

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

«Візуальні моделі об'єктів в слайдскроллерах з
елементами платформінгу»

Виконав студент групи КН-19-2
ФЕСЮК Владислав Миколайович

Керівник роботи:
СУГОНЯК Інна Іванівна

Рецензент:
ПЕТРОСЯН Руслан Валерікович

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **ФЕСЮК Владислав Миколайович**

Керівник роботи: **СУГОНЯК Інна Іванівна**

Тема роботи: **«Візуальні моделі об'єктів в слайдскроллерах з елементами платформінгу»,**

затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №10/с**

Термін здачі закінченої роботи: **«23» червня 2023 р.**

Вихідні дані роботи: об'єктом дослідження є гра в жанрі сайдскроллер екшн з елементами платформінгу. Предметом дослідження є двигун Unity, платформа розміщення публікацій, її побудова на сучасних технологіях розробки.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Сугоняк І.І	02.02.2023р.	02.02.2023р.
2	Сугоняк І.І	03.04.2023р.	03.04.2023р.
3	Сугоняк І.І	15.05.2023р.	15.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2023 – 3 квітня 2023	Виконано
2	Проектування структури системи	3 квітня 2023 – 12 травня 2023	Виконано
3	Написання програмного коду	12 травня 2023 – 10 червня 2023	Виконано
4	Оформлення пояснювальної записки	12 червня 2023 – 16 червня 2023	Виконано

Здобувач вищої освіти **Владислав ФЕСЮК**
Керівник **Інна СУГОНЯК**

РЕФЕРАТ

Кваліфікаційна робота складається з проєтування комп'ютерної гри. Це проєкт сайд-скроллер екшн з елементами платформінгу. Пояснювальна записка до кваліфікаційної роботи складає 81 сторінок, 20 ілюстрацій.

Метою кваліфікаційної роботи є створення за допомогою ігрового двигуна Unity гри 2D стилі з елементами платформінгу. Яка буде містити сюжет, а на шляху вам потрібно буде прокачувати свого героя та приймат бійки з різними чудовиськами.

В проєкті визначено основні завдання які потрібно виконати. Були розібрані аналоги для розуміння кращого аналізу проєкту, наведена загальна структура проєкту, опис алгоритмів взаємодії модулів системи, опис системних структур тобто діаграми класів. А також були проведені деякі тестування систем.

КЛЮЧОВІ СЛОВА: UNITY, 2D, ГРА , ПЛАТФОРМІНГ.

					КН.КР.Б – 122 – 23 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		В.М.Фесюк			Візуальні моделі об'єктів в слайдскроллерах з елементами платформінгу Пояснювальна записка			Літ.	Арк.	Аркушів	
Керівник		І.І.СУГОНЯК								3	81
Рецензент		Р.В. ПЕТРОСЯН						Житомирська політехніка, група КН-19-2			
Н. контр.											
Зав. каф.		М.С. ГРАФ									

ABSTRACT

The graduation thesis of the bachelor, which was created by me, consists of the design of a computer game. This is a side-scroller action project with platforming elements. The explanatory note to the qualification work consists of 76 pages, 20 illustrations.

The aim of the undergraduate work is to create a 2D style game with platforming elements using the Unity game engine. Which will contain a plot, and on the way you will need to pump up your hero and accept fights with various monsters.

The main tasks to be performed are defined in the project. Analogues were analyzed for a better understanding of the project analysis, the general structure of the project was given, a description of the interaction algorithms of the system modules, a description of the system structures, i.e., class diagrams. And some system testing was also carried out.

KEY WORDS: UNITY, 2D, GAME, PLATFORMING

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

РЕФЕРАТ.....	3
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ТА ПЕРЕГЛЯД ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ГРИ У 2D СТИЛІ	10
1.1 Особливості реалізації візуальних об'єктів в 2D просторі.....	10
1.2 Аналіз аналогів програмного продукту.....	12
1.3 Вибір інструментів та засобів для проєкту	15
1.4. Постановка завдання.....	18
Висновки до розділу 1	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ТА СЕРЕДОВИЩА ОЗНАЙОМЛЕННЯ З ІГРОВИМИ МЕХАНІКАМИ.....	20
2.1 Опис структур у грі	20
2.2 Ознайомлення з конструюванням 2D левелінгу	26
2.3 Структура та складові елементи рівнів ігрового додатку	28
Висновки до розділу 2	28
РОЗДІЛ 3. ІНТЕРФЕЙС ТА РОБОТА З ІНТЕРФЕЙСОМ ОЗНАЙОМЛЕННЯ З УПРАВЛІННЯМ ГРИ.....	29
3.1. Реалізації візуальних моделей та алгоритмів їх взаємодії з користувачем	29
3.2 структура інтерфейсу. інтерфейс та порядок роботи.....	31
Висновок до розділу 3	37
ВИСНОВКИ	39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	40
ДОДАТОК.....	42

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ОС – операційна система

UI – інтерфейс користувача

ЛВЛ-(Level) рівень у грі

РС – персональний комп'ютер

АР – Adobe Photoshop

Souls-like - це жанр комп'ютерних ігор, що виник як результат популярності серії ігор Dark Souls, розробленої компанією FromSoftware.

Хаб - сленг в ігровій спільноті, який використовують для назви безпечної для гравця локації, де можна спілкуватися, купляти, продавати та займатися іншими речами в безпеці.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У нашому світі кожна людина так чи інакше грала у комп'ютерні ігри. Не є важливо чи це комп'ютерний геймінг чи мобільний або це приставка. Люди грають у них з різних причин: хтось на цьому раробляє кошти, хтось просто хоче весело провести щас з друзями так чином, а хтось просто хоче на декілька годин відволіктися від буденщини. Але чи задумуються звичайні люди як саме виникли ці ігри?

Хоча зараз 2D ігри мають меншу популярність аніж 3D, але саме 2D ігри були створеними першими у нашому світі. Вони були створені у 1970-х роках. Саме з них почалася еволюція розвитку ігор та через якихось 10 років почнуть з'являтися 3D ігри.

2D ігри це є класикою їх просто зрозуміти вони підходять для людей будьякого віку. Інтуїтивно зручний інтерфейс та відчуття легкої простоти поєднуються у неймовірну подорож тією чи іншою грою.

Оскільки є багато жанрів, тематик та тем для ігор вони є різними у кожній щось можна знайти собі до вподоби. Кожен раз коли розробник починає підбирати цю неймовірну комбінацію під кодовою назвою проектування гри на світ з'являється унікальний проект який знаходить своїх прихильників. Від хардкорних ігор де потрібно тренуватися та точно виконувати необхідні кроки щоб пройти якийсь шмат гри, вбити боса, розгадати загадку тощо до візуальних ігор. Візуальні ігри – це ігри які в яких основний геймплей створений на якийсь історії яку гравцеві потрібно або споглядати прослуховуючи аудіо репліки або читати текст історію з розкриттям сюжету персонажей і тому подібного.

2D ігри точно ввійшли в історію геймдеву. Є культові ігри у цьому жанрі наприклад "Pac-Man" або "Super Mario". Одна з культовіших ігор "Mortal Kombat" на початку створення була виключно у 2D стилістиці. Гравці могли рухалися лише вздовд

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

площини з простими графічними ефектами та анімаціями. Гра стала настільки культовою що розвивається і у наш час після стількох років. Остання гра у цій серії вийшла у 2019 році.

Також в 2D іграх серця геймерів підкорює проста графіка. З її простотою вона виглядає надзвичайно гарно. Піксельна графіка є гарною простою графікою з простим розуміння, що є перед очима у гравця.

Метою кваліфікаційної роботи є дослідження технологій проектування та реалізації візуальних моделей об'єктів в слайдскроллерах з елементами платформінгу та реалізація таких NameLess.

Були виставлені такі пріоритети при розробці які проговорюють наступні завдання:

- проведення аналізу різних ігор розроблених у 2D стилістиці;
- визначення та узагальнення структури системи;
- розробка дизайну гри
- розробка текстур у грі
- тестування ігрових левелів
- проєтування структурних складових та алгоритмів роботи

Предмет дослідження є процес проектування та реалізації візуальних моделей в ігрових додатках типу «слайдконтролер».

Об'єкт дослідження: технології реалізації візуальних моделей та їх взаємодії для різних типів задач ігрового додатку

В роботі використано методи об'єктно-орієнтованого проектування та програмування, методи візуальної розробки додатків, методи та технології підтримки взаємодії користувача з ігровими додатками

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ТА ПЕРЕГЛЯД ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ГРИ У 2D СТИЛІ

1.1 Особливості реалізації візуальних об'єктів в 2D просторі

Основною метою для створення 2D гри є ознайомлення з принципами та методиками роботи з ігровим двигуном Unity. Засвоєння принципів роботи у Unity є ключ до успіху створення проєкту. Створення такого перспективного проєкту підвищить професійні навички розробника та кращого розуміння геймдеву.

У кваліфікаційній роботі необхідно розробити комп'ютерну гру. Проєкт виконаний у 2D стилістиці з елементами платформингу. Для того щоб реалізувати поставлену задачу ми в декілька вирішуємо зробити:

Перед тим як починати роботу хочу більш розповісти про основи 2D. 2 є двовимірним простором або площиною, яка складається з двох осей: горизонтальної (ось X) та вертикальної (ось Y). У 2D-просторі об'єкти можуть рухатись лише вгору, вниз, вправо або вліво. Координати точок у 2D-просторі визначаються парами чисел (x, y), де x - значення по осі X, а y - значення по осі Y.

2D-графіка використовується для відображення об'єктів, таких як малюнки, фотографії, текст або ілюстрації на площині у даному випадку це буде створення текстур для гри. Багато комп'ютерних ігор, мультфільмів, веб-сторінок та графічних додатків використовують 2D-графіку для створення візуальних ефектів.

У 2D-графіці об'єкти можуть мати ширину, висоту, форму, кольори і текстури. Для їх відображення використовуються плоскі формати, такі як растрові (наприклад, JPEG, PNG) або векторні (наприклад, SVG). Растрові формати зберігають інформацію про кожен піксель зображення, тоді як векторні формати описують об'єкти у вигляді математичних формул, що дозволяє їх масштабувати без втрати якості.

Створення 2D текстур - це процес створення зображень, які можуть бути використані для візуального оформлення об'єктів у 2D-графіці.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Текстури використовуються для прикраси об'єктів, надання їм кольорів, текстур, деталей і реалістичного вигляду.

Особливості створення 2D текстур:

Роздільна здатність (резолюція): Роздільна здатність текстури визначається кількістю пікселів у зображенні. Вища роздільна здатність дозволяє отримати більш деталізовані текстури, але при цьому вимагає більше ресурсів комп'ютера для обробки. Були створені текстури у резолюції 32 x 32. Особливості такої резолюції у тому, що навідміну від 16 x 16 текстури мають більш гарний вид. Якщо взяти до прикладу 64 x 64 то в таких випадках або роблять текстури у більш гарній якості на вже створених 32 x 32 або створювати абсолютно інші текстури. Багато є ігор гарних та навіть культових у такій резолюції, для прикладу можна взяти Minecraft. Ця гра обожнювана мільйонами гравців різних вікових категорій по всій планеті. Базові текстури у грі написані саме у 32 x 32 резолюції.

1. Розробити дизайн користувацького інтерфейсу.

1.1 Створити загальну концепцію дизайну користувацького інтерфейсу.

1.2 За допомогою графічних редакторів придумати та створити графічні зображення для основних елементів інтерфейсу тобто (кнопок та панелей у головному меню і не тільки і тд).

1.3 За допомогою графічних редакторів створити цонцепт інтерфейсу та текстур для левелінгу (і не тільки):

1.3.1 Головне меню

1.3.2 Панель ігрових налаштувань

1.3.3 Панель що дозволить загрузити гру у останній збережений момент

1.3.4 Панель для створення нової гри

1.3.5 Панель ігрового користувацького інтерфейсу

2. Проектування основних модулів та механік у грі

2.1 Керування персонажем

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

- 2.2 Додання стрибка та дешу
- 2.3 Додання ігрових предметів
- 2.4 Додання перешкод на мапі
- 2.5 Створення ворожих чудовиськ

Результатом реалізації поставленого завдання яке ми для себе вибрали це є гра в стилі 2D Сайд-скроллер екшн з елементами платформінгу.

1.2 Аналіз аналогів програмного продукту

Аби краще зрозуміти матеріал потрібно для початку розібратися з тим, що є 2D впринципі. 2D означає двовимірний і використовується, щоб описати об'єкти, які мають лише дві виміри - довжину та ширину, але не мають висоти або глибини. Наприклад, зображення на папері, екрані комп'ютера чи на стіні будинку є 2D об'єктами. Геометричні фігури, такі як квадрат, трикутник і коло, також є двовимірними. Комп'ютерні програми, які створюють або обробляють зображення, також можуть працювати з 2D об'єктами.

У галузі відеоігор, 2D означає, що гра відтворюється у двовимірному просторі. Гравець бачить персонажів, об'єкти та фони у бічному профілі або зверху, як у класичних аркадах. Зазвичай, 2D гри засновані на простих движениях, які дозволяють швидко та легко створювати графіку та контент для гри.

2D ігри можуть бути різних жанрів, таких як платформери, файтинги, стратегії, головоломки, аркади та інші. Більшість з них мають простий геймплей та контролі, що дозволяє гравцям.

Для того щоб був матеріал на який можна опертись потрібно проаналізувати ринок та знайти кращих з кращих у цій сфері для того щоб підкреслити кращі рішення для себе та усунути гірші з них.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.1. Salt and Sanctuary

Перший конкурент якомого ми почали досліджувати це компанія Ska Studios і їхня гра Salt and Sanctuary. [18]

Сам продукт вийшов у 2016 році 15 травня. Гра виконана у 2D стилі, що підходить під нами вибрану тему для роботи. Протестувавши даний продукт пожу запевнити, що він є якісним. На мій погляд у гри є невеличкі недоліки: це деякі складні боси через обмежене переміщення та темний сеттінг, що може не всім сподобатися. Також у грі є маханіка рпг, яка у даному концепті може не всім сподобатися.

Дана гра є чудовим прикладом гарного продукту який слід проаналізувати та підчеркнути якісь ідея ненабравши поганих з даного продукту.

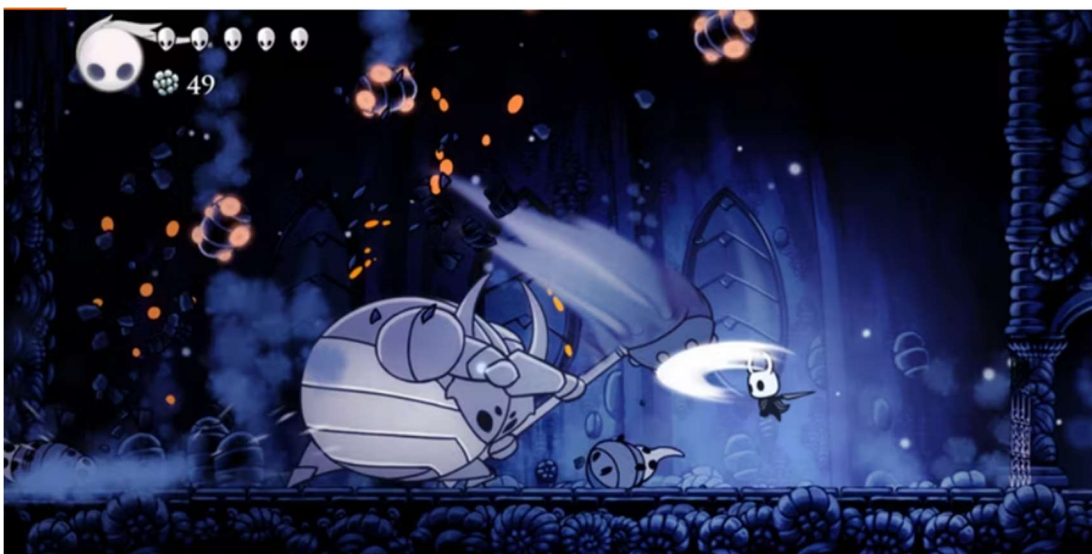


Рис. 1.2. Hollow Knight

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

Даний продукт випустила інді-компанія Team Cherry 24 лютого 2017 року. Ця гра стала швидко популярною через її сеттінг та унікальність. Даний продукт є чудовим прикладом не стандартної SoulsLike 2D гри. [19]

У грі дуже добре продумана генерація світу по якому цікаво бігати є елементи платформеру. Елементи сюжету є також дуже цікавими у даному продукту.

