

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**"Колекційно карткова гра з елементами РПГ та
покроковою боївкою"**

Виконав студент групи ІПЗк-20-1
КОПТЯЄВ Максим Павлович

Керівник роботи:
ЧИЖМОТРИЯ Олена Геннадіївна

Рецензент:
КРАВЧЕНКО Світлана Миколаївна

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення

Андрій МОРОЗОВ

«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОПТЯЄВ Максим Павлович**

Керівник роботи: **ЧИЖМОТРИ Олена Геннадіївна**

Тема роботи: **"Колекційно карткова гра з елементами РПГ та покроковою боївкою",**

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: Об'єктом дослідження є процес розробки гри

"Колекційно карткова гра з елементами РПГ та покроковою боївкою" та взаємодія гравців з грою. Розробка гри спрямована на створення цікавої та захоплюючої гри для гравців.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	16.02.2023	16.02.2023
2	Чижмотря О.Г.	16.02.2023	16.02.2023
3	Чижмотря О.Г.	16.02.2023	16.02.2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	16 лютого 2023 – 18 лютого 2023	виконано
2	Аналіз аналогів програмного продукту	18 лютого 2023 – 25 лютого 2023	виконано
3	Визначення вимог до додатку	25 лютого 2023 – 8 березня 2023	виконано
4	Вибір інструментальних засобів	8 березня 2023 – 10 березня 2023	виконано
5	Визначення варіантів використання	10 березня 2023 – 19 березня 2023	виконано
6	Проектування та реалізація алгоритмів системи	19 березня 2023 – 29 березня 2023	виконано
7	Написання програмного коду	29 березня 2023 – 2 травня 2023	виконано
8	Тестування роботи програмного продукту	2 травня 2023 – 11 травня 2023	виконано
9	Оформлення пояснювальної записки	11 травня 2023 – 30 травня 2023	виконано
10	Попередній захист	5 травня 2023	виконано

РЕФЕРАТ

Випускна робота бакалавра складається з розробки "Колекційно карткова гра з елементами РПГ та покроковою боївкою" та пояснювальної записки. Пояснювальна записка до випускної роботи містить 92 сторінки, 19 ілюстрацій, 1 додаток та 15 літературних джерел.

Метою роботи є розробка за допомогою рушія Unity гри у жанрі колекційна карткова гра.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об'єктної структури системи (діаграма класів). Також наводиться результати тестування системи.

КЛЮЧОВІ СЛОВА: Колекційна карткова гра.

					ІПЗ.КР.Б – 121 – 23 - ІПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		М.П. Коптяєв			Пояснювальна записка "Колекційно карткова гра з елементами РПГ та покроковою боївкою"	Літ.		Арк.	Аркушів
Керівник.		О.Г. Чижмотря						4	92
						Житомирська політехніка, група ІПЗк-20-1			
Реценз.		С.М. Кравченко							
Зав. Каф..		А.В. Морозов							

ABSTRACT

The bachelor's thesis consists of the development of the "A collectible card game with RPG elements and turn-based combat" and an explanatory note. The explanatory note to the final thesis contains 92 pages, 19 illustrations, 1 appendix and 15 literary sources.

The purpose of the work is to develop using the Unity engine a game in the genre of a collectible card game.

The paper defines the main tasks for the development of the system, and analyzes the analogues of the developed system. Scenarios of the software complex, the general structure of the software complex, a description of the interaction algorithms of individual system modules, a description of the object structure of the system (class diagram) are given. The results of system testing are also provided.

KEYWORDS: Collectable card game.

		<i>М.П. Коптяєв</i>			ІІЗ.КР.Б – 121 – 23 - ІІЗ	<i>Арк.</i>
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>5</i>

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1. Технічне завдання на розробку системи.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури.....	22
1.4. Інструментальні засоби реалізації програмного продукту	24
Висновки до першого розділу	26
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ	28
2.1. Визначення варіантів використання системи	28
2.2. Проектування та реалізація алгоритмів роботи системи	33
2.3. Реалізація функціоналу додатку	34
Висновки до другого розділу	43
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ.....	45
3.1. Порядок встановлення та налаштування системи.	45
3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з грою.	45
3.3. Тестування роботи системи	50
Висновки до третього розділу	50
ВИСНОВКИ	52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53

ВСТУП

Актуальність теми: на даний момент дуже великої популярності набувають ігри в жанрі «рогаликів», вони досить прості у своїй основі та при цьому вони дуже цікаві. Також не меншою популярністю користуються колекційно-карткові ігри котрих також не мало, в тому числі і рогаликів як об'єднання цих жанрів. Водночас існує багато ігор з відкритими або коридорними локаціями які можна дослідити, тому об'єднати це все разом виглядає досить привабливо.

Мета даної роботи: розробка та реалізація гри "Munchkin Rogue-like". Ця гра об'єднає у собі жанр рогалика та колекційної карткової гри, дозволяючи гравцям досліджувати відкриті та коридорні локації, збирати карти та битися з монстрами.

Об'єкт дослідження: є процес розробки гри "Munchkin Rogue-like" та взаємодія гравців з грою. Розробка гри спрямована на створення цікавої та захоплюючої гри для гравців.

Предмет дослідження: є функціональні можливості гри "Munchkin Rogue-like". Гра буде включати такі функції, як вибір персонажа, збір карт, бій з монстрами та інші функції, необхідні для успішного проходження гри.

Методи дослідження: у процесі розробки гри "Munchkin Rogue-like" будуть використані наступні методи дослідження: аналіз вимог до функціоналу гри, проектування архітектури та інтерфейсу гри, розробка та тестування програмного забезпечення. Крім того, для забезпечення коректності та ефективності гри будуть використані методи тестування.

Загальний результат дослідження буде представлений у вигляді функціональної гри "Munchkin Rogue-like", яка забезпечить цікаву та захоплюючу гру для гравців, що люблять жанр рогаликів та колекційних карткових ігор.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IOS — iPhone Operating System (Операційна система Айфон).

ПК, РС — персональний комп'ютер.

ПЗ — програмне забезпечення.

C# — об'єктно-орієнтована мова програмування.

Ps — програма для редагування фотографії та малювання (в даній роботі використовувалася для малювання спрайтів)

2D — представлення об'єкта в двох просторових вимірах.

Спрайти — різноманітні 2D графічні об'єкти, що використовуються в Unity.

ООП — об'єктно-орієнтоване програмування, одна із парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

RogueLike (рогалик) — жанр ігор в якому основною особливістю являється покроковість та перманентна смерть персонажа, що означає початок гри спочатку (в подальшому рогалик).

ККІ — Колекційно карткові ігри.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Технічне завдання на розробку системи

Основною метою розробки гри є створення захоплюючого та емоційного ігрового досвіду для гравця. Колекційно карткова гра з елементами РПГ та покроковою боївкою, далі Munchkin Rogue-like повинна надати можливість гравцеві відчувати себе у ролі головного героя, зануритися в ігровий світ і взаємодіяти з ним [9]. Необхідно розробити додаток, який наблизить геймплей до реалістичного.

Для реалізації поставленого завдання, основними етапами є:

1. Розробити дизайн користувацького інтерфейсу.
 - 1.1. Створити загальну концепцію дизайну користувацького інтерфейсу.
 - 1.2. Зверстати та застосувати елементи інтерфейсу за допомогою Unity:
 - 1.2.1. Головне меню
 - 1.2.2. Панель меню паузи
2. Проектування та розробка основних ігрових модулів
 - 2.1. Модуль керування колодою карт
 - 2.2. Модуль ворогів та битв

Бізнес-вимоги

1. Основні цілі: проект створюється з метою розважити користувача та розвинути його навички логічного мислення за допомогою створення найефективнішої збірки персонажа.
2. Представлення проекту: проект буде реалізовано у вигляді гри в жанрі Колекційна карткова гра с елементами РПГ та Roguelike.

Вимоги користувачів

1. Користувач

Користувач буде мати можливість битися з різноманітними монстрами, збирати відповідну колоду з обмеженої кількості карт, споряджати свого персонажа відповідно до зібраної колоди або ж навпаки збирати колоду

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

відповідно до спорядження, робити покупки різноманітних товарів у магазині, що будуть зустрічатися на карті.

Функціональні вимоги

1. Геймплей та боївка: Гра повинна надавати цікавий ігровий досвід користувачеві за допомогою боївки як у Колекційно карткових іграх та гравець не повинен втрачати інтерес до гри поміж боїв.
2. Повна свобода дій: Гра повинна забезпечувати повну свободу дій гравцеві на стільки наскільки це можливо у даному жанрі.
3. Можливість збереження інформації: Система повинна зберігати інформацію про ігровий процес гравця [\[10\]](#).

