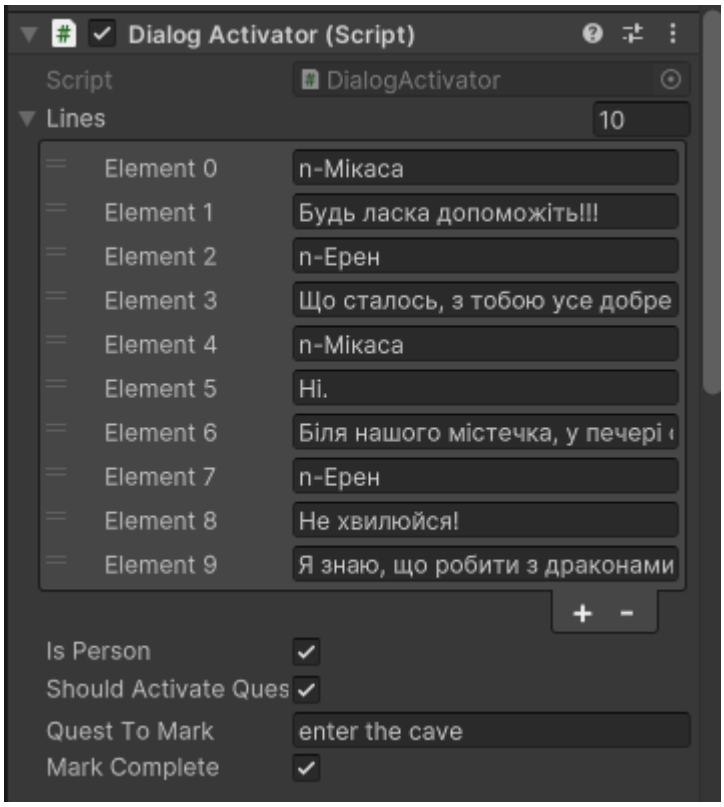

Рисунок 3.13 – Скрипт активації діалогової системи

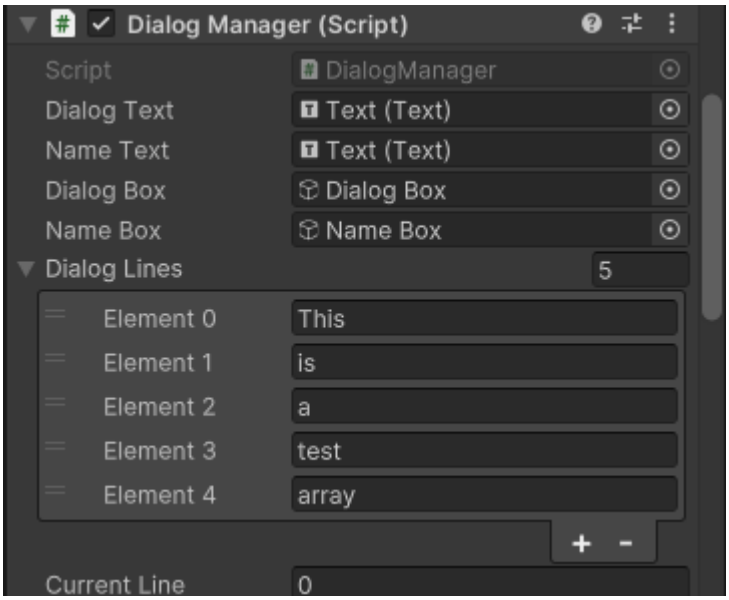

Рисунок 3.14 – Скрипт керування діалоговою системою

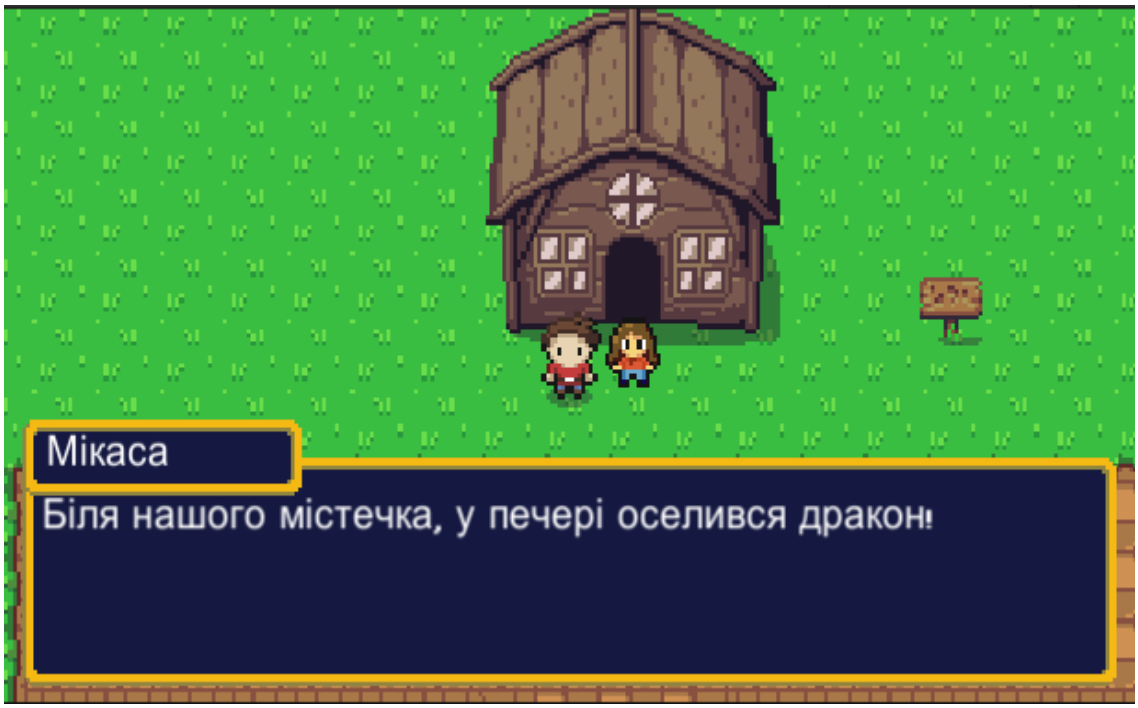

Рисунок 3.15 – Вигляд вікна діалогу

Рисунок 3.16 – Скрипт бойової системи

Рисунок 3.17 – Вигляд бойової системи

Користувач має можливість поставити гру на паузу, рисунок 3.18. В меню паузи є декілька функцій:

- Інвентар – доступ до предметів, які зібрав або купив гравець;
- Характеристики – характеристики персонажів;
- Зберегти – зберегти прогрес гри;
- Закрити – закрити меню та продовжити гру;
- Вийти – перейти до головного меню.

Для цього був створений скрипт GameMenu, рисунок 3.19, який буде керувати логікою функціонування меню паузи. Також об’єкти Button, за допомогою яких створені кнопки для функцій меню.

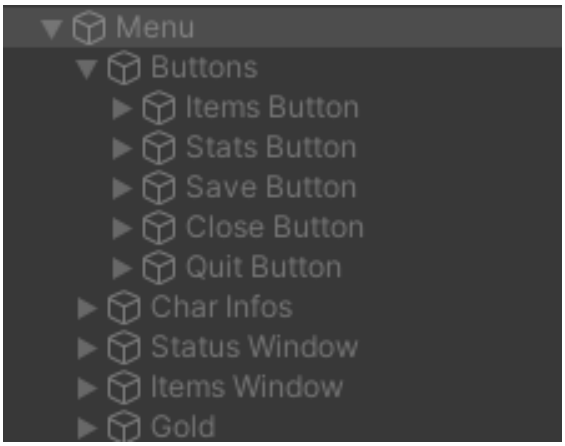

Рисунок 3.18 – Меню паузи

Рисунок 3.19 – Вигляд меню паузи у грі

Основні елементи головного меню ігрового додатка це 3 кнопки та назва гри. Кнопка «Продовжити» - відновити попередню ігрову сесію, «Нова гра» - почату нову ігрову сесію, «Вийти» - закрити додаток та вийти на робочий стіл. Головне меню показано на рисунку 3.20.

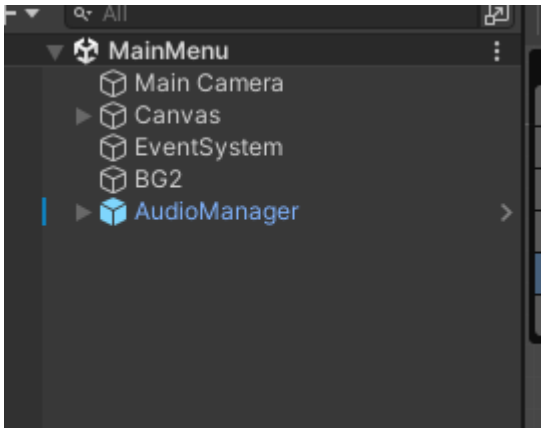

Рисунок 3.20 – Структура головного меню,

Рисунок 3.21 – Вигляд головного меню

Також окрім візуального оформлення гри додаємо аудіо супровід, музика та звуки для окремих дій персонажів, рисунок 3.22.