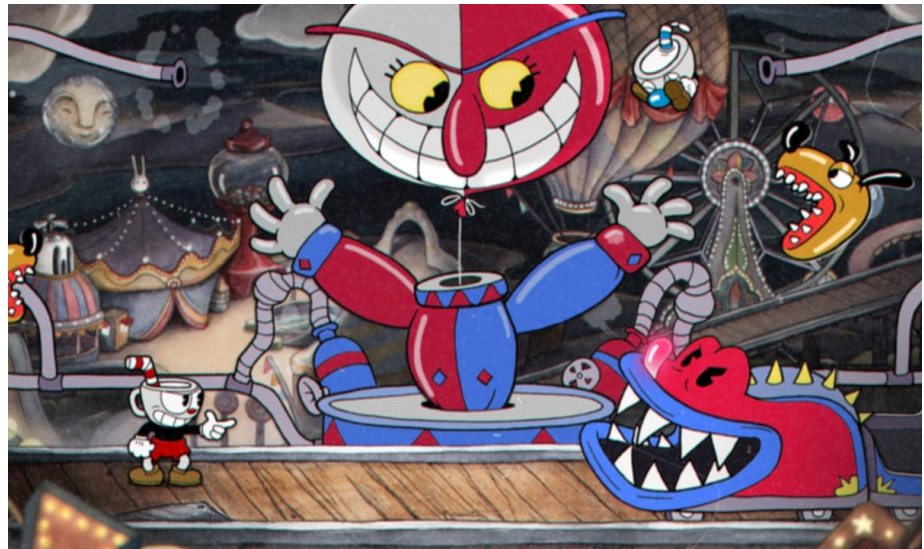


Рис. 1.3. Cuphead

Cuphead – це гра в жанрі run and gun та платформера, розроблена та видана канадською командою розробників StudioMDHR Entertainme. [20]

Анонс гри відбувся у 2013 році, а вихід для персональних комп'ютерів та Xbox One відбувся 29 вересня 2017 року. Керуючи персонажем на ім'я Чашка, гравець бореться із серією босів, щоб повернути борг дияволу. Гра була дуже натхненна роботами аніматорів 1930х років, таких як Fleischer Studios Макса Флейшера , і прагнула зберегти їхні сюрреалістичні якості.

Ігри таких типів менше важать ніж більшість 3D проектів, а також накладають менше навантаження на ваш комп'ютер. Не кожна людина може собі дозволити купити PC з останніми новинками для того щоб їхні PC витягували всі ігри на найвищій графіці у full hd якості. Тому в такі ігри де нема важкого навантаження на PC може пограти набагато більша кількість людей що у подальшому і стане плюсом для реклами, і з комерційної сторони це теж є плюсом, що гру зможе купити набагато більша кількість геймерів.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3 Вибір інструментів та засобів для проєкту

На сьогоднішній день інструментів для створення ігор є велика кількість і альтернатив з якими можна працювати. У кожного двигуна є свої переваги:

- Функціональність;
- Зручний інтерфейс;
- Платні та безкоштовні;

Але оскільки на мою суб'єктивну думку та якщо брати статистику то очевидний вибір двигуна для написання дипломної роботи ми вибираю двигун Unity оскільки цей двигун ми вивчали на протязі навчання в університеті. Одним з головних критерієм у виборі даного двигуна є те, що рушій дає можливість створення гри на різні платформи. Також Unity є безкоштовним двигуном з умовою, що проєкт не юде комерційним тому цей рушій є чудомив вибором у цьому випадку для написання дипломної роботи.

На даний момент у Unity є гарний конкурент Unreal Engine, але оскільки вище були перераховані плюси яких немає у цьому двигуні то було прийняте рішення ним не користуватися були проігноровані роботи з даним двигуном.

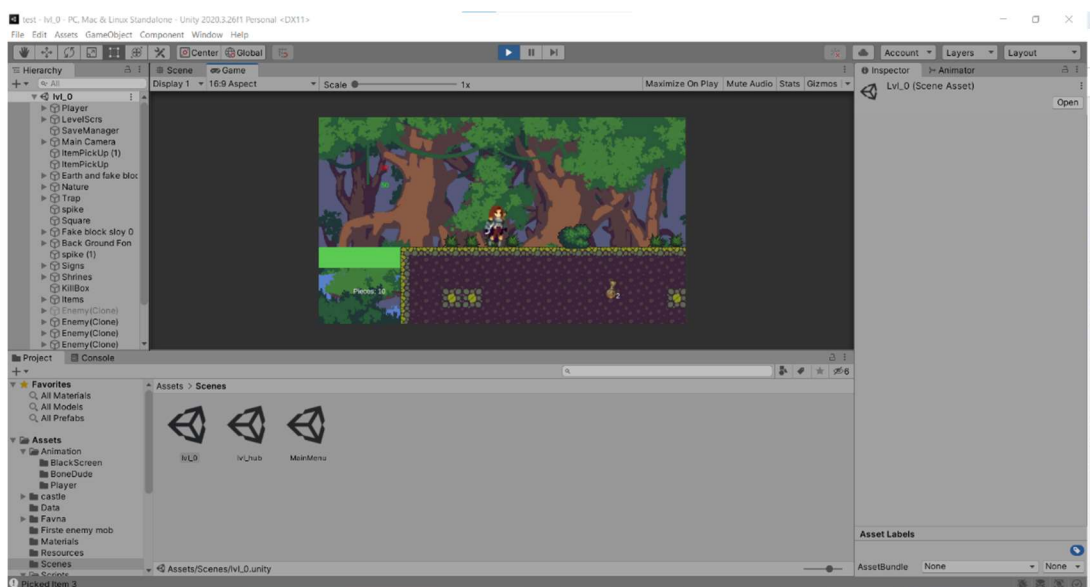


Рис. 1.4. Проєкт у Unity та інтерфейс

Сам Unity був випущений у далекому 2002 році. Все розпочалося коли Девід Геллер та Ніклас яске заснували компанію Unity Technologies. Вони

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

почали створювати відеоігри та відчули потребу у власному інструменті (якій став двигуном) для створення ігор. У 2005 році вийшла перша версія Unity.

Перейду до плюсів двигуна Unity з яким ми працюємо.

- По перше розробка у Unity ведеться мовою програмування C#. Ця мова не досить складна у розумінні та дозволяє увійти у розробку ігор.

- По друге перевага у кросплатформенності, один писаний код з деякими змінами можна імпортувати на інші платформи так як: PC, Mac, Android, IOS, Web та ігрові консолі. Це скорочує витрачений час для переносу продукту на інші платформи.

- По третє це перевага у доступній інформації по Unity. Є багато літератури яка у безкоштовну доступі. Якщо потрібно знайти якийсь конкретний приклад або функцію це не є проблемою. Також у Unity є гарне ком'юніті та форуми. Якщо людина початківець то вона може позадавати складні питання більш досвідченим людям у цій сфері праці.

Після обрання двигуна для написання дипломної роботи потрібно вибрати графічний редактор для створення текстур для самої гри.

Редакторів у наш час є багато: AP, Substance Designer, GIMP, Procedural Texture Generators, PixPlant. Для створення текстур для написання дипломної роботи був обраний графічний редактор AP. Adobe Photoshop: Photoshop є одним з найбільш популярних графічних редакторів та має широкий набір інструментів для створення текстур. Ця програма дозволяє редагувати та оптимізувати текстури, що робить її досить універсальною. Цей редактор є простим у користуванні та його легко засвоїти. Навіть утиїтивно вже можна самому розібратися у функція AP та зрозуміти як і що у ньому працює.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

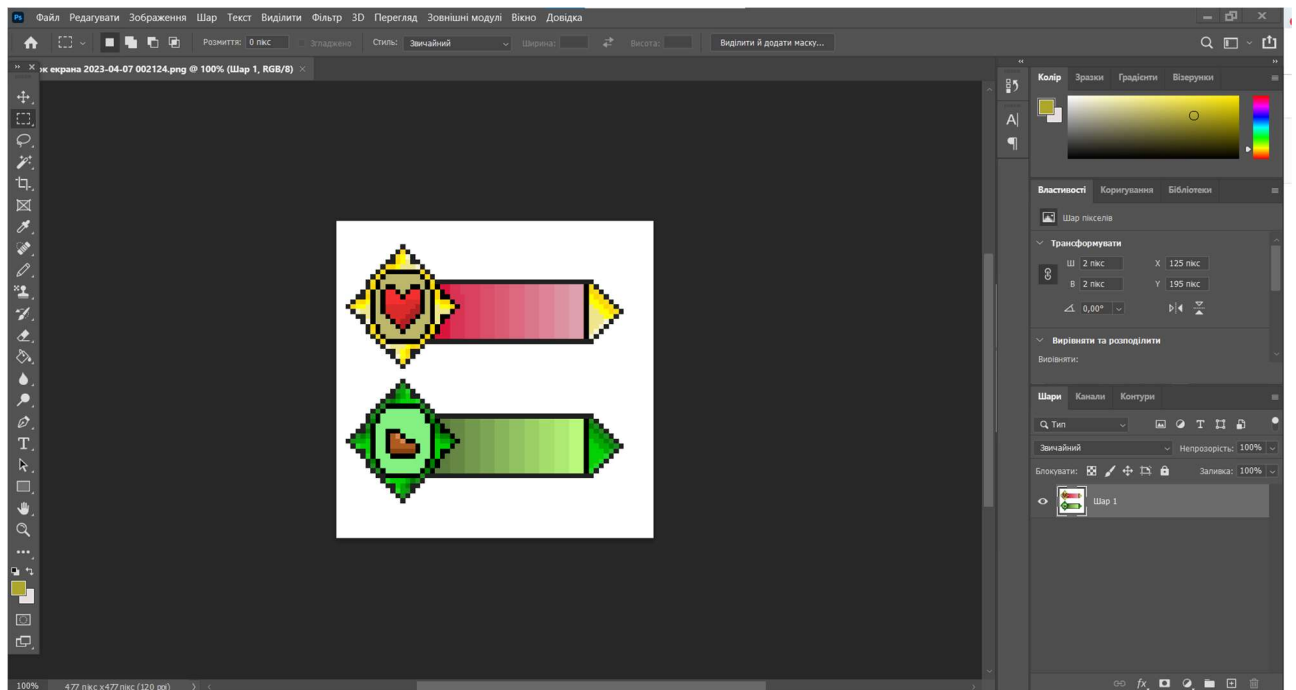


Рис. 1.5. Меню опрацювання АР та інтерфейс

Оскільки ми проходили курс по АР та засвоїв його на непоганому рівні тож і використав саме цей графічний редактор для роботи з кваліфікаційною роботою.

Перейду до переваг АР над іншими графічними радкторами:

- По перше Багатий набір інструментів: Adobe Photoshop має велику кількість інструментів для малювання, тонування та редагування зображень. Завдяки цьому користувачі можуть створювати складні та деталізовані текстури для ігор;
- По друге підтримка великих розмірів зображень: Adobe Photoshop підтримує обробку зображень високої роздільної здатності, що робить його ідеальним для створення текстур для ігор;
- По третє легкість використання: інтерфейс Adobe Photoshop є досить простим та інтуїтивно зрозумілим, що дозволяє користувачам легко вивчати програму та прискорює процес розробки;
- По четверте підтримка шарів: фотошоп дозволяє працювати з різними шарами зображень, що дозволяє користувачам редагувати окремі елементи зображення без впливу на інші;

- По п'яте і найголовніше можливість редагувати фотографії: фотошоп також дозволяє редагувати фотографії та зображення, що дозволяє розробникам створювати реалістичні та деталізовані текстури для ігор;

Під час створення текстур в мене виникли деякі проблеми з комбінуванням інструментів для текстур. З часом ці проблеми були вирішені.

1.4. Постановка завдання

Під час кваліфікаційної роботи були поставлені такі задачі:

Розробка системи переміщення: це створюється для того аби персонаж міг рухатись та переміщатися між рівнями, проводити складні стрибки.

Розробка системи бойових навичок: створюється бойова система для боротьби з монстрами на рівнях.

Розробка системи інвентаризації: створення зручного механізму для збирання предметів на різних локаціях та їх використання для полегшення геймплею або покращення головного героя.

Розробка системи діалогів з персонажами: надання гравцеві можливості спілкуватися з персонажами в головному хабі, які просуватимуть сюжет, розкриватимуть фонову історію та надаватимуть допомогу в подорожі.

Реалізація інших механік, які не входять в жодну з категорій: створення додаткових механік, що покращать геймплей гравця та quality of life features.

Висновки до розділу 1

Проведений аналіз продуктів та середовищ розробки комп'ютерних ігор визначив з якими програмами ми будемо працювати для написання кваліфікаційної роботи. Перегляд різних програм та аналогів дав чітке поняття, що головними функціями в таких програмах це зрозуміли та приємний інтерфейс у якому можна працювати, доступність самої платформи, кросплатформеність дасть змогу перенести свій проект на інші площадки, що скоротить час для перенесення продукту. Наявність гарної

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

літератури та дружнього ком'юніті дасть змогу звернутися до порад на
тяжкими питаннями які можуть виникнути під час проектування
кваліфікаційної роботи.

Визначеність з продуктом двигуна до гри та графічним редактором
аналізом ринку схожих продуктів та вирішення поставлених задач ми
можемо почати працю над написання дипломної роботи.

Визначеність з резолуціями текстур створенням текстур. Вирішенням
низки проблем з створенням текстур.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ТА СЕРЕДОВИЩА ОЗНАЙОМЛЕННЯ З ІГРОВИМИ МЕХАНІКАМИ

2.1 Опис структур у грі

У проєкті головне чітко усвідомлювати як саме буде працювати гра. Потрібно виставляти чіткі пріоритети та мати поняття як саме повинна працювати гра.

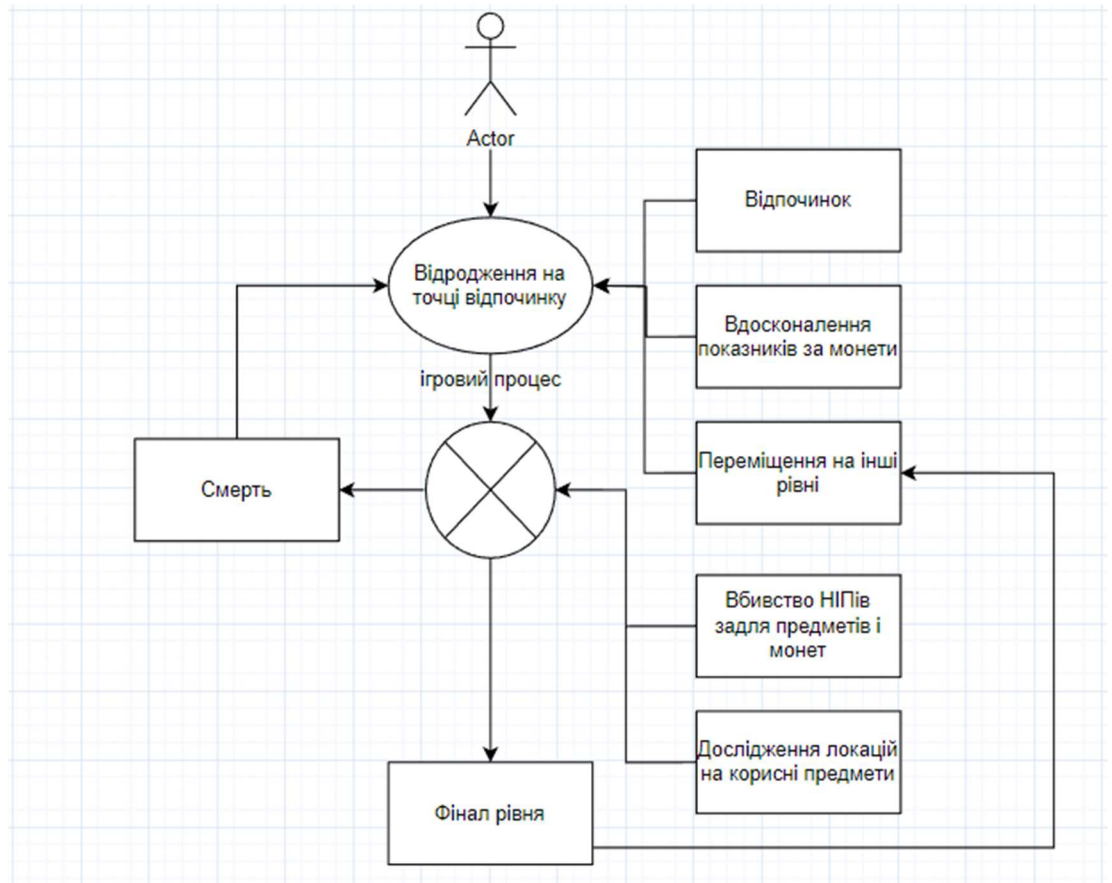


Рис. 2.1. Таблиця ігрового циклу у грі

Головним у 2D грі це рух. Буде швидким та маневреним може бути ключем у розробці грі такого типу. Швидко пересуватися також є важливим якщо вам потрібно битися з ворогами.