Не функціональні вимоги

1. Сприйняття

Користувацький інтерфейс не повинен бути надто перенавантажений різноманітними елементами або ж розкинутий по різноманітним вкладкам, він повинен бути зрозумілий та інтуїтивний. Освоєння у грі та знайомство з усіма механіками не повинно займати більше ніж 2 години;

2. Надійність

В грі не повинно бути неполадок, що будуть заважати гравцю та руйнувати його ігровий досвід;

3. Кросплатформенність

Система повинна працювати не лише на ПК а й на інших платформах таких як: IOS, Android;

Системні вимоги

1. Система повинна задовольняти вимогам на комп'ютері, що знаходиться внаступній мінімальній комплектації:
 - 1.1. 8 ГБ пам'яті
 - 1.2. Процесор з тактовою частотою 3.1 ГГц
2. Операційна система Windows 10

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Антивимоги

1. Не використовує інформацію, заборонену нормами моралі та законодавством;
2. Не використовує інформацію, що є недійсною або може зашкодити психіці користувачів.

Бізнес логіка

1. Реклама нашого сайту — на інших сайтах, умова: власні кошти;
2. Продукти і послуги:
 - 2.1. Проект надає розважальні послуги;
 - 2.2. Продукт є платним.

Вимоги до порядку впровадження:

1. розробка прототипу для замовника та обговорення з ним;
2. впровадження проекту має відбуватися поетапно, включаючи випуск альфа та бета версій;
3. по закінченню відбудеться презентація проекту;
4. термін випуску альфа та бета версій попередньо не встановлено
попередньо випуск кінцевого варіанту проекту призначено на 17.10.2023 року.

Експлуатаційні вимоги

1. На початковому етапі планується купівлю підписки на сервіси, такі як GooglePlay, AppStore, Steam для того щоб стати розробником на платформі та мати змогу завантажити гру на ці сервіси;
2. Планується в перші дні релізу слідкувати за відгуками та якщо будуть помітні помилки одразу все виправляти.

1.2. Аналіз аналогів програмного продукту

В наш час люди все більше використовують електронні пристрої такі як комп'ютер або ж мобільний телефон у повсякденному житті. За допомогою цих пристроїв люди задовольняють свої потреби в тому числі і потреби у розвагах та відпочинку.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

Є досить багато роґаліків, що можуть відповідати різним ігровим жанрам. Ігрова програма «MunchkinRogueLike» повинна бути схожа за своїм функціоналом до роґаліків, що будують свій ігровий процес на картах та їх колекціонуванні, такий піджанр ігор називають ККІ (Колекційно-карткові ігри).

Основним прикладом ККІ Роґаліків є «Slay the spire». Це рольова гра, розроблена американською студією «MegaCrit». Здобула свою основну популярність у 2018 році, через деякий час після того як вона вийшла у ранньому доступі в цифровому магазині ігрової продукції «Steam». Основний ігровий процес побудований на правильному використанні карт та менеджменту мани.



Рисунок 1.1. Гра Slay the spire

Гравець починає зі стандартним набором карт та здібностей а по мірі проходження гри, отримує якісь нові можливості. Завдяки випадковій генерації противників на рівнях кожна нова ігрова сесія представляє собою новий ігровий досвід. Завдяки приналежності до жанру не тільки ККІ, а ще й роґаліків сам ігровий процес досить складний і випробовує гравця на витривалість. А ось елемент ККІ вносить ще більший елемент випадковості, що додає більшої цікавості ігровому процесу.

Основний ігровий процес складається з перемоги над різноманітними монстрами і поповнення колоди новими картами або ж покращення свого героя

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

шляхом зміни ігрової моделі цього героя, або ж його частин. Також основною частиною ігрового процесу є перманентна смерть ігрового персонажу та початок гри спочатку, така риса жанру перейшла якраз від рогаликів.

Взагалі відомих ігор такого жанру небагато, бо в основному займаються такими іграми лише інді-студії (незалежні студії), яких майже ніхто не знає. Як приклад, досить цікавою є «Monster train», де ігровий процес трохи змінюється і потрібно виставляти героїв на різні поверхи вагонів та захищати основну секцію вагона.

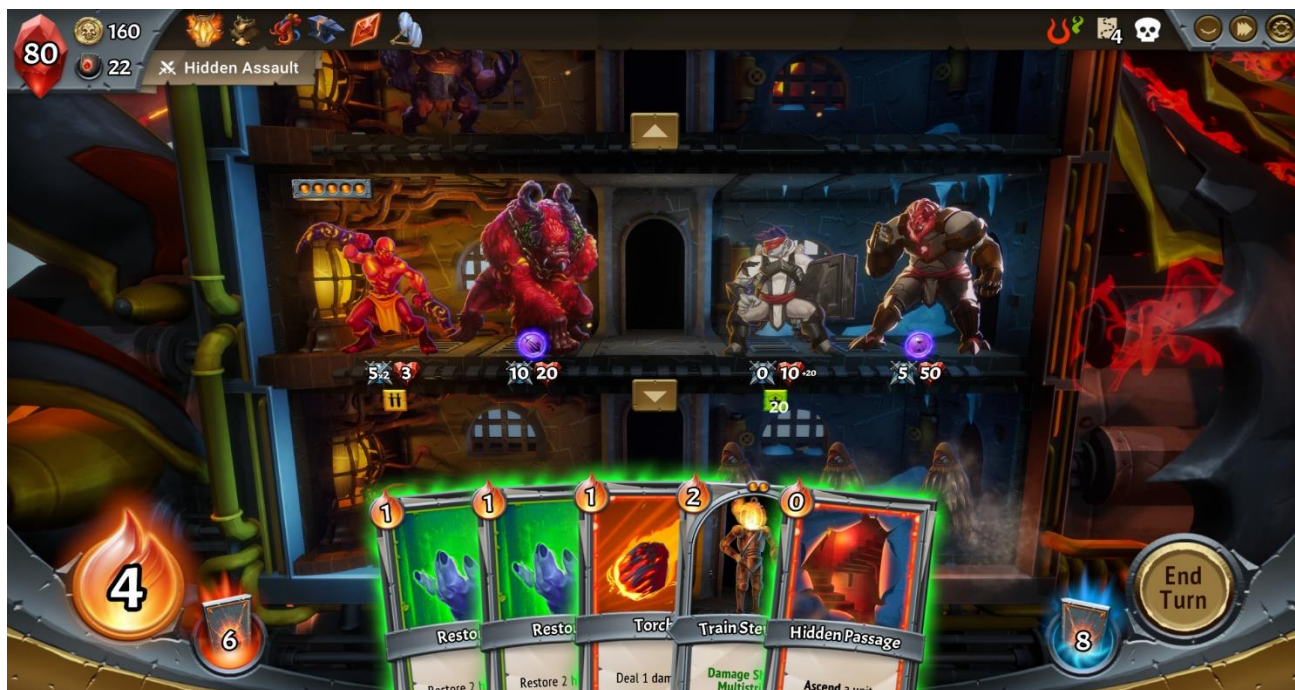


Рисунок 1.2. Гра Monster train

У "Monster train" користувач грає за останню оборону Пекла, керуючи потужним потягом, що виконує місію захисту Серця Пекла від сил вторгнення. Гра поєднує в собі елементи баштової оборони, карткових ігор та механіки roguelike.

Мета полягає в тому, щоб захистити три поверхи поїзда від хвиль ворогів, стратегічно розміщуючи юнітів, заклинання та артефакти вздовж шляху поїзда. Кожен поверх поїзда являє собою бойову зону, де зіштовхуються юніти гравця та ворожі сили. Битви відбуваються покроково, що дозволяє гравцеві приймати тактичні рішення, щоб максимізувати свої шанси на перемогу.

Протягом гри гравці будуть збирати колоду, обираючи карти різних кланів, кожен з яких має свій унікальний стиль гри та здібності. Карти можна покращувати, а також отримувати нові в міру проходження гри. Створення сильної колоди, яка добре синергує з обраним гравцем кланом, є запорукою успіху.

Одним з цікавих аспектів "Monster train" є вертикальність битв. Юніти та заклинання можуть переміщатися між поверхами, створюючи стратегічні можливості та виклики. Гравці повинні ретельно управляти своїми ресурсами, приймати рішення про те, яких юнітів розгорнути, і вирішувати, як найкраще використовувати свої заклинання і артефакти, щоб подолати натиск ворога.