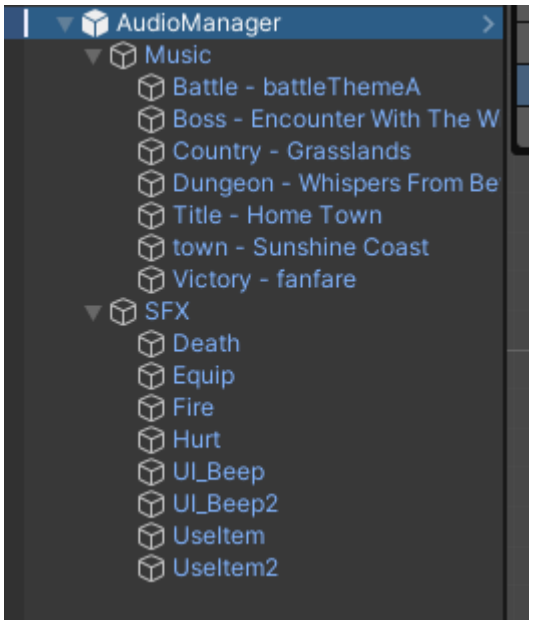

Рисунок 3.22 – Об'єкти для додання аудіо файлів

Рисунок 3.23 – Аудіо файли

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи бакалавра є розроблений ігровий додаток жанру 2D RPG у середовищі Unity «Chronicles of Eldia».

Аналіз предметної області та додатків-аналогів дозволив визначити, у якому середовищі розробки та за допомогою якого програмного забезпечення потрібно виконати ігровий додаток.

Моделювання розробки за допомогою діаграм дозволило визначити основні етапи розробки та розподілити ресурси. Складено сценарій роботи користувача з додатком у вигляді діаграми варіантів використання.

Тестування дозволило завчасно виявити проблеми функціонування програмних файлів та оптимізувати роботу додатка. Тест підтвердив коректне функціонування на платформах користувачів.

Перевагами створеного додатка є унікальне візуальне та аудіо оформлення, геймплей та механіки, характерні для жанру RPG, мультиплатформеність – додатоком можна користуватись на різних платформах, таких як Window, MacOS, Android.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Metacritik [Електронний ресурс] – Режим доступу до ресурсу: https://www.metacritic.com/ (дата звернення 08.05.2023).

2. Fugas.space [Електронний ресурс] – Режим доступу до ресурсу: https://blog.fugas.space/gamedev-stages/ (дата звернення 08.05.2023).

3. Makeuseof [Електронний ресурс] – Режим доступу до ресурсу: https://www.makeuseof.com/key-technologies-rpg-video-games/ (дата звернення 08.05.2023).

4. MOOC.org [Електронний ресурс] – Режим доступу до ресурсу:

https://www.mooc.org/blog/best-programming-languages-for-game-development (дата звернення 08.05.2023)

5. Steam:Stardew Valey [Електронний ресурс] – Режим доступу до ресурсу: https://store.steampowered.com/app/413150/Stardew_Valley/ (дата звернення 08.05.2023).

6. Steam:Undertale [Електронний ресурс] – Режим доступу до ресурсу: https://store.steampowered.com/app/391540/Undertale/ (дата звернення 08.05.2023).

7. JuegoStudios [Електронний ресурс] – режим доступу до ресурсу: https://www.juegostudio.com/blog/reasons-why-you-need-pick-unity-game-development-for-creating-a-2d-game (дата звернення 08.05.2023)

8. Програма Unity Student Plan [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/products/unity-student (дата звернення 08.05.2023).

9. Історія відеоігор [Електронний ресурс] – Режим доступу до ресурсу: http://betar.org.ua/istoriya-viniknennya-stanovlennya-rozvitku-kompyuternih-videoigor-26-foto/ (дата звернення 08.05.2023).

10. Unity manual [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/index.html (дата звернення 08.05.2023).

11. Unity Asset Store [Електронний ресурс] – Режим доступу до ресурсу: https://assetstore.unity.com/ (дата звернення 09.05.2023).

12. DevGenius [Електронний ресурс] – Режим доступу до ресурсу: https://blog.devgenius.io/unity-3d-c-scripting-cheatsheet-for-beginners-be6030b5a9ed (дата звернення 09.05.2023).

13. Microsoft [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/office/ (дата звернення 09.05.2023).

14. Game Narratives [Електронний ресурс] – Режим доступу до ресурсу: https://www.gamedeveloper.com/design/the-evolution-of-video-games-as-a-storytelling-medium-and-the-role-of-narrative-in-modern-games (дата звернення 15.05.2023).

15. Гейг М. Learning C# Unity Game Development 24 hours in, Sams, 2018, 450 с.

16. Бонд Дж. Г. Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#. Addison-Wesley Professional, 2018. 1024 с.

17. Халперн Дж. Developing 2D Games with Unity: Independent Game Programming with C#. Нью-Йорк: Apress, 2018. 405 с.

18. Unity Learn [Електронний ресурс] – Режим доступу до ресурсу: https://learn.unity.com/ (дата звернення 15.05.2023).

19. Unity forum [Електронний ресурс] – Режим доступу до ресурсу: https://forum.unity.com/ (дата звернення 08.05.2023).

20. Енциклопедія сучасної України [Електронний ресурс] – Режим доступу до ресурсу: https://esu.com.ua/article-4393 (дата звернення 08.05.2023).

21. Gamedev.dou [Електронний ресурс] – Режим доступу до ресурсу: https://gamedev.dou.ua/articles/top-gamedev-companies-spring-2022/ (дата звернення 08.05.2023).

22. 24 Games [Електронний ресурс] – Режим доступу до ресурсу: https://games.24tv.ua/ne-lishe-stalker-naypopulyarnishi-ukrayinski-videoigri-igri_n1711564 (дата звернення 08.05.2023).

23. Itch.io [Електронний ресурс] – Режим доступу до ресурсу: https://itch.io/ (дата звернення 15.05.2023).

24. Reddit [Електронний ресурс] – Режим доступу до ресурсу: https://www.reddit.com/r/ShouldIbuythisgame/comments/49uydy/sib_undertale_or_stardew_valley/ (дата звернення 15.05.2023).

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Технічне завдання
на тему
«Розробка ігрового додатка у середовищі Unity»

ПОГОДЖЕНО:
Доцент кафедри комп’ютерних наук
_____ Неня В.Г.
Студент групи ІТ-92/1
_____ Груздо Д.Р.

2023

1 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ

1.1 Призначення web-додатку

Проект має надавати гравцям можливість поринути можливість відчути себе частиною фантастичного світу, в якому вони можуть відчути вплив своїх виборів на історію і персонажів. Головним завданням «Chronicles of Eldia» є надання можливості гравцям досліджувати великий світ, брати участь у пригодах, розкривати таємниці та збирати різноманітні предмети.

1.2 Мета створення web-додатку

Створення цікавого продукту, для людей різних шарів суспільства та національності. Привернути увагу до сфери українського гейм-деву, та розвинути його. Застосувати сучасні технологічні рішення та інструменти для реалізації геймплею, взаємодії з гравцями та надання візуального та звукового досвіду.

1.3 Цільова аудиторія

Цільова аудиторія проекту може включати людей різного віку та інтересів, але зазвичай це гравці, які насолоджуються відеоіграми та фентезі світом.

Основними групами цільової аудиторії «Chronicles of Eldia» можуть бути:

1. Гравці, які люблять екшн та пригоди.
2. Гравці, які цінують історію та можливість впливати на неї.
3. Гравці, які люблять досліджувати ігровий світ.
4. Гравці, які цінують фантастичні історії та світи.

2 ВИМОГИ ДО ДОДАТКУ

1.4 Вимоги до додатку в цілому

1.5 .1 Вимоги до структури й функціонування додатку

Додаток повинен бути доступним в мережі Інтернет, для скачування на одному з сайтів, де є можливість завантаження інді-проектів. Після завантаження exe-файлу та подальшого встановлення на платформу, користувач отримує повний доступ до усіх матеріалів гри.

2.1.2 Вимоги до персоналу

Від розробника вимагається створити продукт, який буде мати мінімальну кількість помилок та багів, що можуть заважати комфортній грі. Підтримувати зв’язок з гравцями, через коментарі на сайті, на якому буде викладена гра, та виправляти помилки знайдені користувачами.

2.1.3 Вимоги до розмежування доступу

Додаток не має потреби до створення баз даних та збереження будь якої інформації.

2.1.4 Вимоги до розмежування доступу

Розроблюваний додаток має бути загальнодоступний. Користувач отримує файл для встановлення гри на платформу, та не має можливості її редагувати.

Розробник вносить корегування та виправляє помилки у роботі додатку у середовищі розробки на власному персональному комп’ютері, після чого готовий білд буде розміщений на сайті.

2.2 Структура додатку

2.2.1 Загальна інформація про структуру додатку

Структура 2D RPG гри може бути досить складною, оскільки вона має багато різноманітних елементів. Проте, основні складові структури гри можуть включати наступні елементи:

1. Світ гри: Це може бути фантастичний світ з різноманітними локаціями, такими як міста, села, ліси, гори, печери, джунглі тощо. Кожна локація може мати свій унікальний дизайн та характеристики.