У даному проєкті користувач буде мати такі можливості:

1. Швидке переміщення по мапі;
2. Прижок для переміщення між платформами;
3. Деш він же ривок може бути корисним щоб переміститися на дальні відстані або через ворогів;

На цьому по переміщенню все тих компонентів вистачить для комфортного пересування по грі, а головне, що гра не буде здаватися повільною. Відчуття швидкого темпу. Зараз на діаграмі буде показано як це буде відтворено у самій грі.

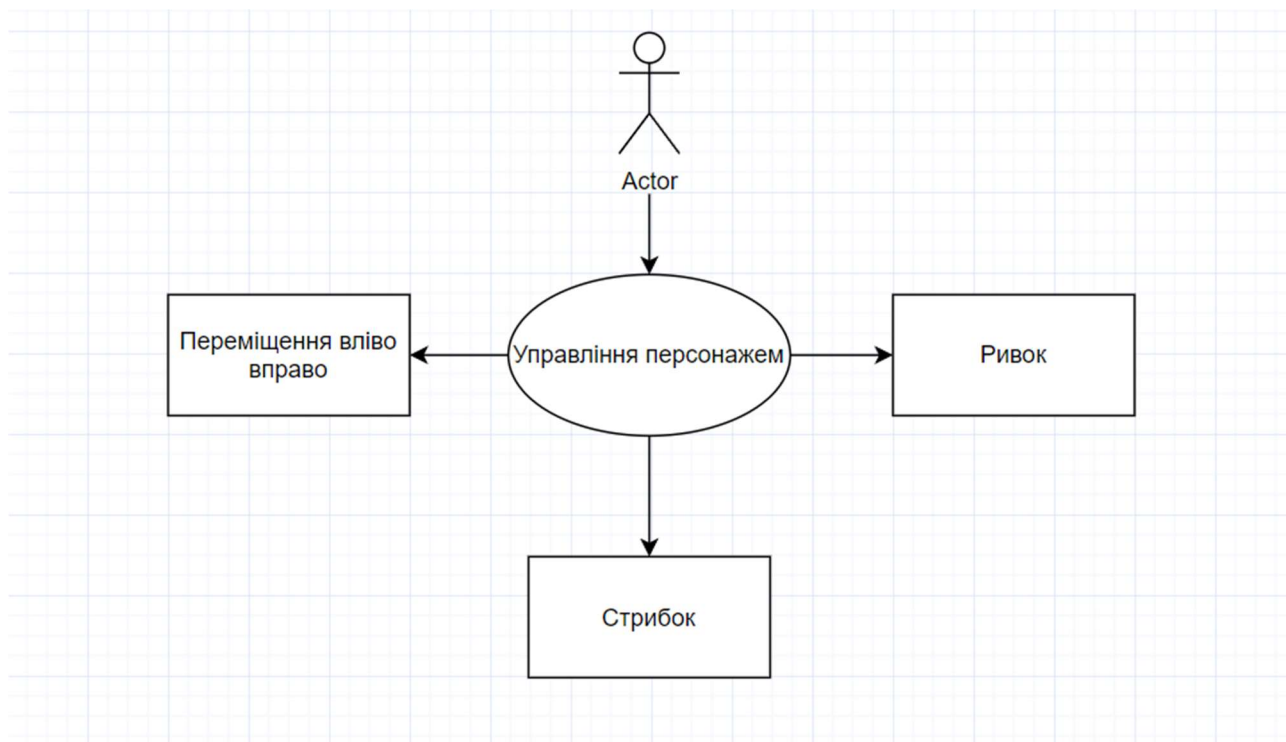


Рис. 2.2. Таблиця з механіками переміщення

Функціональні вимоги:

1. Геймплей гравець може переміщатися по рівню вліво та вправо. За допомогою прижку він може досягати нові місця. Також можна присісти, аби пролізти під чимось або уклонитись від удару. Можна швидко бігти.

- 1) AD – переміщення в ліво та право;
- 2) Shift – біг;
- 3) S – присяд;
- 4) W – прижок;
- 5) F – ривок;

2. Гра не знаходиться у статусі онлайн тому не потребує підключення у мережі інтернет.

Нефункціональні вимоги:

1. Сприйняття:

- Час для освоєння гри не потребує багато часу, а інтуїтивний геймплей допоможе швидко вникнути у процес гри.

- Оскільки гра не є у мережі то можна зберегтися та вийти у будьякий час. А оскільки гра не потребує щоденного входу в неї для отримання бонусів то гру можна на деякий час лишити і продовжити грати з того ж самого місця де ви залишилися останній раз.

- Інтерфейс не є нагромадженням тобто на екрані знаходяться лише ті частини UI які потрібні, вони не займають багато місця і весь геймплей добре видно на місці.

2. Продуктивність. Гра працює належним чином та не просідає під час гри, а фпс у грі стабільно тримає потрібну кількість кадрів без перебоїв.

Бойова механіка:

Оскільки гра в жанрі souls-like, бойова система зосереджена на ухиленнях та парирування. Гравець зможе завдавати легкі та сильні удари та комбінувати їх. Також доступний блок, який вчасно нажавши, можна відбити атаку повністю

Як працює: ЛКМ - легкий удар. Зажата ЛКМ - важкий удар

ПКМ - Блок(Парирування) shift – ухилення

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

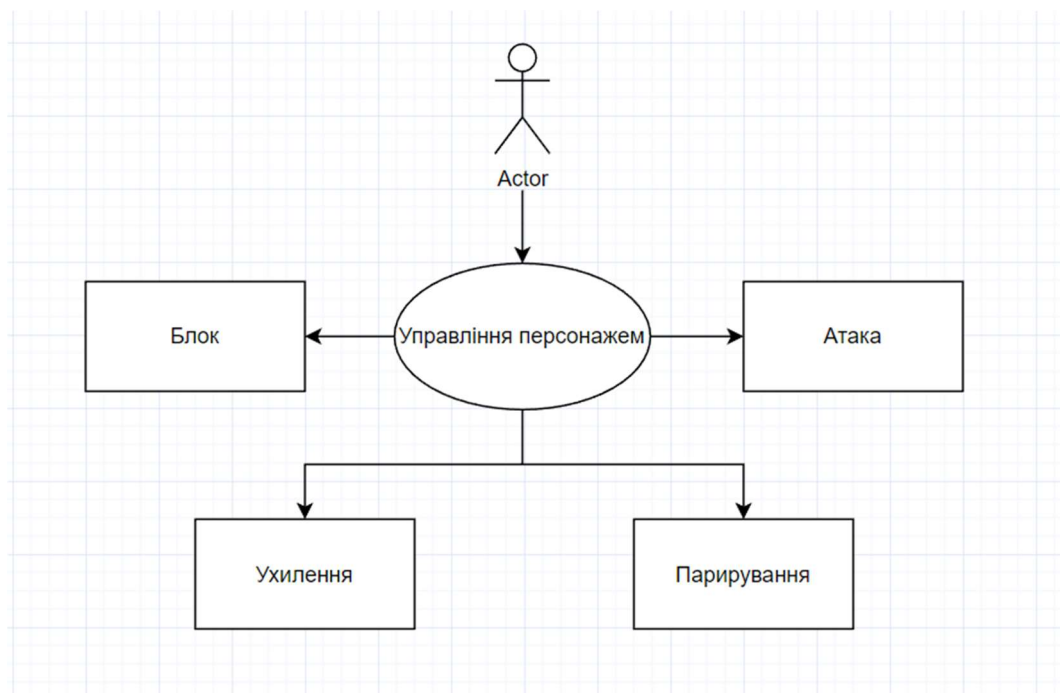


Рис. 2.3. Таблиця з бойовими механіками

Маханіка пов'язана з ворогами:

Вороги будуть поділені на класи, в залежності від яких вони будуть полювати на ГГ. Головна ціль ворогів - змусити гравця рухатися і не бути завжди в обороні.

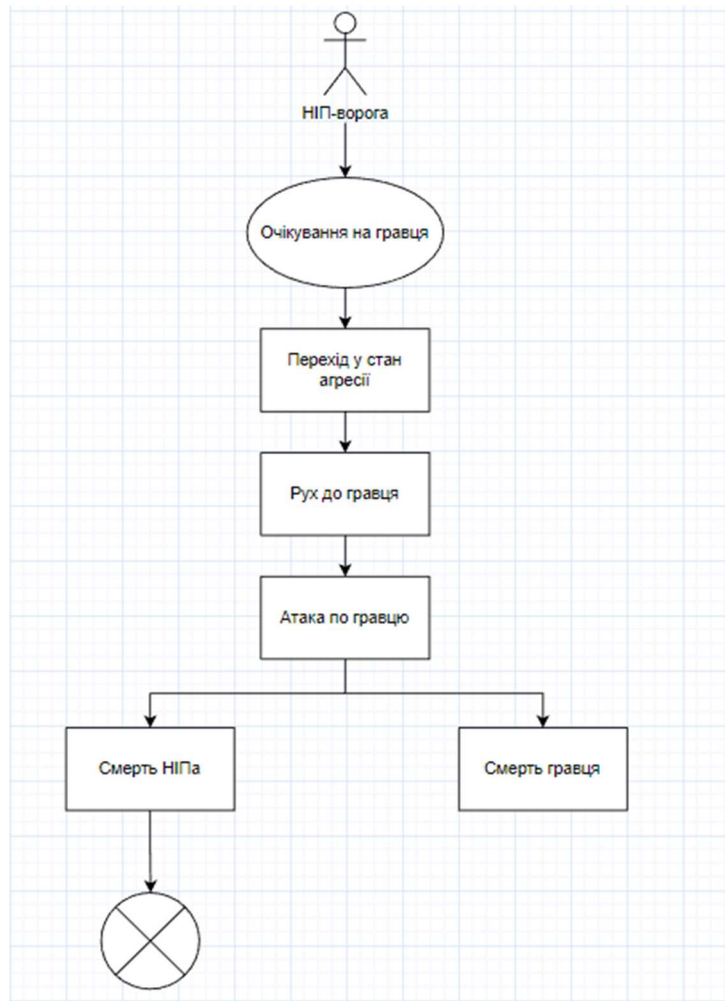


Рис. 2.4. Аналіз поведінки ворожого НІП у грі
Бойові механіки:

Здоров'я можна відновити відпочивши у відведеному місці, наповнивши склянки лікувальним зіллям, які вже в бою можна буде використати.

Після смерті головний герой повертається на місце останнього відпочинку та втрачає всі отримані ресурси з ворогів(часточки). Вороги ж заново відновлюються у зазначеному їм місцях.

Діалоги:

Знаходячись у мирній локації, гравець може відвідати декілька місць: таверна, крамниця, кузня. Там він може поспілкуватися з НІП-ами та дізнатися інформацію про лор світу та зачіпки до його помсти.

Покращення:

Перемагаючи ворогів, гравець буде отримувати часточки, які можна буде витратити на покращення характеристик. Після смерті всі невикористані часточки буде втрачено.

Перемагаючи босів, гравець отримуватиме нові здібності, які допоможуть йому в подальшому проходженні, такі як подвійний прижок, подвійне ухилення і т.д.

Звуки та музика у грі:

Звуковий стиль грає важливу роль у грі. Оскільки сеттінг це середньовіччя, то і звуки необхідні такі, щоб підтримувати атмосферу та поринення гравця в цей світ. Будуть переважати духові та оркестричні мотиви, а також задній план як і у справжніх селах та міста середньовіччя: звуки коней, цокіт чобот і т.д.

Звукові ефекти:

- Інтерфейс:
- Впливаюче вікно
- Підтвердження в вікні
- Відхилення в вікні
- Пауза гри
- Назад в меню
- Обрати пункт в меню

Спец. ефекти:

- Звуки удару
- Звуки блоку
- Звуки парирування
- Звуки переміщення
- Звуки бігу
- Звуки прижку
- Звуки присідання

Музика:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Оскільки гра знаходиться в сеттінгу середньовіччя, то і музика повинна підходити. Підійдуть треки в жанрі оркестрової та духової. Музика під час бою буде швидка та агресивна, щоб змусити гравця грати в активний наступ та показати всю рішучість головного героя.

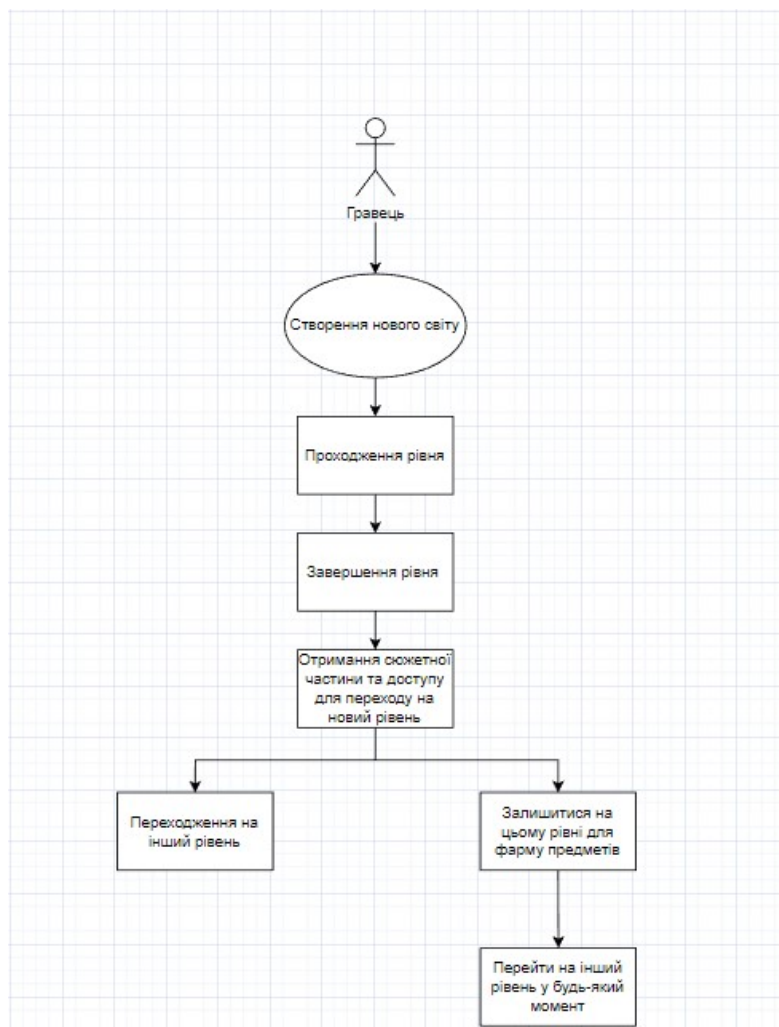


Рис. 2.5. Загальна схема структури гри

2.2 Ознайомлення з конструюванням 2D левелінгу

Створення візуальної складової ігрового додатку включає ряд етапів та процесів. Ось загальна схема, яка описує основні кроки в процесі розробки візуальної складової ігрового додатку:

Проектування концепції: Перш ніж розпочати розробку візуальної складової, важливо визначити загальну концепцію гри. Це включає визначення жанру, цільової аудиторії, характеристик геймплею та інших ключових аспектів гри.

Створення концепт-артів: Після визначення концепції гри створюються концепт-арти, що демонструють зовнішній вигляд персонажів, об'єктів, локацій та інших елементів гри. Це допомагає уточнити візуальний стиль та забезпечити єдність дизайну. Для того щоб виконати дану функцію потрібно виконати кілька кроків:

Розробка 2D-графіки: У випадку, коли гра використовує 2D-графіку, анімацію, фони та інші елементи створюються у графічних редакторах, наприклад, Adobe Photoshop або Illustrator.

Анімація: Якщо гра містить рухомі об'єкти або персонажів, їх потрібно анімувати це можна зроби у двигуні Unity.

- Проектування рівнів: Перший етап - це створення концепту рівнів вашої гри. Ви можете використовувати папір і олівець для створення скетчів або спеціальні програми для проектування рівнів. На цьому етапі важливо розробити дизайн рівня, включаючи розташування перешкод, пасток, ворогів та інших ігрових елементів.

- Створення артів: Для створення графічних елементів рівня, таких як тло, платформи, об'єкти, персонажі і т.д., використовуються графічні програми, такі як Photoshop або Illustrator. Важливо розробити візуальний стиль, який відповідає загальному настрою гри.

- Розміщення об'єктів: На цьому етапі розміщуються об'єкти рівня, такі як платформи, перешкоди, вороги та інші елементи гри. Це виконується за допомогою спеціальних інструментів редактора рівнів або шляхом програмування.

- Додавання логіки: Після розміщення об'єктів рівня потрібно налаштувати їх поведінку. Це включає програмування руху ворогів, реагування на взаємодію з гравцем, визначення логіки зіткнень та інші аспекти гри.