У грі реалізована система прогресування, що означає, що кожне проходження гри є унікальним, з процедурно згенерованими рівнями, подіями та зустрічами з ворогами. У міру проходження гравці зустрічатимуть все більш складних ворогів і босів, відкриваючи нові карти, клани та здібності.

Основною особливістю даної гри являється можливість «синергій» різних героїв, тобто покращення характеристик героїв якщо їх щось об'єднує.

Також це доволі складна гра, що змушує гравця думати перш ніж ставити героїв на поле, і які карти коли використовувати, а ще більша складність отримується дякуючи трьом ігровим полям.

А ось основним представником ККІ є «Hearthstone». На перший погляд досить легка гра але якщо поглибитися у ігровий процес то виявляється, що це досить складна гра, що потребує від гравця не лише «гострого» розуму але й удачі.

Дія гри розгортається у всесвіті Warcraft і включає в себе різноманітних героїв, кожен з яких має свої унікальні здібності та карти.

У Hearthstone гравці створюють колоди карт, що представляють різні заклинання, міньйонів та зброю, а потім змагаються один проти одного, використовуючи ці колоди. Мета — звести здоров'я суперника до нуля, стратегічно використовуючи карти, щоб контролювати ігрову дошку і перехитрити опонентів.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

До прикладу декілька основних моментів гри, чим вона може зацікавити гравця:

- Режими гри: Hearthstone пропонує кілька ігрових режимів. Основним режимом є рейтингова гра, де гравці змагаються з іншими гравцями з аналогічним рівнем майстерності, щоб піднятися по сходах і заробити нагороди. Існує також Casual Play для більш спокійних матчів, Arena, де гравці створюють колоду з серії випадкових карт, і Tavern Brawl, режим, що змінюється щотижня, з унікальними правилами та випробуваннями;
- Набори карт і доповнення: до Hearthstone випускаються різноманітні набори карт і доповнення, які вводять у гру нові карти, механіки та теми. Щороку відбувається нова Стандартна ротація, під час якої старі набори карт виводяться зі змагального формату, щоб зберегти гру свіжою та збалансованою. Гравці можуть отримувати карти з колод, крафту або беручи участь в ігрових подіях;
- Класи та герої: Hearthstone має десять різних класів, кожен з яких представляє певного героя чи лиходія Warcraft. Кожен клас має власну унікальну силу героя та специфічні карти, які визначають його стиль гри та стратегію. Це такі класи: Друїд, Мисливець, Маг, Паладин, Священик, Розбійник, Шаман, Чаклун, Воїн і Мисливець на демонів;
- Змагальна сцена: Hearthstone має процвітаючу змагальну сцену, з професійними гравцями та організованими турнірами. Blizzard проводить офіційні турніри, такі як Hearthstone Masters Tour та Hearthstone Grandmasters, в яких беруть участь найкращі гравці з усього світу. Також проводяться заходи, організовані спільнотою, та сторонні турніри;
- Соціальні функції: Hearthstone включає соціальні функції, які дозволяють гравцям спілкуватися з друзями, викликати їх на матчі та спостерігати за їхніми іграми. Гравці також можуть приєднуватися або створювати Fireside Gatherings — локальні зустрічі, де люди можуть грати в Hearthstone разом особисто;

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

- Free-to-Play модель: Hearthstone працює за моделлю free-to-play, що дозволяє гравцям завантажувати і грати в гру безкоштовно. Хоча гравці можуть здобувати карти в процесі гри, є також можливість купувати колоди карт та ігрові предмети за реальні гроші;
- Історія та тематика: Дія Hearthstone відбувається у всесвіті Warcraft, відомому своєю багатою історією та культовими персонажами. Гра включає в себе різні локації, раси та події з франшизи Warcraft, пропонуючи знайомий та захоплюючий досвід для шанувальників серії.

Hearthstone здобула значну кількість прихильників і стала однією з найпопулярніших цифрових карткових ігор у всьому світі, з регулярними доповненнями та оновленнями, які роблять ігровий процес свіжим і захоплюючим.

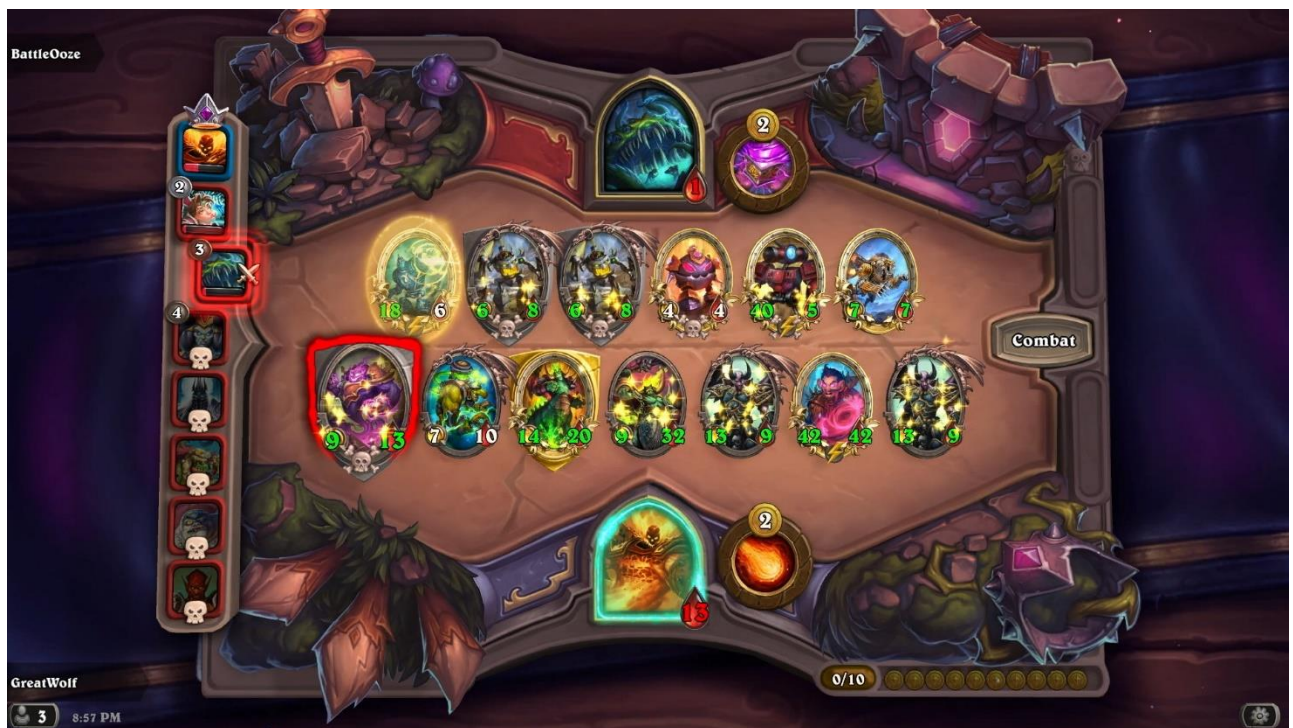


Рисунок 1.3. Гра Hearthstone

Саме те, що ця гра потребує розуму та удачі, ця гра викликає таку цікавість у аудиторії. У світі в неї грає більше мільйона гравців і також вона має не маленький постійний онлайн.

А ось основним представником рогаликів є така гра як «Enter the gungeon». Enter the Gungeon — це відеогра в жанрі рольової гри, розроблена компанією

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

Dodge Roll і видана Devolver Digital. Вона вийшла у 2016 році для різних платформ, включаючи Microsoft Windows, macOS, PlayStation 4, Xbox One і Nintendo Switch.

У Enter the Gungeon гравці керують одним із кількох озброєних до зубів шукачів пригод, які досліджують Gungeon — підземелля, що постійно змінюється і процедурно генерується, наповнене ворогами, пастками та скарбами. Мета полягає в тому, щоб досягти останнього рівня Підземелля і протистояти босу, відомому як Драгун.

Ігровий процес Enter the Gungeon обертається навколо швидких бойових дій та досліджень. Гравці переміщуються кімнатами, беруть участь в інтенсивних перестрілках і ухиляються від куль. У грі представлений широкий вибір фантастичної зброї та предметів, кожен з яких має унікальні властивості та ефекти. Гравці також можуть розкривати секрети, відкривати нових персонажів і знаходити різні синергії між різними видами зброї та предметами.

Enter the Gungeon відома своєю складністю, хитромудрими ворожими схемами та розумними сутичками з босами. Кожен запуск є рандомізованим, пропонуючи свіжий досвід з кожним проходженням. Після смерті гравці втрачають свій прогрес і повинні починати з початку, але вони можуть збирати "Кредити Гегемонії" під час проходження, щоб купувати постійні поліпшення і відкривати нові предмети і персонажів.