2. Головний герой: Це може бути персонаж, який контролюється гравцем, з унікальними навичками та характеристиками.

3. Персонажі: Він може бути населений різноманітними персонажами, такими як непрямі супротивники, NPC (герої, які не контролюються гравцем), монстри тощо. Кожен з цих персонажів може мати свої унікальні риси та характеристики.

4. Бойова система: Це може бути система, яка дозволяє гравцю здійснювати бій з іншими персонажами, монстрами та іншими ворогами. Бойова система може бути заснована на різних принципах, таких як ходова система або дійсна-часова бойова система.

5. Інвентар: Це може бути система, яка дозволяє гравцю збирати та управляти предметами, що знайдені в грі, такі як зброя, броня, еліксири тощо.

6. Місії та завдання: Це можуть бути основні та побічні завдання, які гравець повинен виконати, щоб продовжувати гру. Ці завдання будуть засновані на сюжеті гри.

7. Система діалогів: Це фрази персонажів, які будуть показані гравцеві при взаємодії головного героя з NPC, при проходженні завдань.

2.2.2 Навігація

Навігація у ігровому додатку буде здійснюватись через головне меню. У головному меню будуть такі пункти:

- Continue – продовжити гру з моменту, де попередній ігровий сеанс був завершений.
- New Game – почати гру заново.
- Exit Game – вийти з додатку.

Меню буде мати спеціальне оформлення та дизайн.

2.2.3 Наповнення додатку (контент)

Усі матеріали додатку (Assets) будуть заповнені розробником - аудіо, зображення, текстури.

2.2.4 Дизайн та структура додатку

Дизайн додатку буде створений відповідно до жанру гри, загальної стилістики та сюжету. Будуть створені унікальні оформлення пунктів меню, діалогової системи, різних локацій, персонажів та всього візуального контенту додатка.

За основу розробленого дизайну буде обрано сюжет гри – фентезійний світ.

Структура елементів головного меню додатку схематично показано на рисунку А.1.

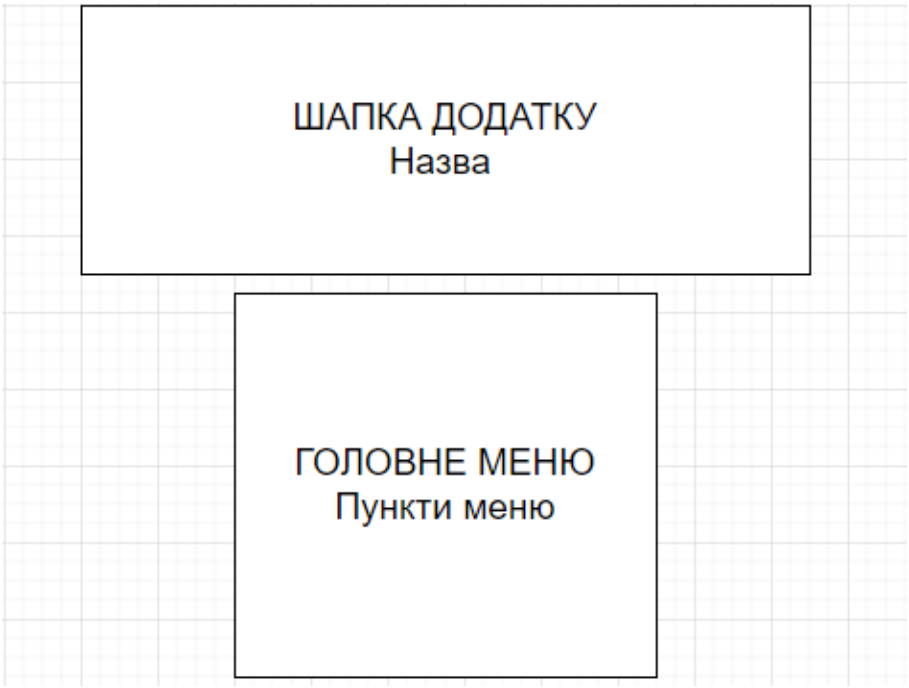

Рисунок А.1 – Схема головного меню

2.2.5 Система навігації (карта додатку)

Карта додатку зображена на рисунку А.2.

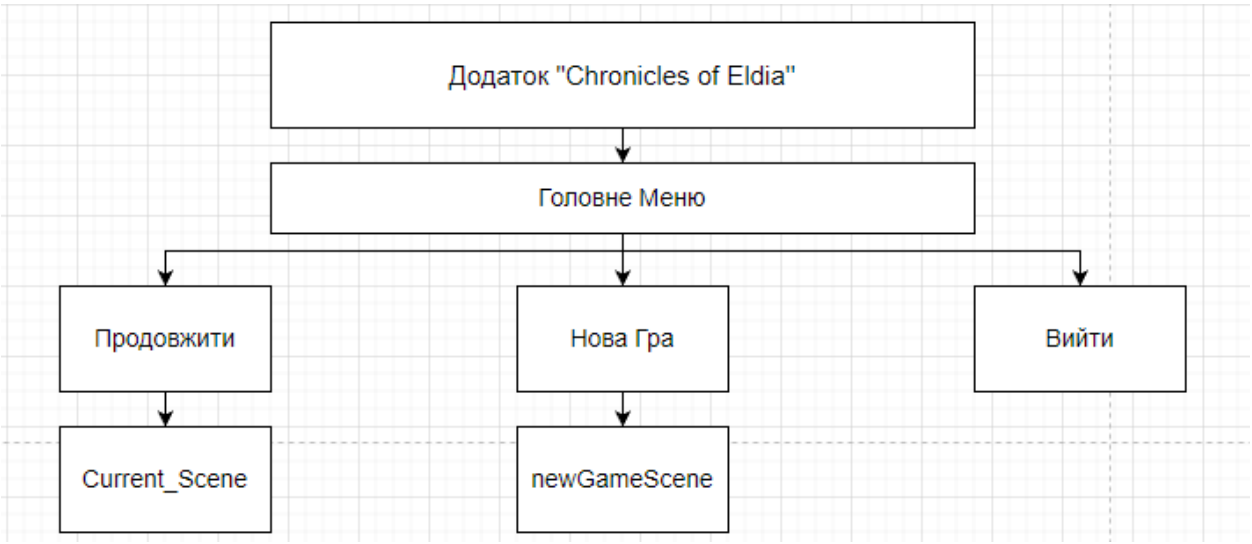

Рисунок А.2 – Карта додатку

1.6 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі загальних вимог до готового ігрового додатку, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Коректне встановлення додатка на платформу користувача	Клієнт
UN-02	Функціонування скриптів головного меню	Клієнт
UN-03	Коректне відтворення аудіо-файлів на платформі користувача	Клієнт
UN-04	Функціонування скриптів керування головним персонажем гри	Клієнт
UN-05	Можливість використовувати систему діалогів	Клієнт
UN-06	Користування системою бойових механік	Клієнт
UN-07	Взаємодія з NPC (персонажі керовані комп’ютером)	Клієнт
UN-08	Збереження прогресу гри	Клієнт

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

- авторизація користувачів;
- навігація через головне меню;

- відображення результатів гри;
- збереження ігрового процесу;
- правильний розрахунок ігрових механік.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	**Системні вимоги**	**Пріоритет**	**Опис**
SR-01	Наявність модуля головного меню	M	Надає можливість гравцю користуватись додатком та орієнтуватись у його функціях
SR-02	Журнал завдань	S	Пункт де користувач може читати деталі свого завдання, для проходження гри
SR-03	Локації гри	M	Різні ігрові області, де розташовується ігровий контент
SR-04	Різні види NPC	S	Різноманіття персонажів, які не керуються гравцем
SR-05	Скрипт розрахунку прогресії	M	Система, яка буде розраховувати кількість досвіду та ресурсів, які гравець буде отримувати впродовж гри

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have © – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація додатку відбувається з використанням:

- Unity
- C#

2.4.2 Вимоги до лінгвістичного забезпечення

Додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення користувача повинне задовольняти наступні вимоги:

- Операційна система: Windows 7 або новіша.
- Процесор: Intel Core i3 або еквівалентний AMD процесор.
- Оперативна пам’ять: мінімум 4 ГБ.
- Відеокарта: мінімум 512 МБ.
- Вільне місце на диску: мінімум 500 МБ.
- Звукова карта: рекомендовано мати звукову карту

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ ДОДАТКУ

Докладний опис етапів роботи зі створення додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка цілей необхідних для досягнення результату	1 день
2	Складання технічного завдання	1 день
3	Створення основної сцени	1 день
4	Створення головного героя	1 день
5	Створення локацій	2 дні
6	Налаштування камери та мапи	1 день
7	Створення діалогової системи	2 дні
8	Система прогресії	1 день
9	Створення головного меню	1 день
10	Інвентар головного героя	1 день
11	Ресурси та магазини у середині гри	1 день
12	Розробка завдань та сюжету	3 дні
13	Бойова система	3 дні
14	Завершення роботи	1 день
	Загальна тривалість робіт	20 днів

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВВЕДЕННЯ ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Для того, щоб додатком могли користуватись гравці необхідно розмістити його у мережі інтернет, у вільному доступі. Для цього буде використаний сайт Itch.io. Itch.io- це онлайн сервіс для розміщення, продажу та завантаження інді-ігор. Для коректного завантаження та використання, потрібно, щоб додаток відповідав вимогам зазначеним у ТЗ.