- Тестування та налагодження: Після завершення побудови рівня важливо протестувати його, щоб переконатися, що він функціонує.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3 Структура та складові елементи рівнів ігрового додатку

Структура та складові елементи рівнів ігрового додатку можуть варіюватися в залежності від конкретної гри та її жанру. Терейн або локація: Це основна область, в якій відбувається гра. У випадку кваліфікаційної роботи вона виконана у 2D стилі. Терейн може включати різноманітні об'єкти, такі як пейзажі, будівлі, вода, рослини тощо. Також потрібно налаштувати камеру. Вона контролює положення та огляд, що дозволяє гравцю бачити поточну область рівня.

Щодо створення рівнів вони можуть містити конкретні задачі або мети, які гравець повинен виконати для продовження гри. Це можуть бути завдання, які потрібно виконати, вороги, яких потрібно перемогти, пазли, які потрібно вирішити або будь-які інші завдання, які спрямовують гравця

Висновки до розділу 2

У даному розділі було розглянуто систему гри. Покроково ознайомилися з механіками гравця на таблицях та побачили логіку поведінки ворогів у грі. Також ознайомився з основами левел дизайну покроково розписав проектування левелінгу. Після засвоєння даних приміток почав конструювати рівні для гри. Розібрали управління гри для чіткого контролювання персонажем у грі і розумінням того як він буде пересуватися у грі. Конкретний розпис кожної механіки дає чітке розуміння того що саме буде у грі та як воно буде працювати безпосередньо. Ознайомився з звуковими ефектами. Поставлення чітких звуків та атмосферної музики є важливим ігровим елементом, який допоможе пірнути у вир даної гри. Ознайомилися з структурою та елементами ігрового додатку. Коротко розповів про терейн та камери показав їхню невід'ємну роль у проекті.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА РОБОТА З ІНТЕРФЕЙСОМ ОЗНАЙОМЛЕННЯ З УПРАВЛІННЯМ ГРИ

3.1. Реалізації візуальних моделей та алгоритмів їх взаємодії з користувачем

Скрипти у геймдеві можуть взаємодіяти з інтерфейсом у грі для забезпечення різних функцій і поведінки. Одним з основних способів взаємодії скриптів з інтерфейсом є обробка подій. Наприклад, скрипт може бути пов'язаний з кнопкою в меню гри і виконувати певні дії, коли користувач натискає на цю кнопку. Крім того, скрипти можуть змінювати візуальний вигляд інтерфейсу.

Робота зі скриптами є важливою складовою процесу створення відеоігор. Скрипти в геймдеві використовуються для програмування різноманітних функцій і поведінки об'єктів в грі. Також скрипти є необхідною частиною для підв'язки з працею над інтерфейсом у грі.

Процес конектування скриптів з інтерфейсом в геймдеві залежно від двигуна на якому створюється гра, але основні кроки можуть бути наступними:

Створення інтерфейсу: Спочатку потрібно створити елементи інтерфейсу, такі як кнопки, текстові поля, спливаючі вікна тощо. Це можна зробити за допомогою редактора інтерфейсу.

Створення скрипту: Потім необхідно створити скрипт, який буде відповідати за функціональність і взаємодію з елементами інтерфейсу.

Прив'язка скрипту до елементів інтерфейсу: Для того, щоб скрипт міг взаємодіяти з елементами інтерфейсу, необхідно здійснити прив'язку скрипту до відповідних елементів. Це можна зробити шляхом присвоєння скрипту відповідних компонентів або обробників подій елементам інтерфейсу. Зазвичай цей процес виконується за допомогою інтерфейсу редактора, де можна просто перетягнути скрипт на відповідний елемент.

Реалізація функціональності: Після прив'язки скрипту до елементів інтерфейсу можна реалізувати необхідну функціональність. Це може

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

включати обробку подій, зміну вигляду елементів, взаємодію з іншими скриптами та інше. В залежності від геймворку або движка, ви можете мати доступ до спеціальних функцій.

Перерахувавши кроки дій та алгоритми наведені вище можна привести у приклад скрипт CameraFollow. Цей скрипт виконує функцію слідування камери за персонажем(головним героєм гри) і взаємодіє з інтерфейсом гри. У скрипті є куча змінних які взаємодіють так чи інакше з інтерфейсом та анімаціями у грі. Додаток [А]

За анімації у грі відповідає перемінна public Animator anim. Завдяки цій перемінній зберігається посилання на компонент Animator завдяки якому створюються анімації і в подальшому з'єднуються між собою та добавляються у скрипт. Додаток [А]

Для того щоб активувати певний тригер у грі використовується метод SetTrigger. Даний метод у скрипті використовується для активації переходу між різними станами анімацій. Наприклад анімація смерті та відродження або анімація прижку та ривок(деш). Точний вигляд і дія тригера в анімації залежить від налаштувань і конфігурацій компонента Animator та анімацій, які були створені для використання в грі. Додаток [А]

Багато перемінних працюють безпосередньо з функціями самої камери як і згадувалось вище, але невідмінною частиною є і використання інтерфейсу. Якщо з анімаціями все стало зрозуміло та автором вказані прості кроки поетапного використання анімацій. З інтерфейсом також пов'язані процеси взяти до прикладу private void PauseStart та public void PauseStop. Перша активує режим паузи а інша знімає його. Здавалось при чому тут інтерфейс, але коли вмикається пауза то на екрані спливає вікно з відповідними кнопками. Для того щоб відбулося якесь дійство потрібно натиснути на кнопку, але для цього потрібно створити кнопки для того щоб був візуал з яким скрипт буде працювати. У іншому випадку скрипт буде, але ми не побачимо як він спрацьовує бо немає візуалу і чи він працює взагалі.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Також для роботи з інтерфейсом використовується метод `changeCanvas` у даному скрипті використовується для зміни відображення елементів інтерфейсу гри. Він приймає певний параметр `i`, який вказує на тип інтерфейсу, що потрібно відобразити.

Залежно від переданого значення `i`, метод `changeCanvas` змінює стан елементів інтерфейсу, роблячи деякі видимими або невидимими. Основна логіка методу включає в собі `switch`-конструкцію, де для кожного значення `i` задається відповідний стан елементів інтерфейсу.

Наприклад, якщо `i` дорівнює 0, метод `changeCanvas` вимикає всі елементи, крім елементу "Default", що відповідає типовому інтерфейсу гри. А якщо `i` дорівнює 1, то вимикається "Default" і вмикається "Shrine", який відображає інтерфейс особливого місця у грі. Додаток [A]

Використовуючи метод `changeCanvas`, можна змінювати відображення різних елементів інтерфейсу в залежності від поточного стану гри або взаємодії з іншими скриптами.

Також у проекті є невеликі скрипти які відповідають за конкретні кнопки. Якщо взяти до прикладу скрипти які закінчуються на `Button` наприклад `TravelButton` або `Panel` тревел панел такі скрипти відповідають за відкриття окремих вікон та взаємодії з якимось дійством. Під час виконання цих скриптів коли відкриваються певні вікна вони взаємодіють з інтерфейсом безпосередньо. Нажавши на одну чи іншу наперед намальовану частину інтерфейсу спрацьовує скрипт та змінюється інтерфейс відповідно до того, що саме повинно візуалізуватися на екрані.

3.2 структура інтерфейсу. інтерфейс та порядок роботи

В даному розділі буде ознайомлення з інтерфейсом у грі. Та взаємодію гравця з ним. Гра буде працювати на таких мінімальних конфігураціях РС.

ОС: Windows 10+

Оперативна пам'ять: 2 GB

Відеокарта: Open GL 3.2+ Compliant

Місце на диску: 2 GB доступного місця

Додаткові примітки: 1080p, 16:9

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

За такими конфігураціями гра буде працювати у стабільних 30-60 кадрах. Цього достатньо для такого типу гри аби пограти з комфортом, але чим більше тим краще.

Після запуску гри на екрані вашого комп'ютера вплине таке вікно з такими типа кнопок:

Нова гра:

Продовжити гру:

Управління:

Вийти з гри:

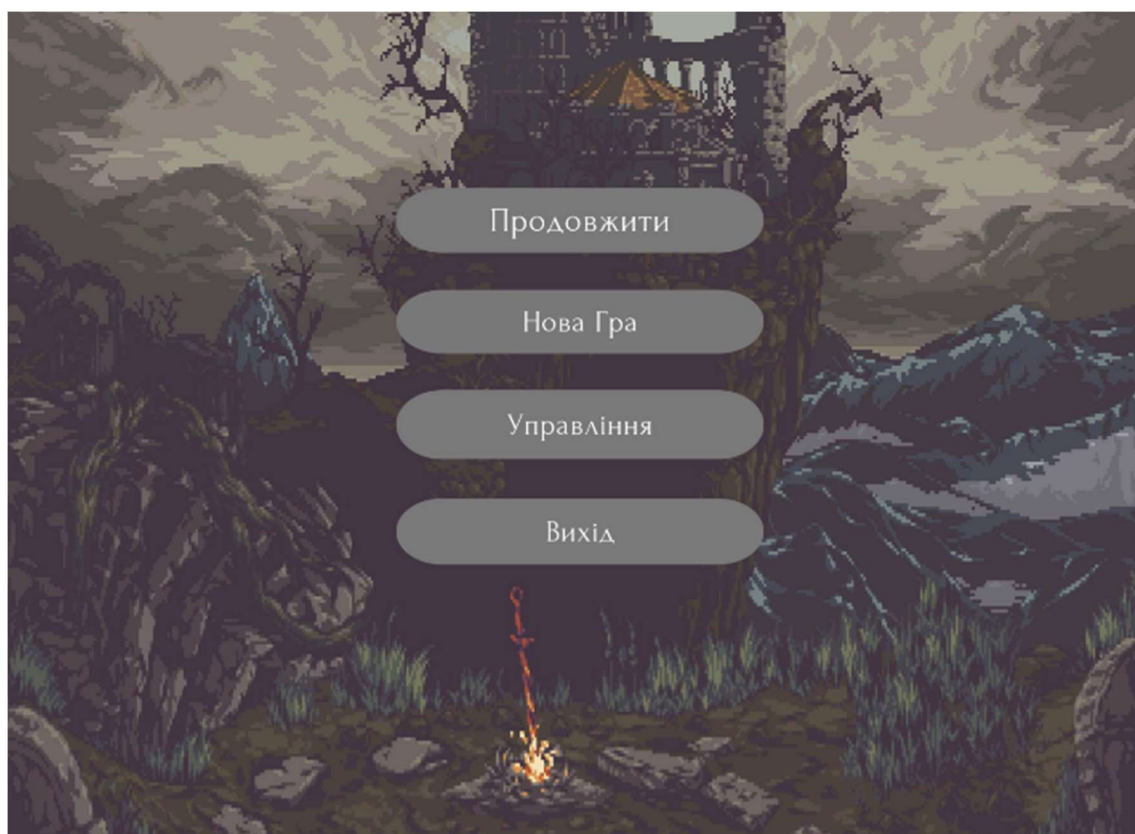


Рис. 3.1. Головне меню гри

Кнопка «Нова гра» відповідає за створення ігрового профілю для початку нової гри. Така функція дозволить на вашому акаунті мати декілька створених профілів для гри декількох людей, що є дуже зручним, якщо ви граєте у гру в двох або більше людей.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

Кнопка «Продовжити гру» відповідає за загруження ігрових даних та процесу гри з останнього вашого місця збереження гри. Для того щоб зберегти гру потрібно зробити відповідне дійство у спеціальному місті у грі, після виконання даної процедури гра буде зберігатися.

Кнопка «Управління» ця кнопка допоможе вам зайти у відповідні налаштування у самій грі для ознайомлення з гарячими кнопками. Не всім людям під силу зразу інтуїтивно керувати або бувають випадку коли забуваєш якусь клавішу.

Кнопка «Вихід з гри» допоможе безпечним чином залишити гру та вийти до робочого столу.

У середині грі також є інтерфейс завдяки якому гравець буде більш орієнтуватися і розбиратися у самій грі.

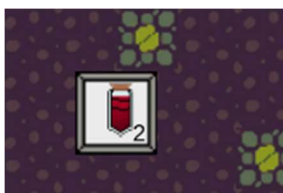


Рис. 3.2. Вигляд інтерфейсу пляшки зі здоров'ям

У грі є спеціальне місце на якому гравець може побачити свої розхідні матеріали наприклад пляшки зі здоров'ям. У цьому полі можуть бути і інші розхідні матеріали. Це вікно допоможе гравцю зорієнтуватися і чітко визначати кількість ресурсів яку він має на дану хвилину.

Також у грі є шкала здоров'я та енергії. Шкала здоров'я показує кількість очок здоров'я у вашого персонажа, якщо у вашого персонажа кількість очок здоров'я опуститься до нуля, то ваш персонаж помирає. Шкала енергії є для того щоб гравець міг розрахувати кількість прижків та ривків які він може використати у конкретний час з поточною кількістю енергії.



Рис. 3.3. Вигляд шкали здоров'я та енергії

Також у грі є можливість нанесення дистанційної шкоди супернику. Для цього потрібно натиснути клавішу Q для того щоб метнути кинжал. Після натискання клавіші ви втрачаєте один кинжал та запускаєте його у напрямку свого зору.



Рис. 3.4. Вигляд інтерфейсу зміна пляшки здоров'я на кинжал

Для переходу між локаціями у грі створені переходи у вигляді спеціальних алтарів. Така зміна локацій вже устаканилася у цьому жанрі і вважається нормою. Започаткували цю традицію ігри серії «Souls-like» а саме «Demon Souls».

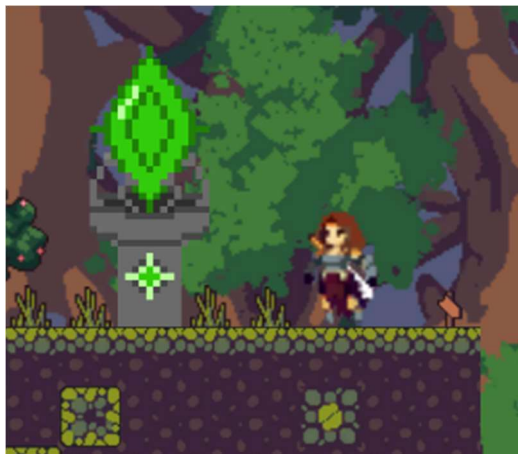


Рис. 3.5. Вигляд тотему для відпочинку прокачки персонажа та переміщення

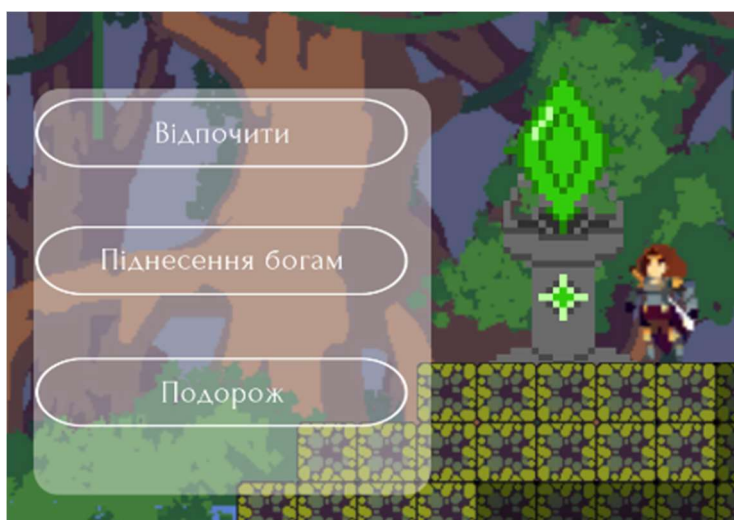
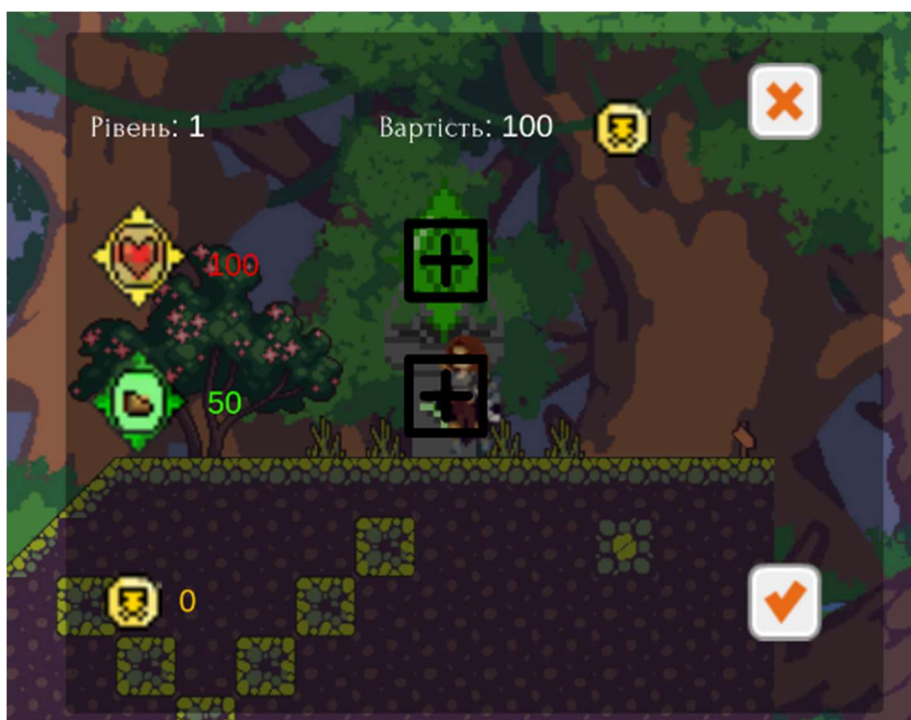