Гра також підтримує кооперативний багатокористувацький режим, що дозволяє гравцям об'єднуватися з друзями в локальній або мережевій грі для спільного подолання Гунгена.

Enter the Gungeon отримала схвальні відгуки критиків за захоплюючий ігровий процес, стильний піксельний арт і широкий вибір зброї та предметів. У неї з'явилася віддана фан-база, і її хвалять за реіграбельність і глибину. Сам ігровий процес складається з зачистки кімнат від монстрів за допомогою різної зброї. А елемент випадкової генерації кімнат робить кожну ігрову сесію унікальною.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 1.4. Гра Enter the gungeon

Одним з прикладів також виступає Magic: The Gathering Arena. Magic: The Gathering Arena — це цифрова адаптація популярної фізичної карткової гри Magic: The Gathering (MTG). Розроблена компанією Wizards of the Coast, вона була запущена у 2019 році та здобула значну кількість прихильників як серед казуальних, так і серед змагальних гравців.

Magic: The Gathering Arena точно відтворює стратегічний ігровий процес фізичної карткової гри в цифровому форматі. Гравці будують колоди, використовуючи віртуальні карти з величезної колекції, кожна з яких представляє різних істот, заклинання, артефакти та чари. Ці карти організовані в колоди, що містять мінімум 60 карт, і гравці можуть налаштовувати свої колоди відповідно до свого стилю гри та стратегії.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18



Рисунок 1.5. Гра Magic: The Gathering Arena

Гра пропонує різні ігрові режими, серед яких:

- Гра: цей режим дозволяє гравцям зіткнутися з опонентами у випадкових матчах, використовуючи свої власні колоди.
- Рейтинговий: У цьому режимі гравці змагаються з іншими в системі сходів, щоб заробити звання і піднятися по змагальній таблиці лідерів. Він забезпечує більш складне та структуроване середовище.
- Драфт: це обмежений формат, в якому гравці збирають свої колоди, вибираючи карти з бустерів. Magic: The Gathering Arena відтворює цей досвід онлайн, дозволяючи гравцям брати участь у драфті та створювати колоди з обмеженого набору карт.
- Закриті: Подібно до драфту, закриті події вимагають від гравців створення колод з обмеженого набору карт, але замість драфту вони отримують попередні набори, що містять бустер-паки. Закриті події — хороший спосіб швидко зануритися в гру та випробувати основні механіки.

Magic: The Gathering Arena постійно поповнюється новими наборами карт, до яких входять карти з останніх розширень і наборів, випущених у фізичній грі. Ці цифрові набори часто збігаються зі стандартним форматом, що представляє собою добірку останніх наборів, які формують основу змагального середовища.

Гра також включає в себе Історичний формат, який охоплює більший пул карт з усієї історії Магії.

Окрім звичайного ігрового процесу, Magic: The Gathering Arena пропонує гравцям можливість брати участь у спеціальних подіях, режимах гри з обмеженим часом та сезонних подіях з унікальними нагородами. У грі передбачена система прогресування, де гравці можуть заробляти нагороди, відкривати нові карти і збирати свою колекцію, виконуючи щоденні та щотижневі квести.

Magic: The Gathering Arena підтримує як казуальну, так і змагальну гру, а також має кіберспортивну сцену з офіційними турнірами та чемпіонатами. Інтерфейс гри зручний, з інтуїтивно зрозумілим управлінням і корисними підказками, які допоможуть новим гравцям вивчити правила і механіку гри.

Загалом, Magic: The Gathering Arena забезпечує захоплюючий цифровий досвід культової карткової гри, дозволяючи гравцям насолоджуватися стратегічною глибиною та азартом Magic: The Gathering у зручному онлайн-форматі.

ККІ також можуть бути частиною основної гри, а також бути на основі великих ігор. Одним і прикладів другого типу є Legends of Runeterra.

Legends of Runeterra — це безкоштовна цифрова колекційна карткова гра, розроблена та видана Riot Games. Дія гри розгортається у всесвіті League of Legends, де представлені чемпіони, персонажі та історії з популярної багатокористувацької онлайн-гри МОБА (англ. Multiplayer Online Battle Arena).

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

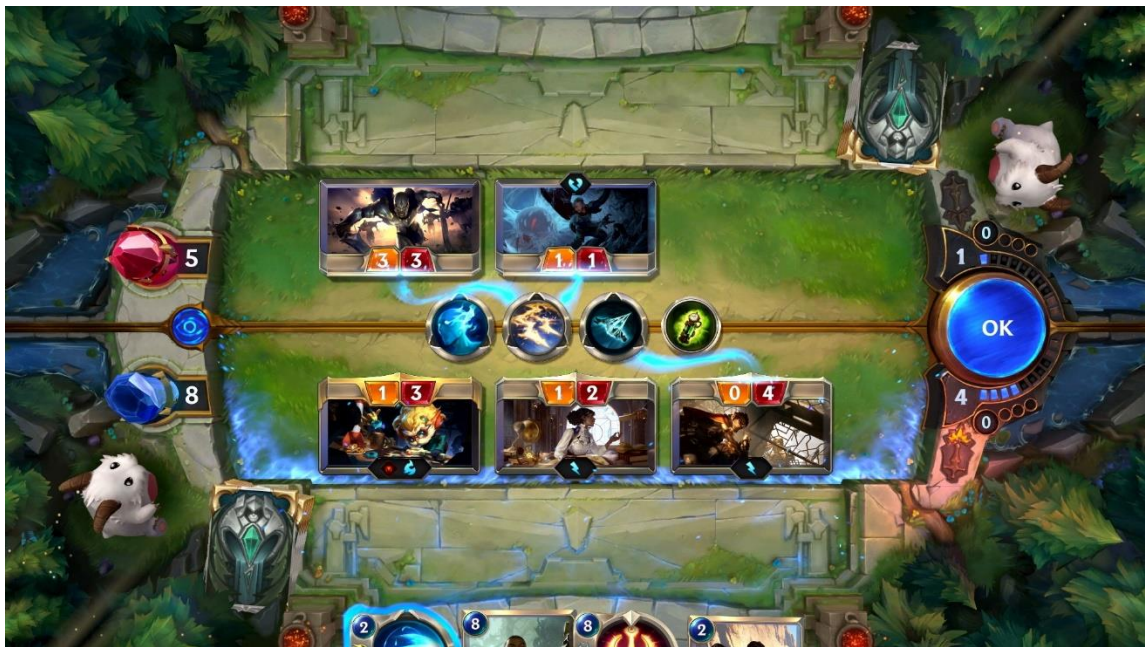


Рисунок 1.6. Гра Legends of Runeterra

У Legends of Runeterra гравці беруть участь у покрокових битвах один проти одного, використовуючи власні колоди карт. Мета полягає в стратегічному розгортанні юнітів, заклинань та інших карт, щоб звести Нексус (еквівалент загальної кількості життів гравця) опонента до нуля.

Ось деякі ключові особливості та аспекти гри:

- Гра розгортається навколо культових чемпіонів з League of Legends, кожен з яких має свої унікальні здібності та стиль гри. Чемпіони є найпотужнішими картами в грі і часто служать центральним елементом колоди, забезпечуючи сильні ефекти та синергію.
- Гравці будують свої колоди, використовуючи карти з різних регіонів Рунетери, кожна з яких представляє певну фракцію або тему. Кожна колода може містити максимум 40 карт і може поєднувати до двох регіонів, що відкриває широкий спектр стратегічних можливостей.
- Легенди Рунетери (Legends of Runeterra) робить акцент на прийнятті стратегічних рішень та управлінні ресурсами. Гравці по черзі розігрують карти, атакують і захищаються. У грі використовується система атак по черзі, коли гравці по черзі атакують своїми юнітами і захищаються від атак суперника.

- У грі представлена унікальна система мани. У гравців є мана заклинань і звичайна мана. Мана заклинань може бути збережена і використовуватися виключно для створення заклинань, в той час як звичайна мана використовується для гри юнітами та іншими картами, які не є заклинаннями. Ця система додає глибини в ігровий процес і дозволяє будувати більш гнучкі стратегії.
- Кожен хід розділений на кілька раундів, що дозволяє гравцям реагувати на дії суперника в реальному часі. Ця інтерактивна структура ходів додає додатковий рівень стратегії та залучення гравців.
- Легенди Рунетери регулярно випускає нові набори карт і доповнення, розширюючи пул карт і надаючи гравцям нові можливості для створення колод і стратегій. Ці оновлення привносять у гру свіжий контент, чемпіонів та ігрові механіки.
- Гра пропонує рейтингову систему сходів, де гравці можуть змагатися з іншими, щоб підніматися по кар'єрних сходах і заробляти нагороди. Існують також події з обмеженим часом, спеціальні ігрові режими та сезонні нагороди, щоб урізноманітнити ігровий процес і зробити його цікавим.