ДОДАТОК Б

Планування робіт

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є ігровий додаток жанру – 2D RPG.

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 – Деталізація методом SMART розміщені у табл. Б.1

Specific (конкретна)	Створити ігровий додаток у жанрі – 2D RPG.
Measurable (вимірювання)	Результатом роботи є оцінка користувачів.
Achievable (досяжна)	Реалізація проекту здійснюється у середовищі розробки Unity, з використання мови програмування C#.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у певний строк.

Планування змісту структури робіт

Для планування змісту структури робіт служить WBS діаграма – інструмент проектного менеджменту, що використовується для декомпозиції проекту на менші, керовані елементи. Нижче наведено структуру WBS, у якій детально описано роботи, які потрібно виконати на кожному етапі створення проекту. Діаграма WBS зображена на рисунку Б.1.

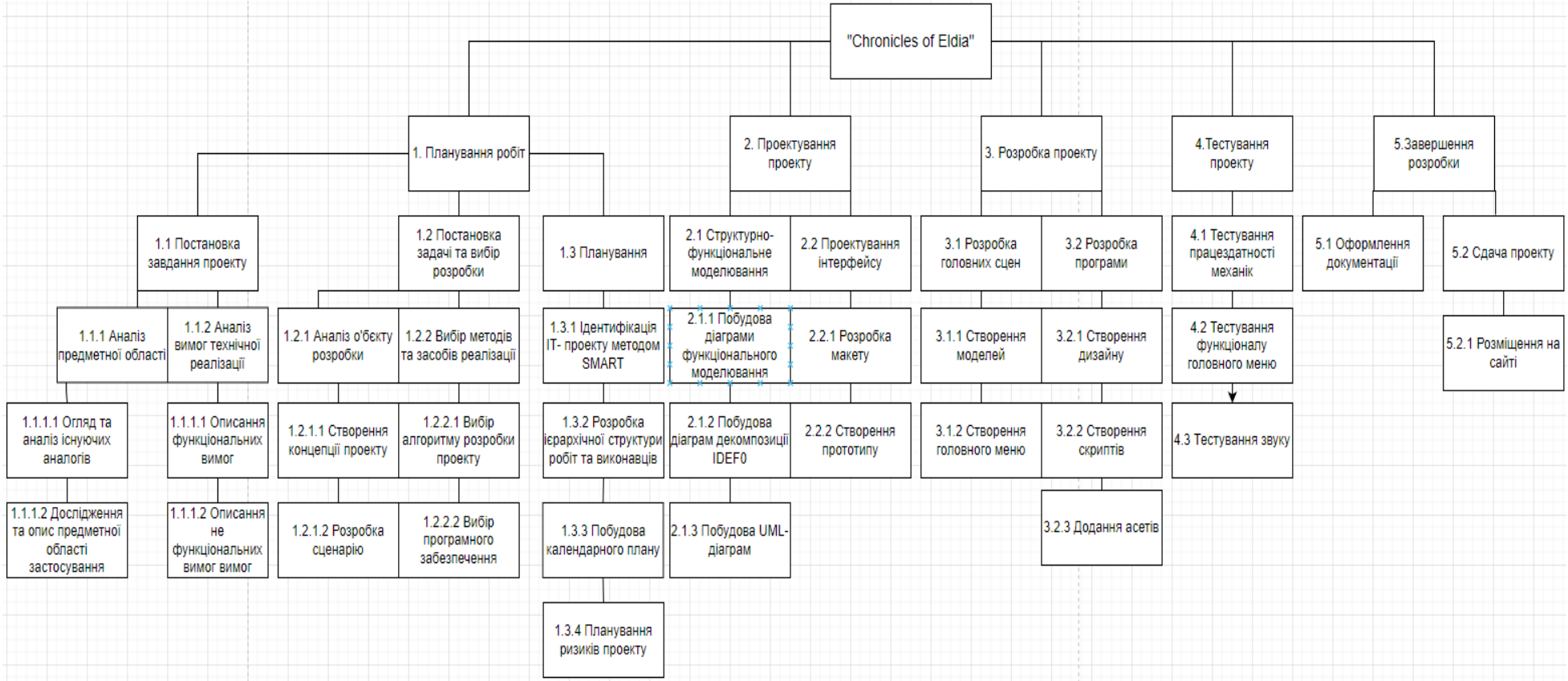

Рисунок Б.1 – WBS Структура робіт проекту

Планування структури організації, для впровадження готового проекту (OBS)

OBS (Organizational Breakdown Structure) діаграма - це інструмент, що використовується для візуалізації структури організації проекту та розподілу відповідальностей за проект між членами команди. Основною метою OBS діаграми є забезпечення чіткого розуміння того, яка роль та відповідальність призначена для кожного члена команди. OBS діаграма зображена на рисунку Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Груздо Д.Р.	Виконує розробку основного функціоналу та інтерфейс проекту
Дизайнер	Груздо Д.Р.	Створює зовнішній вигляд моделей та дизайн інтерфейсу
Тестувальник	Груздо Д.Р.	Відповідає за тестування працездатності додатку, перевірку на критичні помилки
Консультант проекту	Неня В.Г.	Формує завдання на розробку проекту
Менеджер	Груздо Д.Р.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

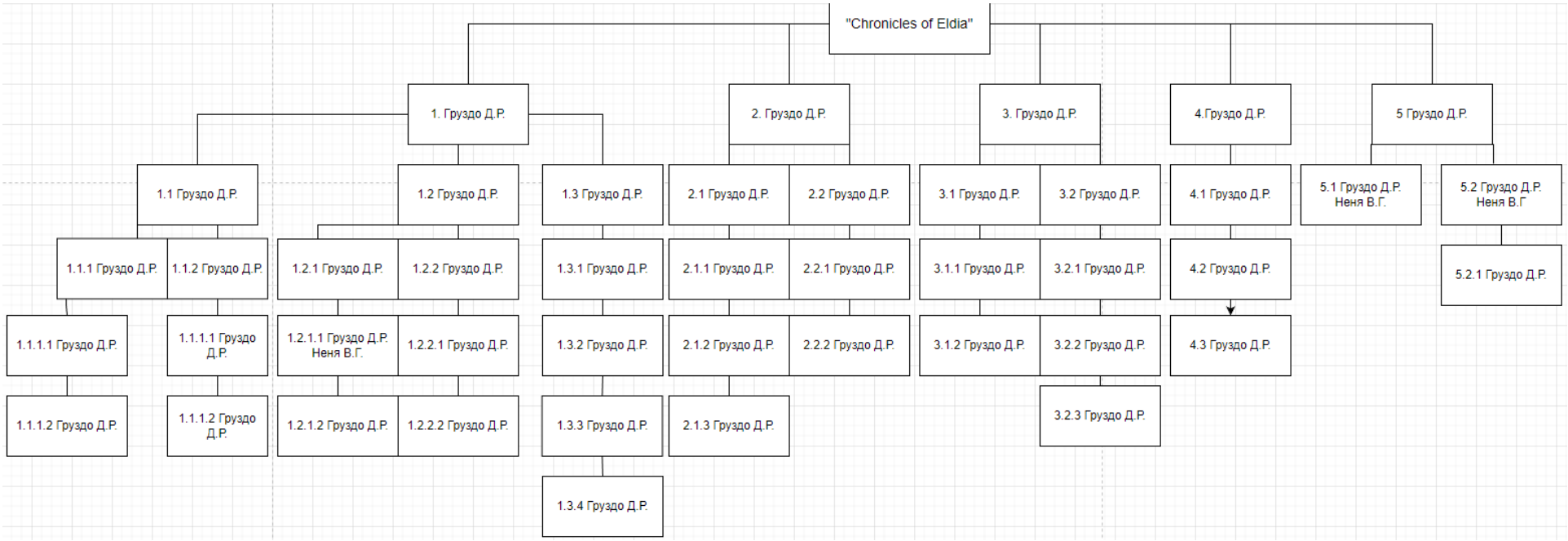

Рисунок Б.2 – Організаційна структура проекту (OBS)

Діаграма Ганта

Діаграма Ганта - це інструмент проектного менеджменту, який використовується для візуалізації термінів та послідовності завдань проекту. Вона складається з графіка, на якому зображені часові рамки проекту та завдання, які потрібно виконати, їх тривалість та залежності між ними. На рисунках Б.3-Б.6 зображені діаграма Ганта та список робіт діаграми Ганта.