Рис. 3.6. Вигляд інтерфейсу вибору дій у тотемі



Змн.	Арк.	№ докум.	Підпис	Дата

Рис. 3.7. Вигляд інтерфейсу з меню прокачки героя

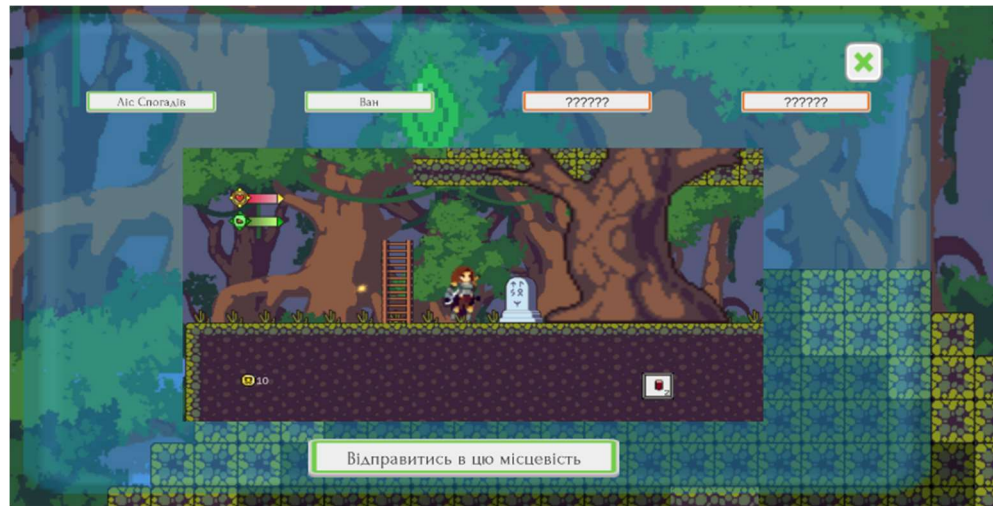


Рис. 3.8. Вигляд інтерфейсу під час подорожі

Між тим коли персонаж переноситься або грузиться до самої гри то в грі є екран переходу між локаціями та загрузки ігрової сесії.



Рис. 3.9. Вигляд екрану загрузки

Під час екрану загрузки будуть відображатися частинки історії самої гри, а атмосферні арти ще більше поглинуть вас у саму гру. Для того щоб почати гру після екрану загрузки потрібно натиснути клавішу Enter.

Якщо ви випадкову збили загрузку, але боїтесь за процес гри то це не є бідою адже у грі є пауза. Можна спокійно поставити гру на паузу та піти по своїм справами. Щоб зробити це дійство потрібно надати клавішу англійську букву пі «р».



Рис. 3.10. Вигляд меню під час паузи

Якщо ви забули якусь клавішу то можна без проблем зайти в управління гри та подивитися всі гарячі клавіші які є.

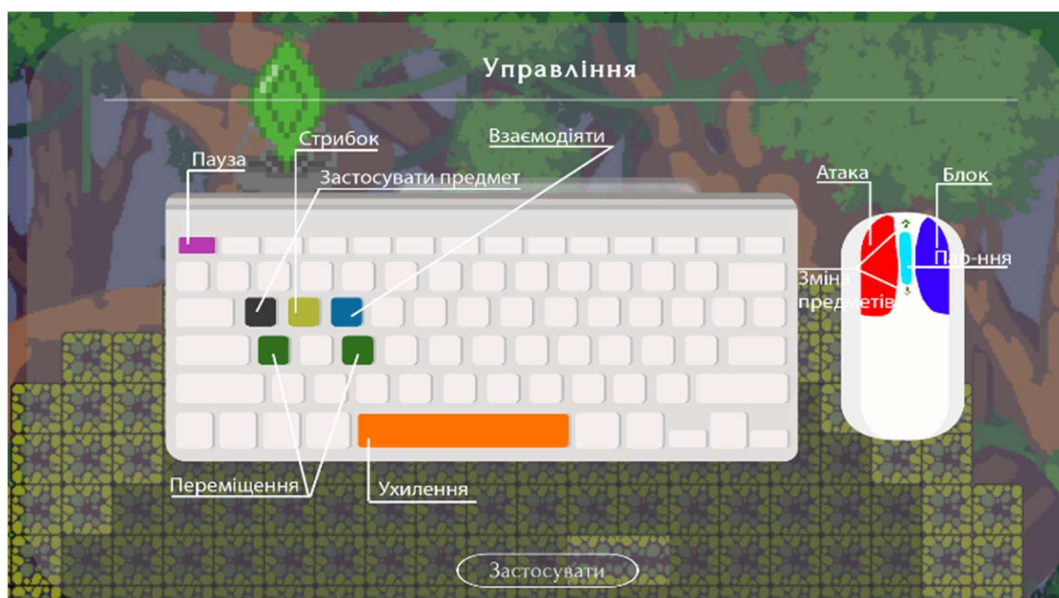


Рис. 3.11. Інтерфейс управління гарячими клавішами

Висновок до розділу 3

В даному розділі було візуальне ознайомлення з інтерфейсом у грі. На рисунках були показані кожні частини інтерфейсу окремо. Розбір кнопок та їх конфігурації у грі. У грі було проведене тестування додатку для усунення багів та перевірки маршруту гравця для комфортного переміщення у самій грі. Реалізовані усі механіки які були заплановані на поетапному плані у грі. Автором було проведене ознайомлення з працею скриптів та інтерфейсу у проекті. Розписане поєднання методів та їхнього використання та поєднанням з інтерфейсом у проекті. Був розбір елементів скрипта які

взаємодіяли з створенням інтерфейсу. Була показана чітка взаємодія між скриптами та інтерфейсом. Пояснені основні елементи контактування з інтерфейсом та їхню роль у проекті.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час написання дипломної роботи був спроектований та реалізований ігровий проект візуальні моделі об'єктів в слайдскроллерах з елементами платформінгу. Для того щоб реалізувати поетапно запланований проект було вирішено покроково наступні задачі.

Проведений аналіз ігрового ринку та конкурентів який допоміг визначитися з аспектом гри, жанром гри, механіками та текстурами продумати та вистроїти маршрути для рівнів. Огляд різних типів UI також допоміг вирішити певні питання які виникали під час створення UI для власного проекту.

Я провів аналіз функціональності у самій грі. Перевірив здатність проходження заданого маршруту. Переконався, що маршрут прокладений коректно і гравець зможе успішно прийти до цілі.

Була здійснена перевірка усіх базових механік у грі. Усі механіки коректно пройшли перевірку та працюють коректно.

Після обрання жанру гри була вибрана програма для створення текстур у AR, після цього пішов процес малювання текстур для проекту. Усі текстури пройшли перевірку на коректність, це було переглянуто для того випадку, щоб гравець не провалився ніде у текстури.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Форум по створюванню 2D ігор [Електронний ресурс] – Режим доступу до ресурсу: <https://unity.com/2d-solution-guide>.
2. Документація по створенню 2D ігор [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/Quickstart2DCreate.html>.
3. Методи створення власних текстур у Photoshop [Електронний ресурс] – Режим доступу до ресурсу: <https://www.smashingmagazine.com/2014/07/creating-custom-textures-photoshop-techniques/>.
4. Як створювати піксельні текстури [Електронний ресурс] – Режим доступу до ресурсу: <https://www.adobe.com/creativecloud/design/discover/pixel-art.html>.
5. Онлайн редактор для створення піксельних текстур [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/pixel-grimoire/how-to-start-making-pixel-art-2d1e31a5ceab>.
6. Робота з Tilemap 2D unity [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/class-Tilemap.html>.
7. Інструкції щодо користування Tilemap [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.unity.com/tutorial/introduction-to-tilemaps#5f35935dedbc2a0894536cff>.
8. Оптимізація ресурсу за допомогою Tilemap [Електронний ресурс] – Режим доступу до ресурсу: <https://unity.com/how-to/optimize-performance-2d-games-unity-tilemap>.
9. "Unity in Action: Multiplatform Game Development in C#" автора Джошуа Міддлетона 408 с.
10. "Mastering Unity 2D Game Development" автора Саймона Чіля та Саймона Ламберта 590 с.
11. Книга "Unity Game Development Cookbook" автора Паріса Батела має приблизно 624 с.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

12. Книга "Learning C# by Developing Games with Unity" автора Харрісона Фердінандуса має приблизно 606 с.
13. DotNetPearls [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.dotnetperls.com/between-before-after>
14. C# Documentation [Електронний ресурс] - Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/dotnet/csharp/>
15. Souls-Like Games [Електронний ресурс] - Режим доступу до ресурсу:
<https://en.wikipedia.org/wiki/Soulslike>
16. Unity discussions [Електронний ресурс] - Режим доступу до ресурсу:
<https://discussions.unity.com/>
17. Steam [Електронний ресурс] – Режим доступу до ресурсу:
https://store.steampowered.com/app/283640/Salt_and_Sanctuary/
18. Steam [Електронний ресурс] – Режим доступу до ресурсу:
<https://store.steampowered.com/app/268910/Cuphead/>
19. Steam [Електронний ресурс] – Режим доступу до ресурсу:
https://store.steampowered.com/app/367520/Hollow_Knight/
20. Документація Unity [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.unity3d.com>

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

Скрипт CameraFollow додаток [A]

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using TMPro;
using UnityEngine;
using UnityEngine.PlayerLoop;
using UnityEngine.UIElements;
using Cursor = UnityEngine.Cursor;
using Image = UnityEngine.UI.Image;

public class CameraFollow : MonoBehaviour
{
    public Animator anim;

    public int defaultScale = 8;
    public int battleScale = 6;
    private Transform target;
    public Vector3 offset;
    public float damping;
    private Camera cam;
    private Vector3 velocity = Vector3.zero;
    private GameObject Default;
    private GameObject Shrine;
    private GameObject Grade;
    private GameObject Travel;
    private GameObject Dialogue;
    private GameObject pauseMenu;
    public GameObject YouDied;
    public GameObject blackScreen;

    public bool noDamping = false;
    public bool inPause = false;

    private void Start()
    {
        Cursor.visible = false;

        Default = GameObject.Find("DefaultUI");
        Shrine = GameObject.Find("ShrineUI");
        Grade = GameObject.Find("GradeUI");
        Travel = GameObject.Find("TravelUI");
        Dialogue = GameObject.Find("DialogueUI");
        pauseMenu = GameObject.Find("PauseMenu");
        YouDied = GameObject.Find("Death");
        blackScreen = GameObject.Find("BlackScreen");
        anim = blackScreen.GetComponent<Animator>();
        YouDied.SetActive(false);
        changeCanvas(0);
        cam = GetComponent<Camera>();
        FindPlayer();
        changeScale(defaultScale);
    }

    public void FindPlayer()
    {
        target = GameObject.FindWithTag("Player").transform;
    }

    public void BlackScreenTransparency(int a)
    {
        if (a == 1)
        {
            anim.SetTrigger("Fade");
        }
    }
}
```

```

        else
        {
            anim.SetTrigger("UnFade");
        }
    }
    public void changeCanvas(int i)
    {
        switch (i)
        {
            case 0: // default
                Shrine.SetActive(false);
                Default.SetActive(true);
                Grade.SetActive(false);
                Travel.SetActive(false);
                Dialogue.SetActive(false);
                pauseMenu.SetActive(false);
                Cursor.visible = false;
                break;
            case 1: // shrine
                Grade.SetActive(false);
                Default.SetActive(false);
                Shrine.SetActive(true);
                if
(!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().lvl_0_completed)
                {
                    GameObject tmp = Shrine.transform.GetChild(2).gameObject;
                    tmp.SetActive(false);
                }
                else
                {
                    GameObject tmp = Shrine.transform.GetChild(2).gameObject;
                    tmp.SetActive(true);
                }
                Travel.SetActive(false);
                Dialogue.SetActive(false);
                pauseMenu.SetActive(false);
                Cursor.visible = true;
                break;
            case 2: // grade
                Grade.SetActive(true);
                Default.SetActive(false);
                Shrine.SetActive(false);
                Travel.SetActive(false);
                Dialogue.SetActive(false);
                pauseMenu.SetActive(false);
                Cursor.visible = true;
                break;
            case 3: // dialogue
                Grade.SetActive(false);
                Default.SetActive(false);
                Shrine.SetActive(false);
                Travel.SetActive(false);
                Dialogue.SetActive(true);
                pauseMenu.SetActive(false);
                Cursor.visible = true;
                break;
            case 4: // travel
                Grade.SetActive(false);
                Default.SetActive(false);
                Shrine.SetActive(false);
                Dialogue.SetActive(false);
                Travel.SetActive(true);
                pauseMenu.SetActive(false);
                Cursor.visible = true;
                break;

```

```

        case 5: // pauseMenu
            Grade.SetActive(false);
            Default.SetActive(false);
            Shrine.SetActive(false);
            Dialogue.SetActive(false);
            Travel.SetActive(false);
            pauseMenu.SetActive(true);
            Cursor.visible = true;
            break;
    }
}
private void Update()
{
    if (GameObject.FindWithTag("Player").GetComponent<PlayerChara>().isDead)
    {
        noDamping = true;
        return;
    }
    else if(noDamping && cam.transform.position != target.position)
    {
        noDamping = false;
        transform.position = target.position;
    }
    Vector3 movePos = target.position + offset;
    transform.position = Vector3.SmoothDamp(transform.position, movePos, ref
velocity, damping);
    if (Input.GetKey(KeyCode.P) && !inPause)
    {
        PauseStart();
    }
}

private void PauseStart()
{
    inPause = true;
    Time.timeScale = 0f;
    changeCanvas(5);
}

public void PauseStop()
{
    inPause = false;
    Time.timeScale = 1f;
    changeCanvas(0);
}

public void changeScale(int scale)
{
    cam.orthographicSize = Mathf.Lerp(cam.orthographicSize, scale,
Time.deltaTime);
}
}

```

Скрипт DialogueSRC

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using TMPro;
using Unity.Mathematics;
using UnityEngine;
using UnityEngine.PlayerLoop;
using UnityEngine.UIElements;
using Cursor = UnityEngine.Cursor;

```

```

using Image = UnityEngine.UI.Image;

public class DialogueSRC : MonoBehaviour
{
    private PlayerClass player;
    private Camera cam;
    private GameObject Dialogue;
    private CameraFollow camSCR;
    private GameObject messageObj;
    private GameObject namObj;
    private GameObject spriteObj;
    private GameObject bgObj;
    private GameObject heroSayObj;
    private GameObject sayButtonPrefab;
    private GameObject buttonPos;
    public List<GameObject> btnList;
    private List<string> que;
    private List<string> queType;
    private string messageText;
    private string nameStr;
    private string itemID;

    void Start()
    {
        player = GameObject.FindWithTag("Player").GetComponent<PlayerClass>();
        sayButtonPrefab = Resources.Load("sayBttn") as GameObject;
        cam = GetComponent<Camera>();
        Dialogue = GameObject.Find("DialogueUI");
        camSCR = GetComponent<CameraFollow>();
        spriteObj = Dialogue.transform.GetChild(1).gameObject;
        bgObj = Dialogue.transform.GetChild(0).gameObject;
        namObj = Dialogue.transform.GetChild(2).transform.GetChild(0).transform.GetChild(0).gameObject;
        messageObj = Dialogue.transform.GetChild(2).transform.GetChild(0).transform.GetChild(1).gameObject;
        heroSayObj = Dialogue.transform.GetChild(3).gameObject;
        buttonPos = heroSayObj.transform.GetChild(0).gameObject;
    }

    void Update()
    {
    }

    public IEnumerator DialogueSrc(TextAsset textFile, string queID)
    {
        que = new List<string>();
        queType = new List<string>();
        btnList = new List<GameObject>();
        camSCR.BlackScreenTransparency(1);
        yield return new WaitForSeconds(1.5f);
        camSCR.BlackScreenTransparency(2);
        camSCR.changeCanvas(3);
        var text = textFile.text;
        nameStr = getBetween(text, "name:", "\r");
        itemID = getBetween(text, "itemID:", "\r");
        var message = getBetween(text, "msg_00:", "\r");
        var spr_path = getBetween(text, "sprite:", "\r");
        var bg_path = getBetween(text, "bg:", "\r");

        switch (nameStr)
        {
            case "Кузнець":
                if (player.que_01_00_smith_open && !player.que_01_00_smith_completed)
                {
                    var que_01 = getBetween(text, "que_01_00:", "\r");
                }
            }
        }
    }
}