Legends of Runeterra вирізняється доступним ігровим процесом, глибокою стратегією та високою виробничою цінністю. Вона забезпечує візуально привабливий та захоплюючий досвід як для фанатів League of Legends, так і для новачків у жанрі цифрових карткових ігор.

1.3. Вибір архітектури

Для розробки даної системи використовувався найпопулярніший патерн, а саме MVC [\[1,11\]](#).

Model-View-Controller (MVC) є однією з найпоширеніших архітектурних парадигм для розробки програмного забезпечення. Ця парадигма дозволяє розділити логіку додатку на три основні компоненти: Модель (Model), Представлення (View) та Контролер (Controller). Кожен з цих компонентів

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

виконує свої відповідні функції, що забезпечують розділення відповідальності та полегшують розробку та підтримку програмного забезпечення.

Основні принципи MVC:

1. Модель (Model): Це компонент, який відповідає за представлення даних та бізнес-логіку додатку. Він зберігає дані, здійснює операції з ними, обробляє запити та повідомлення від контролера або інших джерел. Модель може також надсилати повідомлення або сповіщення представленню про зміни в даних, щоб оновити їх.
2. Представлення (View): Це компонент, який відповідає за візуальне представлення даних моделі та взаємодію з користувачем. Воно відображає дані моделі у вигляді графічного інтерфейсу, веб-сторінки або будь-якого іншого способу представлення. Представлення може також надсилати повідомлення контролеру про події, які відбуваються в інтерфейсі (наприклад, натискання кнопки).
3. Контролер (Controller): Це компонент, який відповідає за прийняття вхідних подій від користувача та взаємодію з моделлю та представленням. Він обробляє події, спрацьовує запити користувача, оновлює модель та оновлює представлення. Контролер служить посередником між моделлю та представленням, дозволяючи їм взаємодіяти безпосередньо один з одним.

Model-View-Controller (MVC) є шаблоном проектування програмного забезпечення, який дозволяє ефективно організувати структуру коду та поділити його на різні компоненти з різними відповідальностями. MVC використовується в багатьох фреймворках та платформах, включаючи Unity.

Використання MVC для розробки ігрових двигунів, таких як Unity, має кілька переваг:

1. Розділення відповідальностей: MVC дозволяє розділити логіку додатку на три основні компоненти: модель (Model), представлення (View) та контролер (Controller). Це дозволяє покращити читабельність, підтримку коду та зручність управління. Наприклад, модель відповідає за дані та

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

бізнес-логіку, представлення — за відображення інтерфейсу користувача, а контролер — за обробку введення та взаємодію з моделлю та представленням.

2. Модульність: Кожна частина MVC може бути розроблена незалежно від інших, що спрощує роботу багатій команди розробників або сприяє повторному використанню компонентів. Наприклад, дизайнер може працювати над виглядом гри (представленням), програміст може розробляти логіку гри (модель), а розробник інтерфейсу може створювати взаємодію з користувачем (контролер). Всі ці компоненти можуть легко комбінуватися та співпрацювати без необхідності залежності один від одного.
3. Тестування: Завдяки розділенню на модель, представлення та контролер, кожен компонент може бути легко тестований окремо. Це полегшує написання автоматизованих тестів, оскільки можна перевіряти окремо функціональність кожного компонента. Модель може бути перевірена на правильність розрахунків та логіку, представлення — на правильне відображення інтерфейсу користувача, а контролер - на коректну обробку введення.
4. Підтримка розширення та змін: Завдяки розділенню компонентів, зміни в одному з них не повинні впливати на інші. Наприклад, якщо потрібно змінити спосіб відображення гри, можна змінити лише представлення, не змінюючи модель або контролер. Це робить систему більш гнучкою та легко змінюваною.

Отже, використання MVC у рушії Unity дозволяє краще організувати структуру коду, полегшує розробку, тестування та підтримку, а також сприяє модульності та зручності співпраці розробників.

1.4. Інструментальні засоби реалізації програмного продукту

Очевидним рішенням даної задачі є розробка ігрового додатку з використанням ігрового рушія, а саме рушій Unity, багатоплатформений інструмент для розробки дво- та тривимірних додатків та ігор, що працює на

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

операційних системах Windows (Досить простий для нового користувача та досить перспективний у своєму вивченні). Середовище Photoshop, яке використовувалось для малювання графіки та створення анімацій. А також мову програмування C#, що є об'єктно-орієнтованою мовою програмування.

Технології об'єктно-орієнтованого програмування, 2D анімування, скриптингу зменшили кількість зв'язків між окремими частинами проекту, що дозволило зменшити час розробки проекту та покращити його функціонал та зовнішній вигляд.

Для розробки вибране середовище Unity 2021.3.12f1. Движок Unity підійде для новачків і любителів [\[12\]](#).

Unity — багатоплатформений інструмент для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows. Програмне забезпечення, створене за допомогою Unity, підтримують DirectX та OpenGL.

Технічні характеристики:

- Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо у редакторі.
- Система успадкування об'єктів;
- Підтримка імпортування великої кількості форматів файлів;
- Вбудований конструктор локацій

Adobe Photoshop має досить зручний функціонал для малювання графіки у піксель артi, що дозволяє зручно працювати у будь-яких об'ємах. Була вибрана саме ця програма через те, що вона дозволяє не тільки намалювати елементи сцени, по типу ландшафту, героїв, ворогів, а й анімувати їх одразу ж у самій програмі за допомогою шкали часу. В свою чергу це дозволяло перенести у проект Unity уже готовий спрайт або ж анімацію цілком.

А вже за допомогою самих інструментів Unity підключити ці анімації до скриптів. Також у самому редакторі є система наслідування, що дозволяє досить зручно налаштувати сцену через ієрархію об'єктів.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

При імпорті спрайтів у рушій користувач може налаштувати спрайт як зручно, його розмір, положення, Alpha-канал. А також підключити до цього спрайту будь-який скрипт, що дозволить змінити його уже шляхом програмування.

Мова програмування C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтанутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java [3,4]. Мова має сувору статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# підтримує суворо типізовані неявні оголошення змінних з ключовими словами var і неявно типізовані мисиви з ключовим словом new [], за яким слідує ініціалізатор колекції.

Мова C# не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції.

Для редагування C# використовувалось досить зручне середовище Visual studio code, що дозволяє встановити деякі розширення для самого движка Unity і легше орудувати скриптами [13].

А для більш розвернутої роботи зі шрифтами використовувався асет TextMeshPro.

Висновки до першого розділу

У даному розділі було проаналізовано проблематику, методи та засоби вирішення задачі створення захоплюючої та емоційної ігрової гри. Було

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

визначено основні бізнес- та функціональні вимоги до проекту, а також системні та не функціональні вимоги.

З метою розробки ефективної гри, було визначено основні етапи проектування та розробки системи, включаючи розробку дизайну користувацького інтерфейсу та ігрових модулів. Бізнес-логіка проекту передбачає створення розважальних послуг та продаж проекту як платного продукту.

Не функціональні вимоги до системи передбачають зрозумілий та інтуїтивний користувацький інтерфейс, відсутність неполадок в грі, а також кросплатформеність. Системні вимоги передбачають наявність комп'ютера з 8 ГБ пам'яті та процесором з тактовою частотою 3.1 ГГц та операційною системою Windows 10.

Для створення ігрового процесу буде використано піджанр ігор ККІ (Колекційно-карткові ігри), схожий за своїм функціоналом до рогаликів, що будують свій ігровий процес на картах та їх колекціюванні. Основним прикладом цього піджанру є «Slay the spire».

В цілому, розробка ігрової програми "MunchkinRogueLike" має на меті створити захоплюючий та емоційний ігровий досвід для гравця, який буде базуватись на колекціюванні карт та боях з монстрами.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ

2.1. Визначення варіантів використання системи

Даний підрозділ розглядає основні функціональні можливості системи ігрової програми "MunchkinRogueLike" і описує різні сценарії використання, які задовольняють потреби користувачів.

Це включає в себе нову гру, продовження гри, взаємодію з оточенням та ворогами. Гравець може зібрати колоду для різноманітних битв.