ДИАГРАММА ГАНТА

	Режим задачі	Название задачи	Длительность	Начало	Окончание	Пред
1		**Планування робіт**	**3 дней**	**Пн 17.04.23**	**Ср 19.04.23**	
2		**Постановка завдання проекту**	**1 день**	**Пн 17.04.23**	**Пн 17.04.23**	
3		**Аналіз предметної області**	**1 день**	**Пн 17.04.23**	**Пн 17.04.23**	
4		Огляд та аналіз існуючих аналогів	1 день	Пн 17.04.23	Пн 17.04.23	
5		Дослідження та опис предметної області застосування	1 день	Пн 17.04.23	Пн 17.04.23	
6		**Аналіз вимог технічної реалізації**	**1 день**	**Пн 17.04.23**	**Пн 17.04.23**	
7		Описання функціональних вимог	1 день	Пн 17.04.23	Пн 17.04.23	
8		Описання не функціональних вимог	1 день	Пн 17.04.23	Пн 17.04.23	
9		**Постановка задачі та вибір розробки**	**1 день**	**Вт 18.04.23**	**Вт 18.04.23**	
10		**Аналіз об'єкту розробки**	**1 день**	**Вт 18.04.23**	**Вт 18.04.23**	
11		Створення концепції проекту	1 день	Вт 18.04.23	Вт 18.04.23	
12		Розробка сценарію	1 день	Вт 18.04.23	Вт 18.04.23	
13		**Вибір методів та засобів реалізації**	**1 день**	**Вт 18.04.23**	**Вт 18.04.23**	
14		Вибір алгоритму розробки	1 день	Вт 18.04.23	Вт 18.04.23	
15		Вибір програмного забезпечення	1 день	Вт 18.04.23	Вт 18.04.23	
16		**Планування**	**1 день**	**Ср 19.04.23**	**Ср 19.04.23**	
17		Ідентифіцація IT-проекту методом SMART	1 день	Ср 19.04.23	Ср 19.04.23	
18		Розробка ієрархічної структури робіт та виконавців	1 день	Ср 19.04.23	Ср 19.04.23	
19		Побудова календарного плану	1 день	Ср 19.04.23	Ср 19.04.23	
20		Планування ризиків проекту	1 день	Ср 19.04.23	Ср 19.04.23	
21		**Проектування проекту**	**2 дней**	**Чт 20.04.23**	**Пт 21.04.23**	
22		**Структруно функціональне моделюванн**	**1 день**	**Чт 20.04.23**	**Чт 20.04.23**	
23		Побудова діаграми функціонального моделювання	1 день	Чт 20.04.23	Чт 20.04.23	
24		Побудова діаграм декомпозиції IDEF0	1 день	Чт 20.04.23	Чт 20.04.23	
25		Побудова UML-діаграм	1 день	Чт 20.04.23	Чт 20.04.23	
26		**Проектування інтерфейсу**	**1 день**	**Пт 21.04.23**	**Пт 21.04.23**	
27		Розробка макету	1 день	Пт 21.04.23	Пт 21.04.23	
28		Створення прототипу	1 день	Пт 21.04.23	Пт 21.04.23	
29		**Розробка проекту**	**17 дней**	**Пн 24.04.23**	**Вт 16.05.23**	
30		**Розробка головних сцен**	**7 дней**	**Пн 24.04.23**	**Вт 02.05.23**	

Рисунок Б.5 – Список робіт для побудови діаграми Ганта

30			**Розробка головних сцен**	**7 дней**	**Пн 24.04.23**	**Вт 02.05.23**
31			Створення моделей	6 дней	Пн 24.04.23	Пн 01.05.23
32			Створення головного меню	1 день	Вт 02.05.23	Вт 02.05.23
33			**Розробка програми**	**10 дней**	**Ср 03.05.23**	**Вт 16.05.23**
34			Створення дизайну	4 дней	Ср 03.05.23	Пн 08.05.23
35			Створення скриптів	5 дней	Вт 09.05.23	Пн 15.05.23
36			Додавання асетів	1 день	Вт 16.05.23	Вт 16.05.23
37			**Тестування проекту**	**1 день**	**Ср 17.05.23**	**Ср 17.05.23**
38			Тестування працездатності механік	1 день	Ср 17.05.23	Ср 17.05.23
39			Темтування функціоналу головного менк	1 день	Ср 17.05.23	Ср 17.05.23
40			Тестування звуку	1 день	Ср 17.05.23	Ср 17.05.23
41			**Завершення розробки**	**1 день**	**Чт 18.05.23**	**Чт 18.05.23**
42			Оформлення документації	1 день	Чт 18.05.23	Чт 18.05.23
43			Сдача проекту	1 день	Чт 18.05.23	Чт 18.05.23
44			Розміщення на сайті	1 день	Чт 18.05.23	Чт 18.05.23

Рисунок Б.6 – Продовження списку робіт для побудови діаграми Ганта

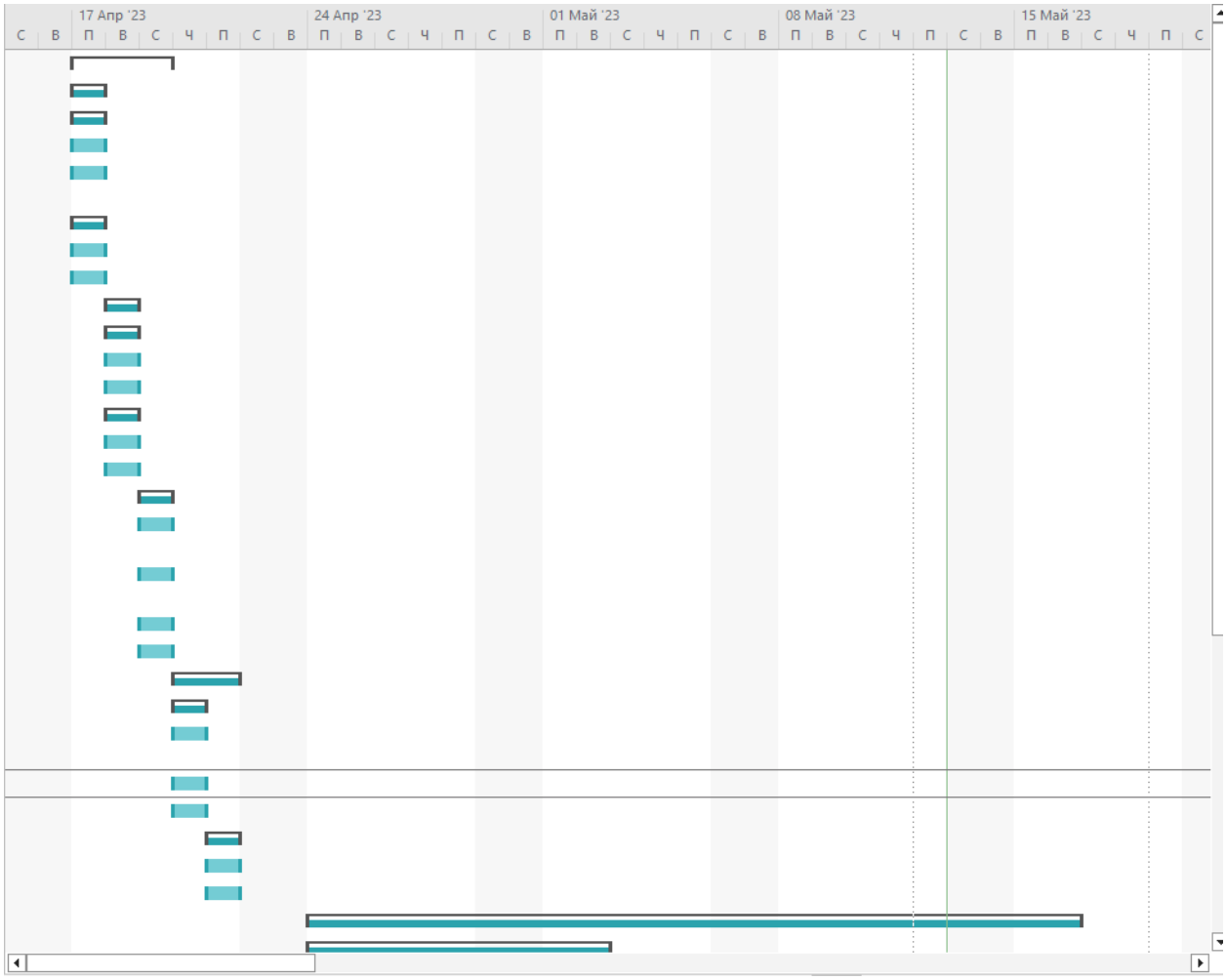

Рисунок Б.3 – Діаграма Ганта

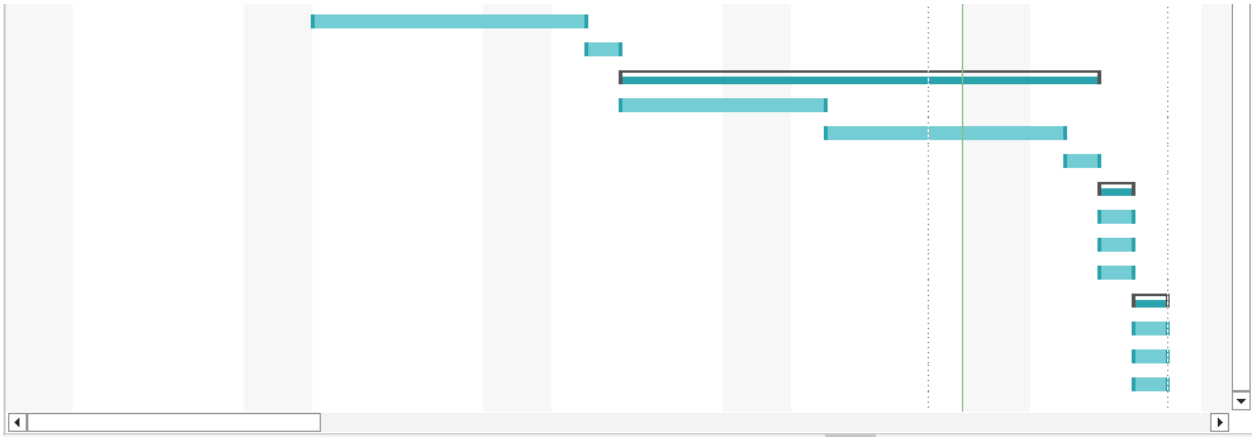

Рисунок Б.4 – Продовження діаграми Ганта

Аналіз ризиків

Аналіз ризиків є важливою частиною управління проектом, оскільки дозволяє ідентифікувати потенційні проблеми та забезпечувати можливість попереднього планування та управління їхнім впливом на проект. Виконаємо якісну та кількісну оцінку ризиків.