```

```

        que.Add(que_01);
        queType.Add("que_01_00");
    }
    else if (player.que_01_01_smith_open &&
!player.que_01_01_smith_completed)
    {
        var que_01 = getBetween(text, "que_02_00:", "\r");
        que.Add(que_01);
        queType.Add("que_02_00");
    }
    if (player.amount[player.inv.IndexOf(0)] > 0)
    {
        var que_grade = getBetween(text, "que_grade_00:", "\r");
        que.Add(que_grade);
        queType.Add("que_grade_00");
    }
    break;
    case "Алхімік":
        if (player.que_01_00_alchemist_open &&
!player.que_01_00_alchemist_completed)
        {
            var que_01 = getBetween(text, "que_01_00:", "\r");
            que.Add(que_01);
            queType.Add("que_01_00");
        }
        else if (player.que_01_01_alchemist_open &&
!player.que_01_01_alchemist_completed)
        {
            var que_01 = getBetween(text, "que_02_00:", "\r");
            que.Add(que_01);
            queType.Add("que_02_00");
        }
        if (player.amount[player.inv.IndexOf(1)] > 0)
        {
            var que_grade = getBetween(text, "que_grade_00:", "\r");
            que.Add(que_grade);
            queType.Add("que_grade_00");
        }
        break;
    case "Жнець полювання":
        if (player.que_01_00_reaper_open &&
!player.que_01_00_reaper_completed)
        {
            var que_01 = getBetween(text, "que_01_00:", "\r");
            que.Add(que_01);
            queType.Add("que_01_00");
        }
        else if (player.que_01_01_reaper_open &&
!player.que_01_01_reaper_completed)
        {
            var que_01 = getBetween(text, "que_02_00:", "\r");
            que.Add(que_01);
            queType.Add("que_02_00");
        }
        if (player.amount[player.inv.IndexOf(5)] > 0)
        {
            var que_grade = getBetween(text, "que_grade_00:", "\r");
            que.Add(que_grade);
            queType.Add("que_grade_00");
        }
        break;
    }

    var que_ex = getBetween(text, "que_exit:", "\r");
    que.Add(que_ex);
    queType.Add("que_exit");

```

```

        Sprite spr = LoadSpriteFromFile(spr_path);
        Sprite bg = LoadSpriteFromFile(bg_path);
        spriteObj.GetComponent<Image>().sprite = spr;
        bgObj.GetComponent<Image>().sprite = bg;
        messageObj.GetComponent<TextMeshProUGUI>().text = message;
        namObj.GetComponent<TextMeshProUGUI>().text = nameStr;
        quePrint(que, queType, text);
    }

    private void quePrint(List<string> que, List<string> queID, string textFile)
    {
        var posDiff = 0;
        for (int i = 0; i < que.Count; i++)
        {
            var posVec = new Vector3(buttonPos.transform.position.x,
            buttonPos.transform.position.y - posDiff, 0);
            GameObject btnnSay = Instantiate(sayButtonPrefab, posVec,
            quaternion.identity, heroSayObj.transform);
            posDiff += 1;
            btnnList.Add(btnnSay);
            if (queID[i].Contains("exit"))
            {
                btnnSay.AddComponent<LeaveDialogue>();
            }
            else if (queID[i].Contains("grade"))
            {
                btnnSay.AddComponent<BttnDialogueSRC>();
                btnnSay.GetComponent<BttnDialogueSRC>().bttnCount = i;
                btnnSay.GetComponent<BttnDialogueSRC>().messageID = queID[i];
                btnnSay.GetComponent<BttnDialogueSRC>().textString =
                getAllDialogue(queID[i], textFile);
                btnnSay.GetComponent<BttnDialogueSRC>().messageObj = messageObj;
                btnnSay.GetComponent<BttnDialogueSRC>().bttnList = btnnList;
                btnnSay.GetComponent<BttnDialogueSRC>().isGrade = true;
                btnnSay.GetComponent<BttnDialogueSRC>().name = nameStr;
            }
            else
            {
                btnnSay.AddComponent<BttnDialogueSRC>();
                btnnSay.GetComponent<BttnDialogueSRC>().bttnCount = i;
                btnnSay.GetComponent<BttnDialogueSRC>().messageID = queID[i];
                btnnSay.GetComponent<BttnDialogueSRC>().textString =
                getAllDialogue(queID[i], textFile);
                btnnSay.GetComponent<BttnDialogueSRC>().messageObj = messageObj;
                btnnSay.GetComponent<BttnDialogueSRC>().bttnList = btnnList;
                btnnSay.GetComponent<BttnDialogueSRC>().isGrade = false;
                btnnSay.GetComponent<BttnDialogueSRC>().idOpen = itemID;
                btnnSay.GetComponent<BttnDialogueSRC>().isText = true;
            }
            btnnSay.transform.GetChild(0).GetComponent<TextMeshProUGUI>().text =
            que[i];
        }
    }

    public string getAllDialogue(string messID, string text)
    {
        int lstDgt = 1;
        messageText = "";
        string id = getBetween(messID, "que_", "_0");
        while (true)
        {
            var tmpText = "";
            if (lstDgt >= 10)
            {
                tmpText = getBetween(text, "mess_" + id + "_" + lstDgt + ":", "\r");
            }
        }
    }

```

```

    }
    else
    {
        tmpText = getBetween(text, "mess_" + id + "_0" + lstDgt + ":", "\r");
    }
    if (tmpText == "")
    {
        break;
    }
    messageText += tmpText + "\r\n";
    lstDgt++;
}
return messageText;
}
public static string getBetween(string file, string strStart, string strEnd)
{
    if (file.Contains(strStart) && file.Contains(strEnd))
    {
        int Start, End;
        Start = file.IndexOf(strStart, 0) + strStart.Length;
        End = file.IndexOf(strEnd, Start);
        return file.Substring(Start, End - Start);
    }

    return "";
}
public Sprite LoadSpriteFromFile(string path)
{
    // Load the texture from the file
    Texture2D texture = LoadTextureFromFile(path);

    // Create a sprite from the texture
    Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
texture.height), Vector2.zero);

    return sprite;
}
public Texture2D LoadTextureFromFile(string path)
{
    // Load the image data from the file
    byte[] imageData = System.IO.File.ReadAllBytes(path);

    // Create a new texture and load the image data into it
    Texture2D texture = new Texture2D(2, 2);
    texture.LoadImage(imageData);

    return texture;
}
}

```

Скрипт EnemyBehavior

```

using System.Collections;
using System.Collections.Generic;
using System.Net.Http.Headers;
using UnityEngine;
using UnityEngine.EventSystems;

public class EnemyBehavior : MonoBehaviour
{
    public EnemyClass enemyClass;
    public bool inAction = false;
    void Start()
    {
        enemyClass = GetComponent<EnemyClass>();
        enemyClass.playerObj = GameObject.FindWithTag("Player");
    }
}

```

```

        enemyClass.animController.SetTrigger("start");
        transform.localScale = new Vector3(1, 1, 0);
    }

    void Update()
    {
        if (enemyClass.isDead || enemyClass.isStunned || !enemyClass.canAttack)
        {
            enemyClass.rb.velocity = Vector2.zero;
            return;
        }

        if (enemyClass.hpCurrent <= 0 && !enemyClass.isDead)
        {
            enemyClass.rb.velocity = Vector2.zero;
            enemyClass.Death();
        }

        enemyClass.distanceToPlayer = Vector2.Distance(transform.position,
        enemyClass.playerObj.transform.position);
        if (enemyClass.SeePlayer() && !enemyClass.CanAttack())
        {
            enemyClass.Move();
        }
        else if (enemyClass.CanAttack() && !enemyClass.isAttacking && !inAction)
        {
            enemyClass.rb.velocity = Vector2.zero;
            StartCoroutine(enemyClass.Attack());
        }
    }

    private IEnumerator Pause(float time)
    {
        inAction = true;
        yield return new WaitForSeconds(time);
        inAction = false;
    }
}

```

Скрипт BtnDialogueSRC

```

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

public class BtnDialogueSRC : MonoBehaviour
{
    public string messageID;
    public string textString;
    public GameObject messageObj;
    public int btnCount;
    private Button btn;
    public List<GameObject> btnList;
    public bool isGrade;
    public bool isText;
    public string name;
    public string idOpen;
    private bool isPrinting = false;
    private int strCount = 1;
    private bool isClicked = false;
    void Start()
    {
        Debug.Log(textString);
        btn = GetComponent<Button>();
    }
}

```

```

    btn.onClick.AddListener(onClickFunc);
}

private IEnumerator printText()
{
    var text = GetLine(textString, strCount);
    if (text == "")
    {
        EndButton();
    }
    isPrinting = true;
    for (int i = 0; i < text.Length; i++)
    {
        messageObj.GetComponent<TextMeshProUGUI>().text += text[i];
        yield return new WaitForSeconds(0.01f);
    }
    isPrinting = false;
    strCount++;
}

private void onClickFunc()
{
    if (isClicked)
    {
        return;
    }

    if (isText)
    {
        switch (idOpen)
        {
            case "0":
                if
(!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_00_smith_comple
ted)
                {
                    GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_00_smith_completed
= true;
                }
                else if
(!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_01_smith_comple
ted)
                {
                    GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_01_smith_completed
= true;
                }
                break;
            case "5":
                if
(!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_00_reaper_comple
ted)
                {
                    GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_00_reaper_comple
ted = true;
                }
                else if
(!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_01_reaper_comple
ted)
                {
                    GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_01_reaper_comple
ted = true;
                }
                break;
        }
    }
}

```

```

        case "1":
            if
                (!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_00_alchemist_completed)
                {
                    GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_00_alchemist_completed = true;
                }
            else if
                (!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_01_alchemist_completed)
                {
                    GameObject.FindWithTag("Player").GetComponent<PlayerClass>().que_01_01_alchemist_completed = true;
                }
            break;
        }
        isText = false;
    }

    if (isGrade)
    {
        switch (name)
        {
            case "Кузнець":
                GameObject.FindWithTag("Player").GetComponent<PlayerClass>().damage += 5;
                GameObject.FindWithTag("Player").GetComponent<PlayerClass>().
                    .amount[GameObject.FindWithTag("Player").GetComponent<PlayerClass>().inv.IndexOf(0)]--;
                ;
                break;
            case "Жнець полювання":
                if
                    (!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().lvl_1_open)
                    {
                        GameObject.FindWithTag("Player").GetComponent<PlayerClass>().lvl_1_open = true;
                    }
                else if
                    (!GameObject.FindWithTag("Player").GetComponent<PlayerClass>().lvl_2_open)
                    {
                        GameObject.FindWithTag("Player").GetComponent<PlayerClass>().lvl_2_open = true;
                    }
                GameObject.FindWithTag("Player").GetComponent<PlayerClass>().
                    .amount[GameObject.FindWithTag("Player").GetComponent<PlayerClass>().inv.IndexOf(5)]--;
                ;
                break;
            case "Алхімік":
                GameObject.FindWithTag("Player").GetComponent<PlayerClass>().flaskMax++;
                GameObject.FindWithTag("Player").GetComponent<PlayerClass>().
                    .amount[GameObject.FindWithTag("Player").GetComponent<PlayerClass>().inv.IndexOf(1)]--;
                ;
                break;
        }
        isGrade = false;
    }
    isClicked = true;
    messageObj.GetComponent<TextMeshProUGUI>().text = "";
    for (int i = 0; i < btnList.Count; i++)

```

```

        {
            if (i != btnCount)
            {
                btnList[i].SetActive(false);
            }
        }
        StartCoroutine(printText());
    }
    private string GetLine(string text, int lineNo)
    {
        string[] lines = text.Replace("\r", "").Split('\n');
        return lines.Length >= lineNo ? lines[lineNo-1] : null;
    }

    private void EndButton()
    {
        messageObj.GetComponent<TextMeshProUGUI>().text = "Щось ше?";
        for (int i = 0; i < btnList.Count; i++)
        {
            if (i != btnCount)
            {
                btnList[i].SetActive(true);
            }
            else
            {
                btnList[i].SetActive(false);
            }
        }
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Mouse0) && !isPrinting && isClicked)
        {
            messageObj.GetComponent<TextMeshProUGUI>().text = "";
            StartCoroutine(printText());
        }
    }
}

```

Скрипт EnterSRC

```

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

public class EnterSRC : MonoBehaviour
{
    private GameObject questionMark;
    [SerializeField] private TextAsset textFile;
    private bool inside = false;
    private bool inShop = false;
    private AudioSource entersfx;
    void Start()
    {
        entersfx = GetComponent<AudioSource>();
        questionMark =
        GameObject.FindWithTag("Player").transform.GetChild(3).gameObject;
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            questionMark.SetActive(true);
            inside = true;
        }
    }
}

```

```

    }
}

private void OnTriggerExit2D(Collider2D other)
{
    if (other.CompareTag("Player"))
    {
        questionMark.SetActive(false);
        inside = false;
        inShop = false;
    }
}

void Update()
{
    if (inside && Input.GetKeyDown(KeyCode.E) && !inShop)
    {
        entersfx.Play();
        GameObject.FindWithTag("Player").GetComponent<PlayerChara>().inDialogue =
true;
        StartCoroutine(GameObject.FindWithTag("MainCamera").GetComponent<DialogueSRC>().Dialog
ueSrc(textFile, "que_01_00"));
        inShop = true;
    }
}
}

```

Скрипт LeaveDialogue

```

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

public class LeaveDialogue : MonoBehaviour
{
    private Button btnn;
    private CameraFollow cam;
    private DialogueSRC dia;
    private GameObject player;
    void Start()
    {
        cam = GameObject.FindWithTag("MainCamera").GetComponent<CameraFollow>();
        dia = GameObject.FindWithTag("MainCamera").GetComponent<DialogueSRC>();

        btnn = GetComponent<Button>();
        btnn.onClick.AddListener(onClickFunc);
        player = GameObject.FindWithTag("Player");
    }
    private void onClickFunc()
    {
        player.GetComponent<PlayerChara>().inDialogue = false;
        foreach (var i in dia.btnnList)
        {
            Destroy(i);
        }
        cam.changeCanvas(0);
    }

    void Update()
    {
    }
}

```

Скрипт ContinueSRC

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.Windows;
using File = System.IO.File;

public class ContinueSRC : MonoBehaviour
{
    private Button btn;
    private string path;
    void Start()
    {
        path = Application.persistentDataPath + "/player.json";
        btn = GetComponent<Button>();
        btn.onClick.AddListener(Continue);
    }

    void Continue()
    {
        SaveClass tmpdata = SaveSystem.LoadPlayer();
        PlayerPrefs.SetInt("newGame", 1);

        GameObject.Find("LoadManager").GetComponent<Load>().LoadScene(tmpdata.currScene);
    }
    void Update()
    {
    }
}
```

Скрипт exitGameSRC

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;
using UnityEngine.UI;
using Button = UnityEngine.UI.Button;

public class exitGameSRC : MonoBehaviour
{
    private Button btn;
    void Start()
    {
        btn = GetComponent<Button>();
        btn.onClick.AddListener(ExitGame);
    }

    void ExitGame()
    {
        Debug.Log("Exit");
        Application.Quit();
    }
    void Update()
    {
    }
}
```

Скрипт mainMenuSRC

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

public class mainMenuSRC : MonoBehaviour
{
    private string path;
    private GameObject continueBtn;
    void Start()
    {
        continueBtn = GameObject.Find("ContinueBtn");
        path = Application.persistentDataPath + "/player.json";
        if (File.Exists(path))
        {
            continueBtn.SetActive(true);
        }
        else
        {
            continueBtn.SetActive(false);
        }
    }

    void Update()
    {
    }
}
```

Скрипт newGameSRC

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

public class newGameSRC : MonoBehaviour
{
    private Button btn;
    private string path;
    void Start()
    {
        path = Application.persistentDataPath + "/player.json";
        btn = GetComponent<Button>();
        btn.onClick.AddListener(NewGame);
    }

    void NewGame()
    {
        if (File.Exists(path))
        {
            File.Delete(path);
        }
        PlayerPrefs.SetInt("newGame", 0);
        GameObject.Find("LoadManager").GetComponent<Load>().LoadScene("lvl_0");
    }

    void Update()
    {
    }
}
```

```
}
```

Скрипт AdviceTriggerSRC

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

public class AdviceTriggerSRC : MonoBehaviour
{
    public int id;
    public bool isItem = false;