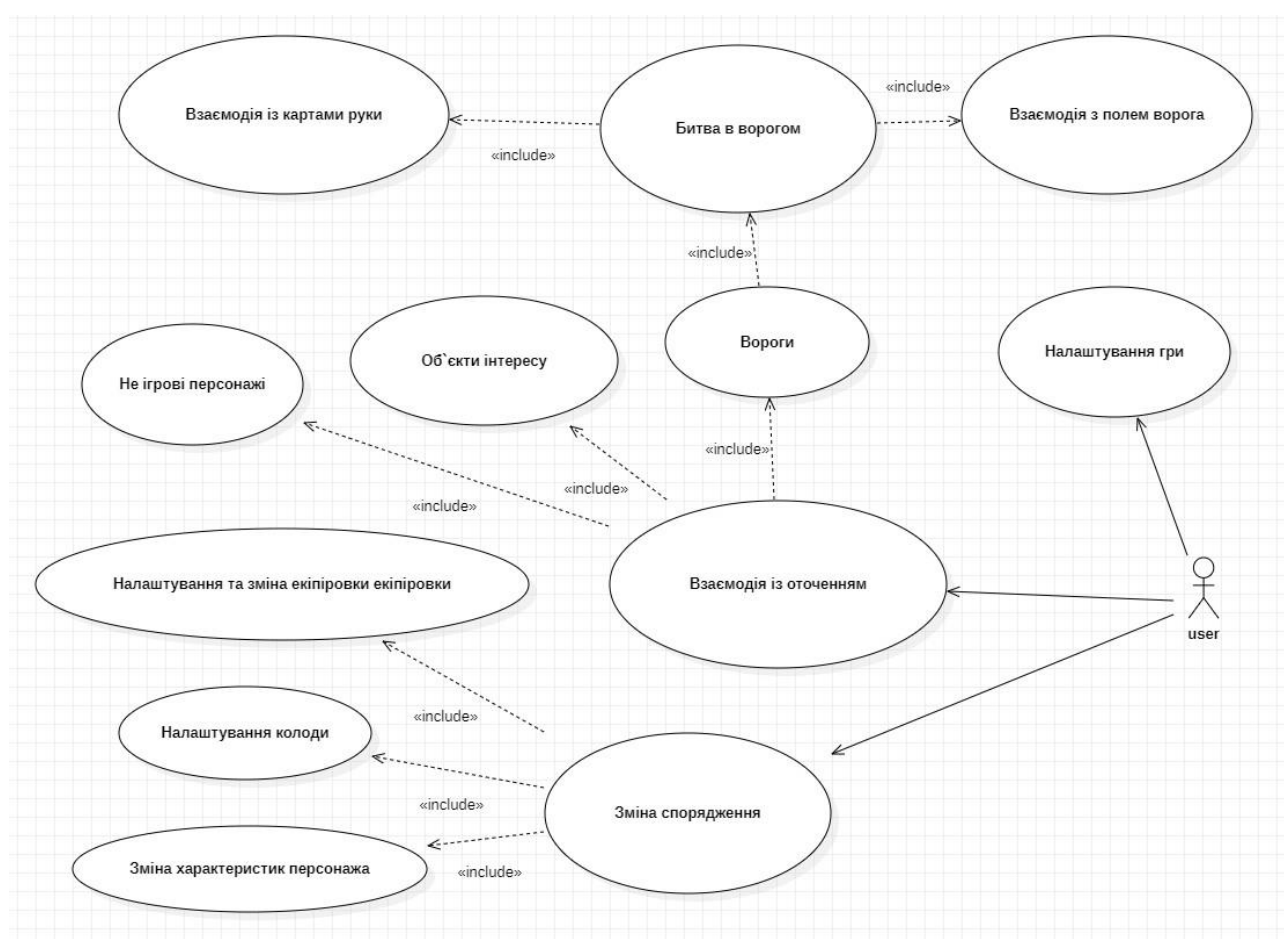


Рисунок 2.1. Діаграма варіантів використання

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать реалізацію програмного комплексу системи. На Рисунок2.2. наведено діаграму класів, які використовуються для керування ігровою сценою (світом).

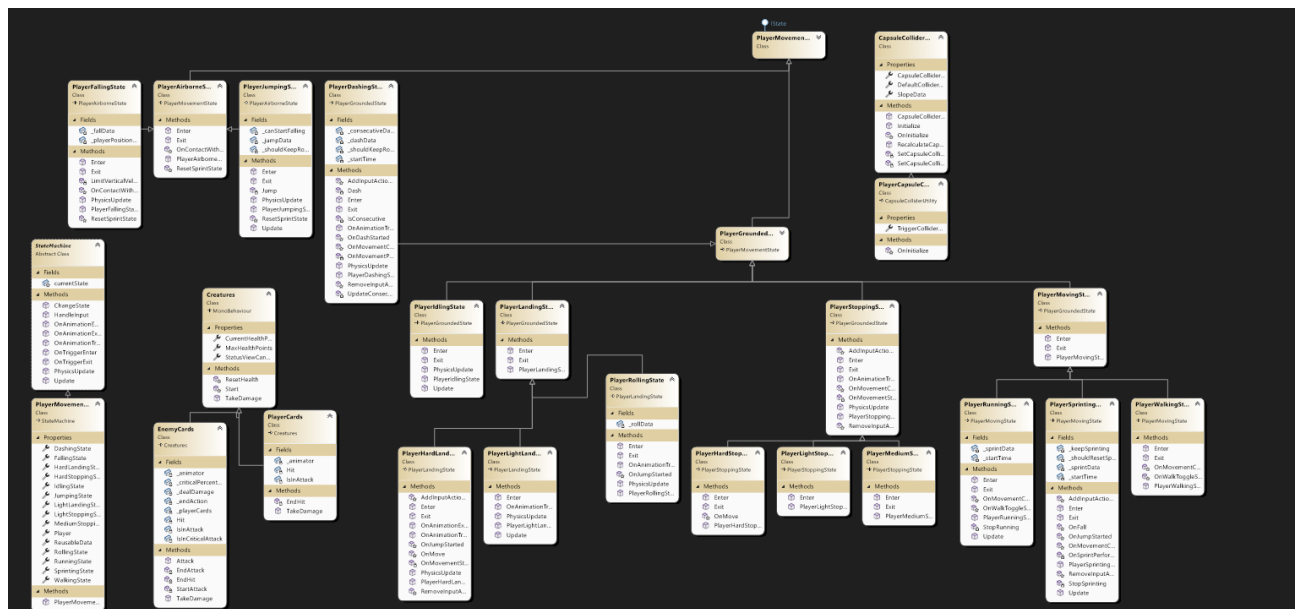


Рисунок 2.2. Діаграма класів проекту

В даному проекті в основному використовувалися [5] такі патерни як «Одинак», «Машина станів» та «Пул об'єктів».

Патерн "Одинак" (Singleton) [7] — це структурний патерн проектування, який використовується для створення класу з одним єдиним екземпляром, що забезпечує глобальний доступ до нього. Цей патерн дозволяє гарантувати, що клас має тільки один екземпляр, та забезпечує механізм для доступу до нього з

будь-якої точки програми. Часто використовується для управління ресурсами, що мають єдиний доступ, наприклад, підключення до баз даних або файлові дескриптори.

Патерн "Машина станів" (State) [6] — це поведінковий патерн проектування, який дозволяє замінити велику кількість умовних операторів на декілька станів об'єкта, що залежать один від одного. Об'єкт може змінювати свій стан внаслідок своєї поведінки. Цей патерн дозволяє покращити читабельність та підтримку коду, оскільки уникнути набагато простіше помилки виключення, що призводять до непередбачуваних наслідків. До того ж, він дозволяє змінювати поведінку об'єкту без необхідності змінювати його код.

Патерн "Пул об'єктів" (Object Pool) [8] — це структурний патерн проектування, який дозволяє створювати пул об'єктів з попередньо визначеною кількістю об'єктів. Клієнт може отримати доступ до об'єкта з пулу, використовувати його та повернути назад, щоб інші клієнти також могли використати його. Таким чином, знижується кількість створюваних об'єктів, що дозволяє зменшити витрати на збір сміття та підвищити продуктивність. Цей патерн зазвичай використовується для керування обмеженими ресурсами, які є обчислювано важкими або мають обмежену кількість доступних екземплярів, наприклад, підключення до баз даних чи мережеві з'єднання.

Для початку розглянемо стандартні методи життєвого циклу класу `MonoBehaviour`, який являє собою базовий у Unity, та використовується для створення компонентів, що керують поведінкою об'єктів у грі. Клас `MonoBehaviour` містить низку стандартних методів життєвого циклу, які викликаються автоматично у різних етапах життєвого циклу об'єкта.