Якісна оцінка ризиків полягає у визначенні ймовірності виникнення ризиків та їхнього впливу на проект. Цей підхід полягає у використанні експертної оцінки та досвіду професіоналів для оцінки того, наскільки важливими є ризики та як вони можуть вплинути на проект. Результатом якісної оцінки ризиків є список потенційних проблем та їхніх відносних пріоритетів, що може бути використано для розробки плану управління ризиками.

Кількісна оцінка ризиків передбачає вимірювання ймовірності виникнення ризиків та їхнього впливу на проект у конкретних числових значеннях. Для цього можна використовувати статистичні методи, математичні моделі та інші інструменти аналізу.

Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У таблиці Б.5 знаходиться класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики — це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним. Оцінюємо ризики за показниками, що знаходяться в табл. Б.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на рис. Б.7.

Таблиця Б.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

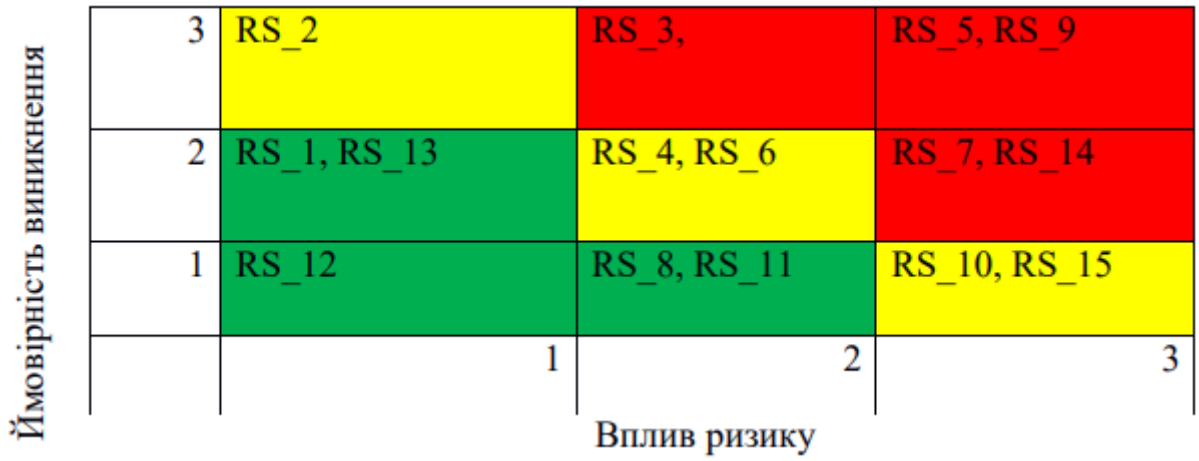

Рисунок Б.7 – Матриця ймовірності виникнення ризиків та впливу ризику

На рисунку Б.7 використано такі позначення:

- зелений колір – прийнятні ризики;
- жовтий колір – виправданні ризики;
- червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в таблиці Б.4.

Таблиця Б.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	8,12,13
2	Виправдані	3≤R≤4	2,4,6,10,11,15
3	Недопустимі	6≤R≤9	1,3,5,9,14

Таблиця Б.5 – Оцінка ймовірності виникнення, величини втрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Середній	Високий	6	Налагодити співпрацю с замовником. Дозволити замовнику контролювати етапи роботи.	Попередження	Оперативне виявлення невідповідностей між заявленими умовами та характеристиками продукту.
RS_2	Відкритий	Поява альтернативного продукту	Низька	Високий	3	Провести аналіз додатків-аналогів	Прийняття	
RS_3	Відкритий	Нечітке завдання на розробку	Середня	Високий	7	Під час планування робіт, чітко обговорити усі види вимог	Попередження	При виявлені невідповідностей характеристик продукту до вимог, треба чітко визначити помилки та зробити правки.
RS_4	Відкритий	Низька кваліфікація розробників проекту	Середня	Середній	4	Переглянути ресурси для підвищення рівня кваліфікації.	Пом’якшення	Врахувати час на підготовку. Знайти ресурси для підвищення рівня знань.

Продовження таблиці Б.5

RS_5	Відкритий	Неоптимальний розподіл часу	Висока	Високий	9	Провести аналіз найактуальніших процесів та робіт.	Пом’якшення	Внести правки до термінів реалізації робіт. Змінити пріоритети процесів роботи.
RS_6	Відкритий	Часте внесення змін у ТЗ	Низька	Середній	4	Описати вимоги до проекту. Обговорити технічні засоби виконання проекту.	Перенос	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.
RS_7	Відкритий	Вибір неефективної технології розробки	Низька	Високий	8	Проаналізувати методи та засоби, для виконання проекту.	Пом’якшення	Виділити час на зміну засобів розробки.
RS_8	Відкритий	Не вірна оцінка масштабів проекту	Низька	Середній	2	Провести аналіз проекту. Детально описати постановку задачі. Проаналізувати масштаби проекту.	Пом’якшення	Перебудова стратегії розробки
RS_9	Відкритий	Помилки проектування	Висока	Високий	9	На етапі проектування тісно співпрацювати с замовником.	Пом'якшення	Здійснювати проміжний контроль результатів виконання проекту.

Продовження таблиці Б.5

RS_10	Відкритий	Збої в роботі програмного забезпечення	Низька	Високий	3	Підготувати резерв програмних засобів.	Попередження	Замінити програмне забезпечення.
RS_11	Відкритий	Відсутність резервних копій даних	Низька	Середній	4	Автоматичне збереження даних. Мати декілька резервних носіїв інформації.	Попередження	Зберігати дані після кожного етапу розробки.
RS_12	Відкритий	Реалізація непотрібної функціональності	Низька	Низький	1	Попередити замовника про додатковий функціонал.	Використання	Проаналізувати вигоди і збитки від змін проекту.
RS_13	Відкритий	Невиконання моніторингу проекту	Низька	Низький	2	Контролювати результати в ході створення проекту.	Перенос	Замовник повинен здійснювати контроль розробки. Надавати проміжні результати після кожного етапу.
RS_14	Відкритий	Виникнення проблем із програмним забезпеченням користувачів.	Середня	Високий	7	Модифікація проекту з врахуванням різних версій програмного забезпечення.	Прийняття	
RS_15	Відкритий	Зміна вимог замовника в процесі розробки проекту	Низька	Високий	3	Узгодити всі питання на початкових етапах, щоб мінімізувати кількість змін під час розробки.	Пом’якшення	Переоцінка проекту кожного разу, коли вимоги змінюються.

ДОДАТОК В

ЛІСТИНГ МОДУЛІВ

Файл GameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour {

    public static GameManager instance;

    public CharStats[] playerStats;

    public bool gameMenuOpen, dialogActive, fadingBetweenAreas, shopActive, battleActive;

    public string[] itemsHeld;
    public int[] numberOfItems;
    public Item[] referenceItems;

    public int currentGold;

       // Use this for initialization
       void Start () {
        instance = this;

        DontDestroyOnLoad(gameObject);

        SortItems();
       }

       // Update is called once per frame
       void Update () {
             if(gameMenuOpen || dialogActive || fadingBetweenAreas || shopActive ||
battleActive)
        {
            PlayerController.instance.canMove = false;
        } else
        {
            PlayerController.instance.canMove = true;
        }

        if(Input.GetKeyDown(KeyCode.J))
        {
            AddItem("Iron Armor");
            AddItem("Blabla");