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            GameObject.Find("AdvisePanel").GetComponent<AdviceSRC>().CheckInt(id,
isItem);
            Destroy(this);
        }
    }
}
```

Скрипт FightBehaviour

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

public class FightBehaviour : MonoBehaviour
{
    [SerializeField] private PlayerMovement move;
    public PlayerClass playerClass;
    private PlayerChara chara;
    private LayerMask enemyMask;
    private float rayDistanse = 5f;
    public bool isFighting = false;
    public bool isAttacking = false;
    public bool isBlocking = false;
    public bool isParry = false;
    public bool isRecentlyHit = false;
    private float unHitTime = 3f;
    private float startUnHitTime = 0f;
    public GameObject cam;
    private float timeBtwAttack;
    private float startTimeBtwAttack = 0.5f;
    private int attackCount = 0;
    private int attackCountMax = 3;
    private float attackDelayTime = 1f;
    private float parryTime = 0.5f;
    private GameObject attackPos;
    private bool parryCan = true;
    public float attackRange = 2f;
    //public float damage = 10f;
    private bool isBlockClicked = false;
    private bool canAttack = true;
    private Animator animCont;

    public AudioSource attacksfx;
    void Start()
    {
        playerClass = GetComponent<PlayerClass>();
    }
}
```

```

        animCont = GetComponent<Animator>();
        chara = GetComponent<PlayerChara>();
        move = GetComponent<PlayerMovement>();
        enemyMask = LayerMask.GetMask("Enemy");
        attackPos = GameObject.Find("attackPos");
        cam = GameObject.Find("Main Camera");
    }

    public bool canBlockCheck()
    {
        if (!isAttacking)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public IEnumerator attack()
    {
        attacksfx.Play();
        animCont.SetTrigger("Attack");
        isAttacking = true;
        canAttack = false;
        yield return new WaitForSeconds(0.25f);
        timeBtwAttack = startTimeBtwAttack;
        Collider2D[] enemyToDamage =
Physics2D.OverlapCircleAll(attackPos.transform.position, attackRange, enemyMask);
        for (int i = 0; i < enemyToDamage.Length; i++)
        {

enemyToDamage[i].GetComponent<EnemyClass>().TakeDamage(playerClass.damage);
        }
        isAttacking = false;
        StartCoroutine(AttackCD());
    }

    public IEnumerator CanParry()
    {
        if (!isAttacking && !isBlocking)
        {
            animCont.SetTrigger("Parry");
            isParry = true;
            parryCan = false;
            yield return new WaitForSeconds(0.2f);
            isParry = false;
            StartCoroutine(ParryCD());
        }
    }

    private IEnumerator AttackCD()
    {
        yield return new WaitForSeconds(startTimeBtwAttack);
        canAttack = true;
    }

    public IEnumerator ParryCD()
    {
        yield return new WaitForSeconds(parryTime);
        parryCan = true;
    }

    void Update()
    {

```

```

        if (chara.isDead || chara.isStunned || chara.inDialogue ||
cam.GetComponent<CameraFollow>().inPause)
            return;

        if (Input.GetKeyDown(KeyCode.Mouse0) && canAttack && move.rb.velocity ==
Vector2.zero)
        {
            StartCoroutine(attack());
        }
        else if (Input.GetKey(KeyCode.Mouse1))
        {
            if (canBlockCheck())
            {
                animCont.SetBool("Block", true);
                isBlocking = true;
                StartCoroutine(CanParry());
            }
        }
        else if (Input.GetMouseButtonDown(2) && parryCan && move.rb.velocity ==
Vector2.zero)
        {
            StartCoroutine(CanParry());
        }
        else
        {
            animCont.SetBool("Block", false);
            isBlocking = false;
        }

        if (isRecentlyHit)
        {
            startUnHitTime += Time.deltaTime;
            if (startUnHitTime >= unHitTime)
            {
                startUnHitTime = 0;
                isRecentlyHit = false;
            }
        }
    }
}

```

Скрипт PlayerChara

```

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

public class PlayerChara : MonoBehaviour
{
    #region BasicStats
    public float hpCurrent = 100f;

    public float spCurrent = 50f;

    private bool spRegenerating = false;
    private bool enduranceRegenerating = false;

    public bool isStunned = false;
    private float stunTime = 1f;

    public GameObject RespawnPoint;
    private LevelCreate levelCreateSRC;
    private FightBehaviour fight;

    public bool isDead = false;
    public bool deathCouroutine = false;

```

```

public bool inDialogue = false;
#endregion

#region MoveCost
[SerializeField] public float jumpCost = 10f;
[SerializeField] public float dashCost = 25f;
#endregion
public PlayerClass player;
public bool decrease = false;
public Camera cam;
public CameraFollow camSRC;
private Animator animCont;

public AudioSource blocksfx;
public AudioSource parrysfx;
public AudioSource hitSFX;

public bool takeDmg(float dmg, int type)
{
    if(GameObject.FindWithTag("Player").GetComponent<PlayerMovement>().isDashing
    && (type == 0 || type == 1)) // if dodging && onlyDodge attack or default attack
        return true;

    if (GameObject.FindWithTag("Player").GetComponent<PlayerMovement>().isDashing
    && type == 2) // if dodging && onlyParry attack
    {
        hpCurrent -= dmg;
        return false;
    }

    if (fight.isBlocking && type == 0 && !fight.isParry) // if block default
    attack
    {
        spCurrent -= dmg;
        blocksfx.Play();
        return false;
    }

    if (fight.isParry && (type == 0 || type == 2)) // if parry default or
    parryOnly attack
    {
        parrysfx.Play();
        StopCoroutine(fight.CanParry());
        return true;
    }

    animCont.SetTrigger("Hurt"); // others
    hpCurrent -= dmg;
    hitSFX.Play();
    return false;
}

public void getPieces(float amount)
{
    var target = player.pieces + amount;
    player.pieces += (int)amount;
}

public bool canJump()
{
    if (spCurrent >= jumpCost)
        return true;
    else
        return false;
}

```

```

public bool canDash()
{
    if (spCurrent >= dashCost)
        return true;
    else
        return false;
}

void Start()
{
    player = GetComponent<PlayerClass>();
    animCont = GetComponent<Animator>();
    cam = Camera.main;
    camSRC = cam.GetComponent<CameraFollow>();
    levelCreateSRC = GameObject.FindWithTag("lvlScr").GetComponent<LevelCreate>();
    RespawnPoint = GameObject.FindWithTag("playerPos");
    fight = GetComponent<FightBehaviour>();
    assignStats();
}

public void RespawnPointAssign(GameObject pos)
{
    RespawnPoint = pos;
}

public void assignStats()
{
    hpCurrent = player.hpMax;
    spCurrent = player.spMax;
}

private IEnumerator RegenSP()
{
    spRegenerating = true;
    yield return new WaitForSeconds(3f);
    while (true)
    {
        if (spCurrent < player.spMax)
        {
            spCurrent += 4;
            yield return new WaitForSeconds(0.3f);
        }
        else
        {
            spRegenerating = false;
            yield return null;
            break;
        }
    }
}

public IEnumerator Death()
{
    deathCouroutine = true;
    isDead = true;
    animCont.SetBool("isDead", true);
    cam.GetComponent<CameraFollow>().YouDied.SetActive(true);
    camSRC.BlackScreenTransparency(1);
    animCont.SetBool("isDead", false);
    yield return new WaitForSeconds(3f);
    assignStats();
    player.pieces /= 2;
    gameObject.transform.position = RespawnPoint.transform.position;
    cam.GetComponent<CameraFollow>().YouDied.SetActive(false);
}

```

```

        levelCreateSRC.GetComponent<LevelCreate>().foreachEnemyFindDestroy();

levelCreateSRC.GetComponent<LevelCreate>().foreachCycle(levelCreateSRC.GetComponent<LevelCreate>().Enemy, levelCreateSRC.GetComponent<LevelCreate>().EnemyPos);
        isDead = false;
        deathCouroutine = false;
        camSRC.BlackScreenTransparency(0);
    }

    private IEnumerator Stun()
    {
        isStunned = true;
        yield return new WaitForSeconds(stunTime);
        isStunned = false;
    }

    void Update()
    {
        if (hpCurrent <= 0 && !deathCouroutine)
        {
            StartCoroutine(Death());
        }
        if (hpCurrent > player.hpMax)
        {
            hpCurrent = player.hpMax;
        }

        if (spCurrent < 0)
            spCurrent = 0;
        if (spCurrent < player.spMax && !spRegenerating)
        {
            StartCoroutine(RegenSP());
        }
        if (spCurrent > player.spMax)
            spCurrent = player.spMax;
    }
}

```

Скрипт PlayerClass

```

using System;
using System.Collections;
using System.Collections.Generic;
using Unity.Mathematics;
using UnityEditorInternal;
using UnityEngine;
using UnityEngine.SceneManagement;

public class PlayerClass : MonoBehaviour
{
    #region PlayerChara
    public float hpMax = 100f;
    public float spMax = 50f;
    public int piecesToGrade = 10;
    public int pieces = 0;
    public int level = 1;
    #endregion

    #region FightBehaviour
    public float damage = 10;
    #endregion

    #region Movement
    public float speed = 8f;
    public float jumpPower = 16f;
    public float spintMult = 1.5f;

```

```

#endregion

#region Inventory
public int flaskMax = 2;
public List<int> inv;
public List<int> amount;
#endregion

#region Progress
public bool lvl_0_open = true;
public bool lvl_hub_open = true;
public bool lvl_1_open = false;
public bool lvl_2_open = false;

public bool lvl_0_completed = false;
public bool lvl_1_completed = false;
public bool lvl_2_completed = false;

//smith
public bool que_01_00_smith_open= true;
public bool que_01_00_smith_completed = false;
public bool que_01_01_smith_open = false;
public bool que_01_01_smith_completed = false;

//reaper
public bool que_01_00_reaper_open = true;
public bool que_01_00_reaper_completed = false;
public bool que_01_01_reaper_open = false;
public bool que_01_01_reaper_completed = false;

//alchemist
public bool que_01_00_alchemist_open = true;
public bool que_01_00_alchemist_completed = false;
public bool que_01_01_alchemist_open = false;
public bool que_01_01_alchemist_completed = false;

//lvl_0_items
public bool lvl_0_shard_picked = false;
public bool lvl_0_flower_picked = false;
public bool lvl_0_scroll_picked = false;

//lvl_1_items
public bool lvl_1_shard_picked = false;
public bool lvl_1_flower_picked = false;
public bool lvl_1_scroll_picked = false;
#endregion
public string currScene;

public Vector3 respPos;

public PlayerClass(PlayerClass data)
{
    hpMax = data.hpMax;
    spMax = data.spMax;
    level = data.level;
    damage = data.damage;
    piecesToGrade = data.piecesToGrade;
    pieces = data.pieces;
    currScene = data.updateScene();
    flaskMax = data.flaskMax;
    inv = data.inv;
    amount = data.amount;
    lvl_0_open = data.lvl_0_open;
    lvl_hub_open = data.lvl_hub_open;
    lvl_1_open = data.lvl_1_open;
    lvl_2_open = data.lvl_2_open;
}

```

```

    lvl_0_completed = data.lvl_0_completed;
    lvl_1_completed = data.lvl_1_completed;
    lvl_2_completed = data.lvl_2_completed;

    que_01_00_smith_open = data.que_01_00_smith_open;
    que_01_00_smith_completed = data.que_01_00_smith_completed;
    que_01_01_smith_open = data.que_01_01_smith_open;
    que_01_01_smith_completed = data.que_01_01_smith_completed;

    que_01_00_reaper_open = data.que_01_00_reaper_open;
    que_01_00_reaper_completed = data.que_01_00_reaper_completed;
    que_01_01_reaper_open = data.que_01_01_reaper_open;
    que_01_01_reaper_completed = data.que_01_01_reaper_completed;

    que_01_00_alchemist_open = data.que_01_00_alchemist_open;
    que_01_00_alchemist_completed = data.que_01_00_alchemist_completed;
    que_01_01_alchemist_open = data.que_01_01_alchemist_open;
    que_01_01_alchemist_completed = data.que_01_01_alchemist_completed;

    lvl_0_shard_picked = data.lvl_0_shard_picked;
    lvl_0_flower_picked = data.lvl_0_flower_picked;
    lvl_0_scroll_picked = data.lvl_0_scroll_picked;

    lvl_1_shard_picked = data.lvl_1_shard_picked;
    lvl_1_flower_picked = data.lvl_1_flower_picked;
    lvl_1_scroll_picked = data.lvl_1_scroll_picked;
}

public string updateScene()
{
    currScene = SceneManager.GetActiveScene().name;
    return currScene;
}

public void GetPos()
{
    var tmpPlayer = GameObject.FindWithTag("Player");
    respPos = new
Vector3(transform.position.x,transform.position.y,transform.position.z);
}

}

```

Скрипт PlayerInventory

```

using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UIElements;
using Image = UnityEngine.UI.Image;

public class PlayerInventory : MonoBehaviour
{
    private PlayerChara chara;
    private PlayerMovement move;
    private FightBehaviour fight;
    private PlayerClass playerClass;
    private GameObject itemBar;
    private Transform posItemUse;
    public int quickCurrent = 0;
    private GameObject knife;
    private GameObject bomb;
    public TextAsset itemsList;
    public TextAsset itemsSprite;
    private string itemsListString;
}

```

```

public AudioSource bombsfx;
public AudioSource knivesfx;
public AudioSource potionsfx;
void Start()
{
    playerClass = GetComponent<PlayerClass>();
    knife = Resources.Load("Knife") as GameObject;
    bomb = Resources.Load("Bomb") as GameObject;
    posItemUse = gameObject.transform.GetChild(1).transform;
    itemBar = GameObject.Find("itemBar");
    chara = GetComponent<PlayerChara>();
    move = GetComponent<PlayerMovement>();
    fight = GetComponent<FightBehaviour>();
    itemsListString = itemsList.text;
    if (!playerClass.inv.Contains(2))
    {
        playerClass.inv.Add(2);
        playerClass.amount.Add(playerClass.flaskMax);
    }
    ChangeSprite();
}
public void TakeItem(int id)
{
    if (playerClass.inv.Contains(id))
    {
        int i = playerClass.inv.IndexOf(id);
        playerClass.amount[i]++;
    } else
    {
        playerClass.inv.Add(id);
        playerClass.amount.Add(1);
    }
}

public void UseItem(int index)
{
    var tmpID = playerClass.inv.IndexOf(index);
    if (playerClass.inv[index] == 2 && playerClass.amount[index] > 0)
    {
        chara.hpCurrent += 25;
        potionsfx.Play();
        playerClass.amount[index]--;
    }
    else if (playerClass.inv[index] == 3 && playerClass.amount[index] > 0)
    {
        var tmppos = new Vector2(posItemUse.position.x, posItemUse.position.y);
        GameObject knife_throw = Instantiate(knife, tmppos, Quaternion.identity);
        knivesfx.Play();
        playerClass.amount[index]--;
    }
    else if (playerClass.inv[index] == 4 && playerClass.amount[index] > 0)
    {
        var tmppos = new Vector2(posItemUse.position.x, posItemUse.position.y);
        GameObject bomb_throw = Instantiate(bomb, tmppos, Quaternion.identity);
        playerClass.amount[index]--;
        bombsfx.Play();
    }
    ChangeSprite();
}

public void ResetFlask()
{
    playerClass.amount[playerClass.inv.IndexOf(2)] = playerClass.flaskMax;
}

```

```

private void ChangeItem(int n)
{
    if (quickCurrent == playerClass.inv.Count-1 && n == 1)
    {
        quickCurrent = 0;
        return;
    }
    else if (quickCurrent == 0 && n == -1)
    {
        quickCurrent = playerClass.inv.Count-1;
        return;
    }

    quickCurrent += n;
}

void Update()
{
    if (GameObject.FindWithTag("MainCamera").GetComponent<CameraFollow>().inPause)
    {
        return;
    }

    if (Input.GetKeyDown(KeyCode.I))
    {
        Debug.Log(playerClass.inv);
    }
    else if (Input.GetKeyDown(KeyCode.Q) && !fight.isAttacking &&
!fight.isBlocking)
    {
        UseItem(quickCurrent);
    }
    else if (Input.GetAxis("Mouse ScrollWheel") > 0f ) // forward
    {
        ChangeItem(1);
        ChangeSprite();
    }
    else if (Input.GetAxis("Mouse ScrollWheel") < 0f ) // backwards
    {
        ChangeItem(-1);
        ChangeSprite();
    }
}

private void ChangeSprite()
{
    var tmp = playerClass.inv[quickCurrent];
    var tmp2 = playerClass.inv.IndexOf(tmp);
    var text = itemsSprite.text;
    var spr_path = getBetween(text,playerClass.inv[quickCurrent].ToString() + ":"
, "\r");
    Sprite itmSpr = LoadSpriteFromFile(spr_path);
    itemBar.transform.GetChild(2).GetComponent<TextMeshProUGUI>().text =
playerClass.amount[tmp2].ToString();
    itemBar.transform.GetChild(1).GetComponent<Image>().sprite = itmSpr;
}

private static string getBetween(string file, string strStart, string strEnd)
{
    if (file.Contains(strStart) && file.Contains(strEnd))
    {
        int Start, End;
        Start = file.IndexOf(strStart, 0) + strStart.Length;
        End = file.IndexOf(strEnd, Start);
        return file.Substring(Start, End - Start);
    }

    return "";
}