Ось декілька стандартних методів життєвого циклу класу `MonoBehaviour`:

- `Awake()`: Цей метод викликається при створенні екземпляру об'єкта. Він виконується перед будь-якими іншими методами та використовується, щоб налаштувати початковий стан об'єкта;

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

- `Start()`: Цей метод викликається один раз після методу `Awake()`. Він використовується для ініціалізації даних або запуску процесів, які повинні початися один раз після створення об'єкта;
- `Update()`: Цей метод викликається кожен кадр гри. Він використовується для оновлення стану об'єкта та взаємодії з іншими об'єктами або користувачем. Багато логіки гри поміщуються саме в цей метод;
- `FixedUpdate()`: Цей метод викликається з фіксованою частотою, визначеною у налаштуваннях фізичного двигуна. Він використовується для оновлення фізики об'єкта та виконання розрахунків, які потребують стабільного часового кроку;
- `LateUpdate()`: Цей метод викликається після всіх методів `Update()` у всіх об'єктах. Він використовується для забезпечення того, щоб всі об'єкти вже оновили свій стан перед тим, як об'єкт поточного класу проведе остаточні зміни або розрахунки;
- `OnEnable()`: Цей метод викликається, коли компонент стає активним. Він використовується для ініціалізації даних або відновлення стану після включення компонента;
- `OnDisable()`: Цей метод викликається, коли компонент вимикається або видаляється. Він використовується для звільнення ресурсів або виконання додаткових операцій перед вимкненням компонента.

Також замість старої системи вводу в рушії яка являла собою перевірку чи натиснув користувач на кнопку кожен кадр використовується нова система яка дозволяє працювати з подіями які викликаються автоматично, та все що потрібно це підписати метод на потрібну подію для подальшої реалізації.

New Input System підтримує різні типи вводу, включаючи клавіатуру, мишу, геймпади, сенсорні екрани, VR та AR. Крім того, вона має гнучкі налаштування, що дозволяють змінювати мапінги вводу, налаштовувати обробники подій та робити розподіл вводу між різними гравцями. Unity New Input System надає розширені можливості та полегшує роботу з вводом у ваших

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

проектах. Також полегшує роботу з кросплатформною за допомогою налаштування конфігурацій управління.

Розглянемо класи які відповідають за збереження стану світу та гравця.

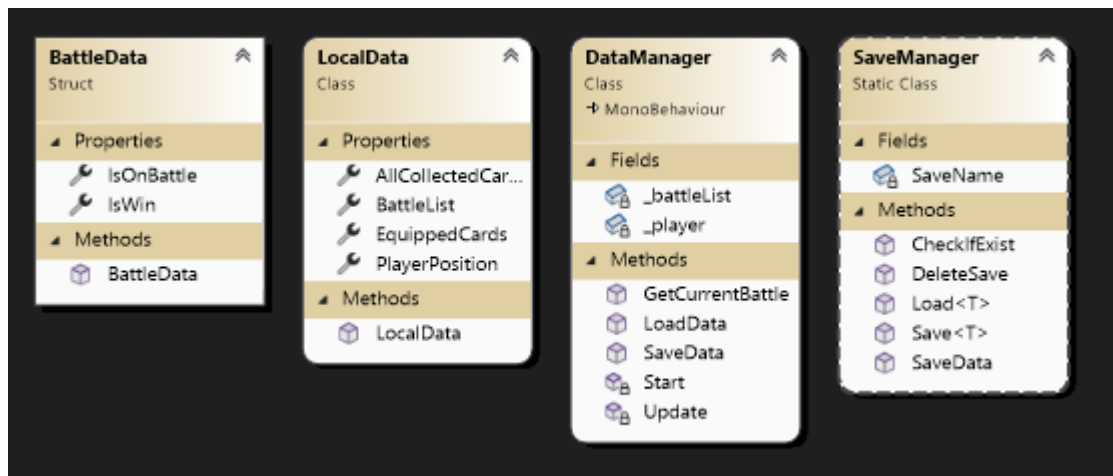


Рисунок 2.3. Діаграма класів завантаження

Клас SaveManager являє собою статичний клас доступ до якого можна отримати з будь-якої частини коду. Він відповідає за зберігання даних у визначеному розробником місці. Збереження відбувається за допомогою серіалізації у JSON файл.

1. SaveName — Статичне поле для позначення імені файлу збереження.
2. CheckIfExists — Відповідає за перевірку наявності у даній ігровій сесії вже готового до використання збереження.
3. DeleteSave — Відповідає за видалення файлу збереження.
4. Load<T> — дозволяє завантажити файл будь-якого формату та повернути його.
5. Save<T> — дозволяє зберегти файл будь-якого формату.
6. SaveData — відповідає за перевірку збереження та правильний формат.

Клас DataManager Відповідає за збереження стану поточної сцени

1. _battleList — Відповідає за список об'єктів битв для контролю вже закінчених боїв.
2. _player — дає доступ до об'єкту гравця для того щоб повернути його на координати останнього збереження.

3. GetCurrentBattle — повертає битву з якою зараз взаємодіє гравець.
4. LoadData — Завантажує стан поточної сцени та синхронізує об'єкти.
5. SaveData — Зберігає стан поточної сцени.

Клас LocalData є класом створеним для серіалізації даних.

1. AllCollectedCards — Список всіх зібраних карт для створення колоди.
2. EquipedCards — Список всіх екіпірованих карт у колоді.
3. BattleList — Список для перевірки стану битв, які завершені або яка наразі активна.
4. PlayerPosition — Зберігає позицію гравця.

Структура BattleData — Відповідає за передачу стану боїв на сцені.

1. IsOnBattle — Показує чи це поточна битва.
2. IsWin — Показує чи виграна ця битва.

2.2. Проектування та реалізація алгоритмів роботи системи

Основним модулем системи є керування гравцем. Даний модуль складається із машини станів, яка контролює всі переміщення гравця, його рух, стрибки та будь-які інші дії. Також за допомогою машини станів була реалізована полегшена навігація поміж анімаціями.

На початку гравець з'являється у світі і має вибір кудись піти. Гравець може виконувати різні дії, такі як рух, стрибки, проходження діалогів та битви. На його шляху будуть зустрічатися тригер зони, які можуть починати бій із супротивником за допомогою колоди карт. Якщо гравець перемагає в битві, він може отримати нагороду, таку як досвід або гроші. При проходженні гри гравець зможе збирати карти, які потім можуть бути використані в битвах. Карту можна зібрати, знайшовши її в грі або купивши в магазині.

Попередньо колоду карт можна скласти відкривши меню інвентарю, в якому видно всі зібрані карти та колоду гравця. Змінювати колоду можна влюбий час за винятком бою. Крім того, у гравця є можливість покращувати свої характеристики, які впливають на його вміння перемагати в битвах.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

Загальний процес гри передбачає проходження різних рівнів, кожен з яких складається з декількох етапів. Гравець повинен успішно пройти всі етапи, щоб перейти на наступний рівень.

Крім того, у гри присутній система збереження, що дозволяє гравцеві зберігати свій прогрес та продовжувати гру з того місця, де він зупинився.