            RemoveItem("Health Potion");
            RemoveItem("Bleep");
        }

        if (Input.GetKeyDown(KeyCode.O))
        {
            SaveData();
        }

        if (Input.GetKeyDown(KeyCode.P))
        {
            LoadData();
        }
```

```
}

public Item GetItemDetails(string itemToGrab)
{

    for(int i = 0; i < referenceItems.Length; i++)
    {
        if(referenceItems[i].itemName == itemToGrab)
        {
            return referenceItems[i];
        }
    }



    return null;
}

public void SortItems()
{
    bool itemAFterSpace = true;

    while (itemAFterSpace)
    {
        itemAFterSpace = false;
        for (int i = 0; i < itemsHeld.Length - 1; i++)
        {
            if (itemsHeld[i] == "")
            {
                itemsHeld[i] = itemsHeld[i + 1];
                itemsHeld[i + 1] = "";

                numberOfItems[i] = numberOfItems[i + 1];
                numberOfItems[i + 1] = 0;

                if(itemsHeld[i] != "")
                {
                    itemAFterSpace = true;
                }
            }
        }
    }
}

public void AddItem(string itemToAdd)
{
    int newItemPosition = 0;
    bool foundSpace = false;

    for(int i = 0; i < itemsHeld.Length; i++)
    {
        if(itemsHeld[i] == "" || itemsHeld[i] == itemToAdd)
        {
            newItemPosition = i;
            i = itemsHeld.Length;
            foundSpace = true;
        }
    }

    if(foundSpace)
    {
        bool itemExists = false;
        for(int i = 0; i < referenceItems.Length; i++)
        {
```

```
                if(referenceItems[i].itemName == itemToAdd)
                {
                    itemExists = true;

                    i = referenceItems.Length;
                }
            }

            if(itemExists)
            {
                itemsHeld[newItemPosition] = itemToAdd;
                numberOfItems[newItemPosition]++;
            } else
            {
                Debug.LogError(itemToAdd + " Does Not Exist!!");
            }
        }

        GameMenu.instance.ShowItems();
    }

    public void RemoveItem(string itemToRemove)
    {
        bool foundItem = false;
        int itemPosition = 0;

        for(int i = 0; i < itemsHeld.Length; i++)
        {
            if(itemsHeld[i] == itemToRemove)
            {
                foundItem = true;
                itemPosition = i;

                i = itemsHeld.Length;
            }
        }

        if(foundItem)
        {
            numberOfItems[itemPosition]--;

            if(numberOfItems[itemPosition] <= 0)
            {
                itemsHeld[itemPosition] = "";
            }

            GameMenu.instance.ShowItems();
        } else
        {
            Debug.LogError("Couldn't find " + itemToRemove);
        }
    }

    public void SaveData()
    {
        PlayerPrefs.SetString("Current_Scene", SceneManager.GetActiveScene().name);
        PlayerPrefs.SetFloat("Player_Position_x",
PlayerController.instance.transform.position.x);
        PlayerPrefs.SetFloat("Player_Position_y",
PlayerController.instance.transform.position.y);
        PlayerPrefs.SetFloat("Player_Position_z",
PlayerController.instance.transform.position.z);

        //save character info
        for(int i = 0; i < playerStats.Length; i++)
```

```
        {
            if(playerStats[i].gameObject.activeInHierarchy)
            {
                PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_active", 1);
            } else
            {
                PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_active", 0);
            }

            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_Level",
playerStats[i].playerLevel);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_CurrentExp",
playerStats[i].currentEXP);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_CurrentHP",
playerStats[i].currentHP);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_MaxHP",
playerStats[i].maxHP);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_CurrentMP",
playerStats[i].currentMP);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_MaxMP",
playerStats[i].maxMP);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_Strength",
playerStats[i].strength);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_Defence",
playerStats[i].defence);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_WpnPwr",
playerStats[i].wpnPwr);
            PlayerPrefs.SetInt("Player_" + playerStats[i].charName + "_ArmrPwr",
playerStats[i].armrPwr);
            PlayerPrefs.SetString("Player_" + playerStats[i].charName + "_EquippedWpn",
playerStats[i].equippedWpn);
            PlayerPrefs.SetString("Player_" + playerStats[i].charName + "_EquippedArmr",
playerStats[i].equippedArmr);
        }

        //store inventory data
        for(int i = 0; i < itemsHeld.Length; i++)
        {
            PlayerPrefs.SetString("ItemInInventory_" + i, itemsHeld[i]);
            PlayerPrefs.SetInt("ItemAmount_" + i, numberOfItems[i]);
        }
    }

    public void LoadData()
    {
        PlayerController.instance.transform.position = new
Vector3(PlayerPrefs.GetFloat("Player_Position_x"),
PlayerPrefs.GetFloat("Player_Position_y"), PlayerPrefs.GetFloat("Player_Position_z"));

        for(int i = 0; i < playerStats.Length; i++)
        {
            if(PlayerPrefs.GetInt("Player_" + playerStats[i].charName + "_active") == 0)
            {
                playerStats[i].gameObject.SetActive(false);
            } else
            {
                playerStats[i].gameObject.SetActive(true);
            }

            playerStats[i].playerLevel = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_Level");
            playerStats[i].currentEXP = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_CurrentExp");
            playerStats[i].currentHP = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_CurrentHP");
```

```
            playerStats[i].maxHP = PlayerPrefs.GetInt("Player_" + playerStats[i].charName
+ "_MaxHP");
            playerStats[i].currentMP = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_CurrentMP");
            playerStats[i].maxMP = PlayerPrefs.GetInt("Player_" + playerStats[i].charName
+ "_MaxMP");
            playerStats[i].strength = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_Strength");
            playerStats[i].defence = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_Defence");
            playerStats[i].wpnPwr = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_WpnPwr");
            playerStats[i].armrPwr = PlayerPrefs.GetInt("Player_" +
playerStats[i].charName + "_ArmrPwr");
            playerStats[i].equippedWpn = PlayerPrefs.GetString("Player_" +
playerStats[i].charName + "_EquippedWpn");
            playerStats[i].equippedArmr = PlayerPrefs.GetString("Player_" +
playerStats[i].charName + "_EquippedArmr");
        }

        for(int i = 0; i < itemsHeld.Length; i++)
        {
            itemsHeld[i] = PlayerPrefs.GetString("ItemInInventory_" + i);
            numberOfItems[i] = PlayerPrefs.GetInt("ItemAmount_" + i);
        }
    }
}
```

Файл PlayerController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour {

    public Rigidbody2D theRB;
    public float moveSpeed;

    public Animator myAnim;

    public static PlayerController instance;

    public string areaTransitionName;
    private Vector3 bottomLeftLimit;
    private Vector3 topRightLimit;

    public bool canMove = true;

       // Use this for initialization
       void Start () {
        if (instance == null)
        {
            instance = this;
        } else
        {
            if (instance != this)
            {
                Destroy(gameObject);
            }
        }

        DontDestroyOnLoad(gameObject);
```

```
        }

        // Update is called once per frame
        void Update () {
         if (canMove)
         {
             theRB.velocity = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")) * moveSpeed;

         } else
         {
             theRB.velocity = Vector2.zero;
         }


         myAnim.SetFloat("moveX", theRB.velocity.x);
         myAnim.SetFloat("moveY", theRB.velocity.y);

         if (Input.GetAxisRaw("Horizontal") == 1 || Input.GetAxisRaw("Horizontal") == -1
|| Input.GetAxisRaw("Vertical") == 1 || Input.GetAxisRaw("Vertical") == -1)
         {
             if (canMove)
             {
                 myAnim.SetFloat("lastMoveX", Input.GetAxisRaw("Horizontal"));
                 myAnim.SetFloat("lastMoveY", Input.GetAxisRaw("Vertical"));
             }
         }

         transform.position = new Vector3(Mathf.Clamp(transform.position.x,
bottomLeftLimit.x, topRightLimit.x), Mathf.Clamp(transform.position.y, bottomLeftLimit.y,
topRightLimit.y), transform.position.z);
    }

    public void SetBounds(Vector3 botLeft, Vector3 topRight)
    {
        bottomLeftLimit = botLeft + new Vector3(.5f, 1f, 0f);
        topRightLimit = topRight + new Vector3(-.5f, -1f, 0f);
    }
}
```

Файл DialogManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DialogManager : MonoBehaviour {

    public Text dialogText;
    public Text nameText;
    public GameObject dialogBox;
    public GameObject nameBox;

    public string[] dialogLines;

    public int currentLine;
    private bool justStarted;

    public static DialogManager instance;

    private string questToMark;
    private bool markQuestComplete;
```

```
private bool shouldMarkQuest;

// Use this for initialization
void Start () {
    instance = this;

    //dialogText.text = dialogLines[currentLine];
   }

   // Update is called once per frame
   void Update () {

    if(dialogBox.activeInHierarchy)
    {
        if(Input.GetButtonUp("Fire1"))
        {
            if (!justStarted)
            {
                currentLine++;

                if (currentLine >= dialogLines.Length)
                {
                    dialogBox.SetActive(false);

                    GameManager.instance.dialogActive = false;

                    if(shouldMarkQuest)
                    {
                        shouldMarkQuest = false;
                        if(markQuestComplete)
                        {
                            QuestManager.instance.MarkQuestComplete(questToMark);
                        } else
                        {
                            QuestManager.instance.MarkQuestIncomplete(questToMark);
                        }
                    }
                }
                else
                {
                    CheckIfName();

                    dialogText.text = dialogLines[currentLine];
                }
            } else
            {
                justStarted = false;
            }


        }
    }

   }

public void ShowDialog(string[] newLines, bool isPerson)
{
    dialogLines = newLines;

    currentLine = 0;