```

```

    }
    private Sprite LoadSpriteFromFile(string path)
    {
        // Load the texture from the file
        Texture2D texture = LoadTextureFromFile(path);

        // Create a sprite from the texture
        Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
        texture.height), Vector2.zero);

        return sprite;
    }
    private Texture2D LoadTextureFromFile(string path)
    {
        // Load the image data from the file
        byte[] imageData = System.IO.File.ReadAllBytes(path);

        // Create a new texture and load the image data into it
        Texture2D texture = new Texture2D(2, 2);
        texture.LoadImage(imageData);

        return texture;
    }
}

```

Скрипт PlayerMovement

```

using System;
using System.Collections;
using System.Collections.Generic;
using UnityEditor;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public Animator animator;
    private SpriteRenderer sprite;
    #region Variables
    private float horizontal;
    public float faceDir;

    [SerializeField] public Rigidbody2D rb;
    [SerializeField] private GameObject groundCheck;
    [SerializeField] private GameObject wallCheck;
    [SerializeField] private LayerMask groundLayer;

    [SerializeField] private float dashingVelocity;
    [SerializeField] private float dashingTime;
    public Vector2 dashingDir;
    public float dashForce = 5f;
    public float dashCost = 25f;
    public float dashCD = 1f;
    public float dashCDcurr = 0f;
    private float lastDodgeTime = -1f;
    public bool isDashing;
    private bool _canDash = true;
    #endregion

    public CapsuleCollider2D coll;

    public GameObject player;
    [SerializeField] private PlayerChara stats;
    [SerializeField] private FightBehaviour fight;
    public PlayerClass playerClass;
    public GameObject interestPoint;
    public AudioSource dashSFX;
}

```

```

public AudioSource jumpSFX;

private void Start()
{
    coll = GetComponent<CapsuleCollider2D>();
    player = GameObject.FindWithTag("Player");
    playerClass = GetComponent<PlayerClass>();
    sprite = GetComponent<SpriteRenderer>();
    animator = GetComponent<Animator>();
    rb = GetComponent<Rigidbody2D>();
    stats = GetComponent<PlayerChara>();
    fight = GetComponent<FightBehaviour>();
    groundCheck = GameObject.Find("groundCheck");
}

void Update()
{
    if (stats.isDead || stats.isStunned || stats.inDialogue || fight.isParry ||
fight.isAttacking )
    {
        rb.velocity = Vector2.zero;
        return;
    }

    if (GameObject.FindWithTag("MainCamera").GetComponent<CameraFollow>().inPause)
    {
        return;
    }

    if (isDashing && AtWall())
    {
        rb.velocity = Vector2.zero;
        return;
    }

    if(isDashing)
        return;

    var inputX = Input.GetAxisRaw("Horizontal");
    var jumpInput = Input.GetKeyDown(KeyCode.W);
    var dodgeInput = Input.GetKeyDown(KeyCode.Space);

    if (fight.isBlocking)
    {
        rb.velocity = Vector2.zero;
        animator.SetFloat("Speed", 0);
        return;
    }
    animator.SetFloat("Speed", Math.Abs(inputX));

    if (inputX > 0)
    {
        player.transform.localScale = new Vector3(6,6,1);
    }
    else if (inputX < 0)
    {
        player.transform.localScale = new Vector3(-6,6,1);
    }

    if (!fight.isBlocking)
    {
        rb.velocity = new Vector2(inputX * playerClass.speed, rb.velocity.y);
        if (jumpInput && isGrounded() && stats.canJump())
        {
            jumpSFX.Play();
            rb.velocity = new Vector2(rb.velocity.x, playerClass.jumpPower);
        }
    }
}

```

```

        stats.spCurrent -= stats.jumpCost;
    }

    if (dodgeInput && CanDodge())
    {
        Dodge();
    }

    if (isGrounded())
    {
        _canDash = true;
        animator.SetBool("isJump", false);
    }
    else
    {
        animator.SetBool("isJump", true);
    }

    if (!fight.isFighting)
    {
        if (inputX == 1f || inputX == -1f)
        {
            faceDir = inputX;
        }
    }
}

private bool CanDodge()
{
    return Time.time - lastDodgeTime > dashCD && stats.spCurrent >= dashCost &&
rb.velocity != Vector2.zero && !isDashing;
}

private void Dodge()
{
    dashSFX.Play();
    lastDodgeTime = Time.time;
    stats.spCurrent -= dashCost;
    isDashing = true;
    coll.isTrigger = true;
    rb.constraints = RigidbodyConstraints2D.FreezePositionY;
    rb.AddForce(FaceDirection() * dashForce, ForceMode2D.Impulse);
    animator.SetBool("isDash", true);
    StartCoroutine(StopDashing());
}

private IEnumerator StopDashing()
{
    yield return new WaitForSeconds(dashingTime);
    isDashing = false;
    animator.SetBool("isDash", false);
    coll.isTrigger = false;
    rb.constraints &= ~RigidbodyConstraints2D.FreezePositionY;
    rb.constraints = RigidbodyConstraints2D.FreezeRotation;
}

public bool isGrounded()
{
    return Physics2D.OverlapCircle(groundCheck.transform.position,
0.5f, groundLayer);
}

public bool AtWall()
{

```

```

        return Physics2D.OverlapCircle(wallCheck.transform.position, 0.1f,
groundLayer);
    }

    public Vector2 FaceDirection()
    {
        if (faceDir == 1)
        {
            return Vector2.right;
        }
        else
        {
            return Vector2.left;
        }
    }
}

```

Скрипт SaveClass

```

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

[System.Serializable]
public class SaveClass
{
    #region PlayerStats
    public float hpMax;
    public float spMax;
    public int level;
    public float damage;
    public int piecesToGrade;
    public int pieces;
    public string currScene;
    public float[] position;
    public int flaskMax;
    public List<int> inv;
    public List<int> amount;
    #endregion

    #region ProgresVars
    public bool lvl_0_open = true;
    public bool lvl_hub_open = true;
    public bool lvl_1_open = false;
    public bool lvl_2_open = false;

    public bool lvl_0_completed = false;
    public bool lvl_1_completed = false;
    public bool lvl_2_completed = false;

    //smith
    public bool que_01_00_smith_open = true;
    public bool que_01_00_smith_completed = false;
    public bool que_01_01_smith_open = false;
    public bool que_01_01_smith_completed = false;

    //reaper
    public bool que_01_00_reaper_open = true;
    public bool que_01_00_reaper_completed = false;
    public bool que_01_01_reaper_open = false;
    public bool que_01_01_reaper_completed = false;

    //alchemist
    public bool que_01_00_alchemist_open = true;
    public bool que_01_00_alchemist_completed = false;
    public bool que_01_01_alchemist_open = false;

```

```

public bool que_01_01_alchemist_completed = false;

//lvl_0_items
public bool lvl_0_shard_picked = false;
public bool lvl_0_flower_picked = false;
public bool lvl_0_scroll_picked = false;

//lvl_1_items
public bool lvl_1_shard_picked = false;
public bool lvl_1_flower_picked = false;
public bool lvl_1_scroll_picked = false;
#endregion

public SaveClass(PlayerClass player)
{
    hpMax = player.hpMax;
    spMax = player.spMax;
    level = player.level;
    damage = player.damage;
    piecesToGrade = player.piecesToGrade;
    pieces = player.pieces;
    currScene = player.updateScene();
    position = new float[3];
    position[0] = GameObject.FindWithTag("Player").transform.position.x;
    position[1] = GameObject.FindWithTag("Player").transform.position.y;
    position[2] = GameObject.FindWithTag("Player").transform.position.z;
    flaskMax = player.flaskMax;
    inv = player.inv;
    amount = player.amount;
    lvl_0_open = player.lvl_0_open;
    lvl_hub_open = player.lvl_hub_open;
    lvl_1_open = player.lvl_1_open;
    lvl_2_open = player.lvl_2_open;
    lvl_0_completed = player.lvl_0_completed;
    lvl_1_completed = player.lvl_1_completed;
    lvl_2_completed = player.lvl_2_completed;

    que_01_00_smith_open = player.que_01_00_smith_open;
    que_01_00_smith_completed = player.que_01_00_smith_completed;
    que_01_01_smith_open = player.que_01_01_smith_open;
    que_01_01_smith_completed = player.que_01_01_smith_completed;

    que_01_00_reaper_open = player.que_01_00_reaper_open;
    que_01_00_reaper_completed = player.que_01_00_reaper_completed;
    que_01_01_reaper_open = player.que_01_01_reaper_open;
    que_01_01_reaper_completed = player.que_01_01_reaper_completed;

    que_01_00_alchemist_open = player.que_01_00_alchemist_open;
    que_01_00_alchemist_completed = player.que_01_00_alchemist_completed;
    que_01_01_alchemist_open = player.que_01_01_alchemist_open;
    que_01_01_alchemist_completed = player.que_01_01_alchemist_completed;

    lvl_0_shard_picked = player.lvl_0_shard_picked;
    lvl_0_flower_picked = player.lvl_0_flower_picked;
    lvl_0_scroll_picked = player.lvl_0_scroll_picked;

    lvl_1_shard_picked = player.lvl_1_shard_picked;
    lvl_1_flower_picked = player.lvl_1_flower_picked;
    lvl_1_scroll_picked = player.lvl_1_scroll_picked;
}
}

```

Скрипт SaveSystem

```
using UnityEngine;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

public class SaveSystem : MonoBehaviour{

    public static void SavePlayer(PlayerClass player)
    {
        Debug.Log("SavePlayer");
        BinaryFormatter formatter = new BinaryFormatter();
        string path = Application.persistentDataPath + "/player.json";
        FileStream stream = new FileStream(path, FileMode.Create);

        SaveClass data = new SaveClass(player);

        formatter.Serialize(stream, data);
        stream.Close();
    }

    public static SaveClass LoadPlayer()
    {
        string path = Application.persistentDataPath + "/player.json";
        if (File.Exists(path))
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream stream = new FileStream(path, FileMode.Open);

            SaveClass data = formatter.Deserialize(stream) as SaveClass;

            stream.Close();

            return data;
        }
        else
        {
            Debug.LogError("Save file not found in " +path);
            return null;
        }
    }
}
```

Скрипт AdviceSRC

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

public class AdviceSRC : MonoBehaviour
{
    public TextAsset adviceFile;
    public TextAsset itemFile;
    public GameObject adviceTextObj;
    public GameObject textPlusItem;
    public GameObject itemIMG;
    public GameObject panel;

    void Start()
    {
        panel.SetActive(false);
    }

    public void CheckInt(int adviceInt, bool isItem)
```

```

{
    panel.SetActive(true);
    if (isItem)
    {
        string itemString = itemFile.ToString();
        adviceTextObj.SetActive(false);
        textPlusItem.SetActive(true);
        itemIMG.SetActive(true);
        var spr_path = getBetween(itemString, adviceInt + ":", "\r");
        Sprite spr = LoadSpriteFromFile(spr_path);
        itemIMG.GetComponent<Image>().sprite = spr;
    }
    else
    {

        string adviceString = adviceFile.ToString();
        adviceTextObj.SetActive(true);
        textPlusItem.SetActive(false);
        itemIMG.SetActive(false);
        var adviceText = getBetween(adviceString, "advice" + adviceInt + ":",
"\r");
        adviceTextObj.GetComponent<TextMeshProUGUI>().text = adviceText;
    }

    StartCoroutine(HideTime());
}

IEnumerator HideTime()
{
    yield return new WaitForSeconds(5f);
    panel.SetActive(false);
}

public static string getBetween(string file, string strStart, string strEnd)
{
    if (file.Contains(strStart) && file.Contains(strEnd))
    {
        int Start, End;
        Start = file.IndexOf(strStart, 0) + strStart.Length;
        End = file.IndexOf(strEnd, Start);
        return file.Substring(Start, End - Start);
    }

    return "";
}

public Sprite LoadSpriteFromFile(string path)
{
    // Load the texture from the file
    Texture2D texture = LoadTextureFromFile(path);

    // Create a sprite from the texture
    Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
texture.height), Vector2.zero);

    return sprite;
}

public Texture2D LoadTextureFromFile(string path)
{
    // Load the image data from the file
    byte[] imageData = System.IO.File.ReadAllBytes(path);

    // Create a new texture and load the image data into it
    Texture2D texture = new Texture2D(2, 2);
    texture.LoadImage(imageData);
}

```

```

    return texture;
}
}

```

Скрипт Load

```

using System.Collections;
using TMPro;
using Unity.Mathematics;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UIElements;
using Image = UnityEngine.UI.Image;
using Random = System.Random;

public class Load : MonoBehaviour
{
    [SerializeField]public GameObject loadingScreen;
    [SerializeField]public Image progressBar;
    [SerializeField]public GameObject hintObj;
    [SerializeField]public GameObject input;
    [SerializeField]public TextAsset hints_file;
    private AsyncOperation asyncOperation;
    private bool isLoadingComplete = false;

    private void Update()
    {
        if (isLoadingComplete && Input.GetKeyDown(KeyCode.Return))
        {
            ActivateScene();
        }
    }

    public void LoadScene(string sceneName)
    {
        var hints_str = hints_file.text;
        var hints_amount = hints_str.Split('\r').Length;
        Random rng = new Random();
        var hint_id = rng.Next(0, hints_amount);
        var hint_text = getBetween(hints_str, hint_id + ":", "\r");
        hintObj.GetComponent<TextMeshProUGUI>().text = "Запис у щоденнику " + (hint_id
+ 1) + "\n" + hint_text;

        var bg = LoadSpriteFromFile("Assets/Images/"+ sceneName + "_load.jpg");
        loadingScreen.GetComponent<Image>().sprite = bg;

        StartCoroutine(LoadSceneAsync(sceneName));
    }

    private IEnumerator LoadSceneAsync(string sceneName)
    {
        loadingScreen.SetActive(true);
        input.SetActive(false);

        yield return new WaitForSeconds(1f); // Затримка на 1 секунду

        asyncOperation = SceneManager.LoadSceneAsync(sceneName);
        asyncOperation.allowSceneActivation = false;

        while (!asyncOperation.isDone)
        {
            float progress = Mathf.Clamp01(asyncOperation.progress / 0.9f);
            progressBar.fillAmount = progress;
            if (progress >= 1.0f)

```

```

        {
            isLoadingComplete = true;
            input.SetActive(true);
        }

        yield return null;
    }

    ActivateScene();
}

private void ActivateScene()
{
    asyncOperation.allowSceneActivation = true;
    loadingScreen.SetActive(false);
}

public static string getBetween(string file, string strStart, string strEnd)
{
    if (file.Contains(strStart) && file.Contains(strEnd))
    {
        int Start, End;
        Start = file.IndexOf(strStart, 0) + strStart.Length;
        End = file.IndexOf(strEnd, Start);
        return file.Substring(Start, End - Start);
    }

    return "";
}

public Sprite LoadSpriteFromFile(string path)
{
    // Load the texture from the file
    Texture2D texture = LoadTextureFromFile(path);

    // Create a sprite from the texture
    Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
texture.height), Vector2.zero);

    return sprite;
}

public Texture2D LoadTextureFromFile(string path)
{
    // Load the image data from the file
    byte[] imageData = System.IO.File.ReadAllBytes(path);

    // Create a new texture and load the image data into it
    Texture2D texture = new Texture2D(2, 2);
    texture.LoadImage(imageData);

    return texture;
}
}

```

Скрипт UI

```

using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using Image = Microsoft.Unity.VisualStudio.Editor.Image;

public class UI : MonoBehaviour
{
    [SerializeField] private Canvas canvas;
    [SerializeField] private GameObject hp;
}

```

```

[SerializeField] private GameObject sp;
[SerializeField] private TextMeshProUGUI pieces;

[SerializeField] private PlayerChara playerStats;

public PlayerClass playerClass;
void Start()
{
    canvas = GetComponent<Canvas>();
    playerStats = GameObject.FindWithTag("Player").GetComponent<PlayerChara>();
    playerClass = GameObject.FindWithTag("Player").GetComponent<PlayerClass>();
}

void Update()
{
    hp.GetComponent<UnityEngine.UI.Image>().fillAmount = playerStats.hpCurrent /
playerClass.hpMax;
    sp.GetComponent<UnityEngine.UI.Image>().fillAmount = playerStats.spCurrent /
playerClass.spMax;

    var pie = playerClass.pieces.ToString();
    pieces.text = pie;
}
}

```