    CheckIfName();

    dialogText.text = dialogLines[currentLine];
    dialogBox.SetActive(true);
```

```
        justStarted = true;

        nameBox.SetActive(isPerson);

        GameManager.instance.dialogActive = true;
    }

    public void CheckIfName()
    {
        if(dialogLines[currentLine].StartsWith("n-"))
        {
            nameText.text = dialogLines[currentLine].Replace("n-", "");
            currentLine++;
        }
    }

    public void ShouldActivateQuestAtEnd(string questName, bool markComplete)
    {
        questToMark = questName;
        markQuestComplete = markComplete;

        shouldMarkQuest = true;
    }
}
```

Файл CameraController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Tilemaps;

public class CameraController : MonoBehaviour {

    public Transform target;

    public Tilemap theMap;
    private Vector3 bottomLeftLimit;
    private Vector3 topRightLimit;

    private float halfHeight;
    private float halfWidth;

    public int musicToPlay;
    private bool musicStarted;

       // Use this for initialization
       void Start () {
        //target = PlayerController.instance.transform;
        target = FindObjectOfType<PlayerController>().transform;

        halfHeight = Camera.main.orthographicSize;
        halfWidth = halfHeight * Camera.main.aspect;

        theMap.CompressBounds();
        bottomLeftLimit = theMap.localBounds.min + new Vector3(halfWidth, halfHeight,
0f);
        topRightLimit = theMap.localBounds.max + new Vector3(-halfWidth, -halfHeight,
0f);

        PlayerController.instance.SetBounds(theMap.localBounds.min,
theMap.localBounds.max);
```

```
        }

        // LateUpdate is called once per frame after Update
        void LateUpdate () {
         transform.position = new Vector3(target.position.x, target.position.y,
transform.position.z);


         transform.position = new Vector3(Mathf.Clamp(transform.position.x,
bottomLeftLimit.x, topRightLimit.x), Mathf.Clamp(transform.position.y, bottomLeftLimit.y,
topRightLimit.y), transform.position.z);

         if(!musicStarted)
         {
             musicStarted = true;
             AudioManager.instance.PlayBGM(musicToPlay);
         }
        }
}
```

Файл MainMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameMenu : MonoBehaviour {

    public GameObject theMenu;
    public GameObject[] windows;

    private CharStats[] playerStats;

    public Text[] nameText, hpText, mpText, lvlText, expText;
    public Slider[] expSlider;
    public Image[] charImage;
    public GameObject[] charStatHolder;

    public GameObject[] statusButtons;

    public Text statusName, statusHP, statusMP, statusStr, statusDef, statusWpnEqpd,
statusWpnPwr, statusArmrEqpd, statusArmrPwr, statusExp;
    public Image statusImage;

    public ItemButton[] itemButtons;
    public string selectedItem;
    public Item activeItem;
    public Text itemName, itemDescription, useButtonText;

    public GameObject itemCharChoiceMenu;
    public Text[] itemCharChoiceNames;

    public static GameMenu instance;
    public Text goldText;

    public string mainMenuName;

    // Use this for initialization
    void Start () {
        instance = this;
       }
```

```
        // Update is called once per frame
        void Update () {
              if(Input.GetButtonDown("Fire2"))
         {
             if(theMenu.activeInHierarchy)
             {
                 //theMenu.SetActive(false);
                 //GameManager.instance.gameMenuOpen = false;

                 CloseMenu();
             } else
             {
                 theMenu.SetActive(true);
                 UpdateMainStats();
                 GameManager.instance.gameMenuOpen = true;
             }

             AudioManager.instance.PlaySFX(5);
         }
        }

    public void UpdateMainStats()
    {
        playerStats = GameManager.instance.playerStats;

        for(int i = 0; i < playerStats.Length; i++)
        {
            if(playerStats[i].gameObject.activeInHierarchy)
            {
                charStatHolder[i].SetActive(true);

                nameText[i].text = playerStats[i].charName;
                hpText[i].text = "HP: " + playerStats[i].currentHP + "/" +
playerStats[i].maxHP;
                mpText[i].text = "MP: " + playerStats[i].currentMP + "/" +
playerStats[i].maxMP;
                lvlText[i].text = "Lvl: " + playerStats[i].playerLevel;
                expText[i].text = "" + playerStats[i].currentEXP + "/" +
playerStats[i].expToNextLevel[playerStats[i].playerLevel];
                expSlider[i].maxValue =
playerStats[i].expToNextLevel[playerStats[i].playerLevel];
                expSlider[i].value = playerStats[i].currentEXP;
                charImage[i].sprite = playerStats[i].charIamge;
            } else
            {
                charStatHolder[i].SetActive(false);
            }
        }

        goldText.text = GameManager.instance.currentGold.ToString() + "g";
    }

    public void ToggleWindow(int windowNumber)
    {
        UpdateMainStats();

        for(int i = 0; i < windows.Length; i++)
        {
            if(i == windowNumber)
            {
                windows[i].SetActive(!windows[i].activeInHierarchy);
            } else
            {
                windows[i].SetActive(false);
```

```
            }
        }

        itemCharChoiceMenu.SetActive(false);
    }

    public void CloseMenu()
    {
        for(int i = 0; i < windows.Length; i++)
        {
            windows[i].SetActive(false);
        }

        theMenu.SetActive(false);

        GameManager.instance.gameMenuOpen = false;

        itemCharChoiceMenu.SetActive(false);
    }

    public void OpenStatus()
    {
        UpdateMainStats();

        //оновлення інформації
        StatusChar(0);

        for(int i = 0; i < statusButtons.Length; i++)
        {
            statusButtons[i].SetActive(playerStats[i].gameObject.activeInHierarchy);
            statusButtons[i].GetComponentInChildren<Text>().text =
playerStats[i].charName;
        }
    }

    public void StatusChar(int selected)
    {
        statusName.text = playerStats[selected].charName;
        statusHP.text = "" + playerStats[selected].currentHP + "/" +
playerStats[selected].maxHP;
        statusMP.text = "" + playerStats[selected].currentMP + "/" +
playerStats[selected].maxMP;
        statusStr.text = playerStats[selected].strength.ToString();
        statusDef.text = playerStats[selected].defence.ToString();
        if(playerStats[selected].equippedWpn != "")
        {
            statusWpnEqpd.text = playerStats[selected].equippedWpn;
        }
        statusWpnPwr.text = playerStats[selected].wpnPwr.ToString();
        if (playerStats[selected].equippedArmr != "")
        {
            statusArmrEqpd.text = playerStats[selected].equippedArmr;
        }
        statusArmrPwr.text = playerStats[selected].armrPwr.ToString();
        statusExp.text =
(playerStats[selected].expToNextLevel[playerStats[selected].playerLevel] -
playerStats[selected].currentEXP).ToString();
        statusImage.sprite = playerStats[selected].charIamge;

    }

    public void ShowItems()
    {
        GameManager.instance.SortItems();
```

```
        for(int i = 0; i < itemButtons.Length; i++)
        {
            itemButtons[i].buttonValue = i;

            if(GameManager.instance.itemsHeld[i] != "")
            {
                itemButtons[i].buttonImage.gameObject.SetActive(true);
                itemButtons[i].buttonImage.sprite =
GameManager.instance.GetItemDetails(GameManager.instance.itemsHeld[i]).itemSprite;
                itemButtons[i].amountText.text =
GameManager.instance.numberOfItems[i].ToString();
            } else
            {
                itemButtons[i].buttonImage.gameObject.SetActive(false);
                itemButtons[i].amountText.text = "";
            }
        }
    }

    public void SelectItem(Item newItem)
    {
        activeItem = newItem;

        if(activeItem.isItem)
        {
            useButtonText.text = "Use";
        }

        if(activeItem.isWeapon || activeItem.isArmour)
        {
            useButtonText.text = "Equip";
        }

        itemName.text = activeItem.itemName;
        itemDescription.text = activeItem.description;
    }

    public void DiscardItem()
    {
        if(activeItem != null)
        {
            GameManager.instance.RemoveItem(activeItem.itemName);
        }
    }

    public void OpenItemCharChoice()
    {
        itemCharChoiceMenu.SetActive(true);

        for(int i = 0; i < itemCharChoiceNames.Length; i++)
        {
            itemCharChoiceNames[i].text = GameManager.instance.playerStats[i].charName;

itemCharChoiceNames[i].transform.parent.gameObject.SetActive(GameManager.instance.playerS
tats[i].gameObject.activeInHierarchy);
        }
    }

    public void CloseItemCharChoice()
    {
        itemCharChoiceMenu.SetActive(false);
    }

    public void UseItem(int selectChar)
    {
```

```
            activeItem.Use(selectChar);
            CloseItemCharChoice();
        }

        public void SaveGame()
        {
            GameManager.instance.SaveData();
            QuestManager.instance.SaveQuestData();
        }

        public void PlayButtonSound()
        {
            AudioManager.instance.PlaySFX(4);
        }

        public void QuitGame()
        {
            SceneManager.LoadScene(mainMenuName);

            Destroy(GameManager.instance.gameObject);
            Destroy(PlayerController.instance.gameObject);
            Destroy(AudioManager.instance.gameObject);
            Destroy(gameObject);
        }
    }
```