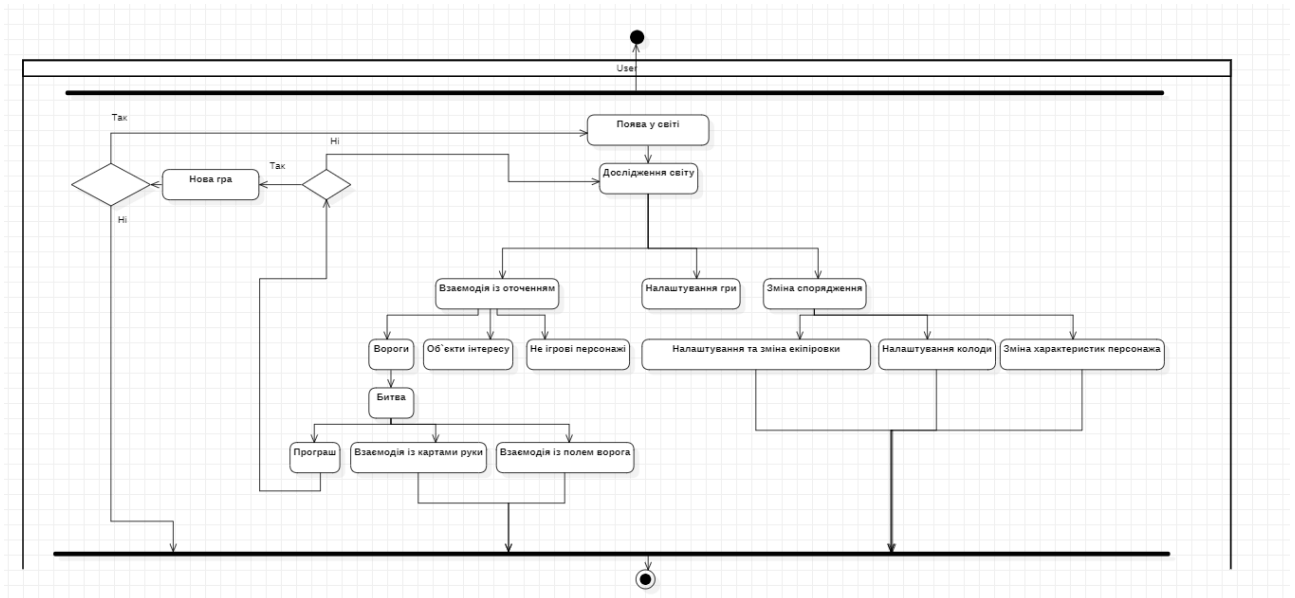


Рисунок 2.4. Діаграма активності та відповідність станів та дій

2.3. Реалізація функціоналу додатку

Зважаючи на те, що основний геймплей відбувається у карткових боях, розберемо реалізацію такого бою.

```
public enum AdditionalEffectsNames
{
    Vampire,
    SelfDamage,
    Poison
}
[Serializable]
public struct AdditionalEffects
{
    [field: SerializeField] public AdditionalEffectsNames AdditionalEffectName { get; private set; }
    [field: SerializeField] public float AdditionalEffectStrength { get; private set; }
}
// [Serializable]
[CreateAssetMenu(fileName = "Card", menuName = "Custom/Cards/Card")]
public class Card : ScriptableObject
{
    [field: SerializeField] public string ID { get; private set; }
    [field: SerializeField] public bool IsUsed { get; private set; } = false;
    [field: SerializeField] public string Name { get; private set; }
    [field: SerializeField] public CardRare CardRare { get; private set; }
    [field: SerializeField, TextArea(5, 10)] public string Description { get; private set; }
}
```

```

[field: SerializeField] public int ManaCost { get; private set; }
[field: SerializeField] public float Damage { get; private set; }
[field: SerializeField] public Sprite Logo { get; private set; }

[field: SerializeField] public List<AdditionalEffects> AdditionalEffectsList
{ get; private set; }

public void ActiveCard(PlayerCards playerCards, EnemyCards enemyCards)
{
    IsUsed = true;
    enemyCards.TakeDamage(Damage);
    if(AdditionalEffectsList.Contains(AdditionalEffectsList.Find(x =>
x.AdditionalEffectName == AdditionalEffectsNames.Vampire)))
        playerCards.TakeDamage(-AdditionalEffectsList.Find(x =>
x.AdditionalEffectName ==
AdditionalEffectsNames.Vampire).AdditionalEffectStrength);
    if(AdditionalEffectsList.Contains(AdditionalEffectsList.Find(x =>
x.AdditionalEffectName == AdditionalEffectsNames.SelfDamage)))
        playerCards.TakeDamage(AdditionalEffectsList.Find(x =>
x.AdditionalEffectName ==
AdditionalEffectsNames.SelfDamage).AdditionalEffectStrength);
}

public void ResetCard()
{
    IsUsed = false;
}
}

```

Для початку клас Card, який відповідає за реалізацію карт. Також ми маємо структуру AdditionalEffects яка використовує перелічування для позначення який саме ефект або ефекти має ця карта.

Метод ActiveCard позначає що ця карта була використана та автоматично накладає ефекти та пошкодження згідно заданим параметрам.

```

public class CardInform : MonoBehaviour
{
    [field: SerializeField] public Card Card { get; private set; }

    // [field: SerializeField] private string ID;
    [field: SerializeField] private TextMeshProUGUI _name;

    [field: SerializeField] private TextMeshProUGUI _description;
    [field: SerializeField] private TextMeshProUGUI _manaCost;
    [field: SerializeField] private float _damage;
    [field: SerializeField] private Image _logo;
    [field: SerializeField] private List<AdditionalEffects> _additionalEffects;

    public void HideCartInfo(Card card)
    {
        Card = card;

        _logo.sprite = null;

        _name.text = "";
        _description.text = "";
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

```

        _manaCost.text = "";
    }

    public void ShowCardInfo(Card card)
    {
        Card = card;

        _logo.sprite = card.Logo;

        _name.text = card.Name;
        _description.text = card.Description;
        _manaCost.text = card.ManaCost.ToString();
    }
}

```

Клас CardInform відповідає за відображення карти на користувацькому інтерфейсі, а саме за її опис, головну картинку, назву, ціну у мані.

```

[System.Serializable]
public class CardHandler : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler
{
    [SerializeField] private RectTransform _canvasRectTransform;
    [SerializeField] private GameObject _tempCardGo;

    private Vector3 _offset;

    public Camera MainCamera;

    public Transform DefaultParent, DefaultTempCardParent;

    public bool IsDraggable = true;
    public bool IsEnoughtMana = true;
    public bool DestroyCard = false;

    public bool DestroyGGCard = false;
    BattleProcess _battleProcess;

    void Awake() {
        MainCamera = Camera.main;
        _tempCardGo = GameObject.Find("TempCardGo");
        _battleProcess = FindObjectOfType<BattleProcess>();
    }

    // private void OnEnable()
    // {
    //     if(GetComponentInParent<Canvas>().GetComponent<RectTransform>())

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

```

//      _canvasRectTransform =
GetComponentInParent<Canvas>().GetComponent<RectTransform>();
// }

    public void SetReferences(RectTransform canvasRectTransform, GameObject
tempCardGo)
    {
        _canvasRectTransform = canvasRectTransform;
        _tempCardGo = tempCardGo;
    }

    public void OnBeginDrag(PointerEventData eventData)
    {
        Cursor.visible = false;
        CardInform card = eventData.pointerDrag.GetComponent<CardInform>();

        _offset = transform.position -
MainCamera.ScreenToWorldPoint(eventData.position);

        if
(RectTransformUtility.ScreenPointToLocalPointInRectangle(_canvasRectTransform,
Input.mousePosition, null, out Vector2 localPoint))
        {
            _offset = transform.position - (Vector3) localPoint ;
        }

        DefaultParent = DefaultTempCardParent = transform.parent;

        //IsDraggable = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.SELF_HAND && GameManager.IsPlayerTurn;
        IsDraggable = _battleProcess.IsPlayerTurn;
        // IsDraggable = true;
        IsEnoughMana = _battleProcess.GetMana() >= card.Card.ManaCost;

        if (!IsDraggable)
        {
            return;
        }

        _tempCardGo.transform.SetParent(DefaultParent);
        _tempCardGo.transform.SetSiblingIndex(transform.GetSiblingIndex());

        transform.SetParent(DefaultParent.parent);

        GetComponent<CanvasGroup>().blocksRaycasts = false;

    }

    public Vector3 GetWorldPositionOnPlane(Vector3 screenPosition, float z) {
        Ray ray = MainCamera.ScreenPointToRay(screenPosition);
        Plane xy = new Plane(Vector3.forward, new Vector3(0, 0, z));
        xy.Raycast(ray, out var distance);
        return ray.GetPoint(distance);
    }

    public void OnDrag(PointerEventData eventData)
    {
        if (!IsDraggable)
        {

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return;
    }

    if
(RectTransformUtility.ScreenPointToLocalPointInRectangle(_canvasRectTransform,
    Input.mousePosition, null, out Vector2 localPoint))
    {
        // Use the localPoint as the mouse position within the canvas
        transform.position = (Vector3)localPoint + _offset;
    }

    CardInform card = eventData.pointerDrag.GetComponent<CardInform>();
    if (_battleProcess.GetMana() >= card.Card.ManaCost)
    {

        if (_tempCardGo.transform.parent != DefaultTempCardParent)
        {
            _tempCardGo.transform.SetParent(DefaultTempCardParent);
        }
    }

    CheckPosition();

}

public void OnEndDrag(PointerEventData eventData)
{
    Cursor.visible = true;
    if (!IsDraggable)
    {
        return;
    }
    transform.SetParent(DefaultParent);
    GetComponent<CanvasGroup>().blocksRaycasts = true;
    CardInform card = eventData.pointerDrag.GetComponent<CardInform>();

    if (DestroyCard = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.EnemyPlace)
    {
        if (DestroyCard && IsDraggable)
        {
            _battleProcess.UpdateMana(-card.Card.ManaCost);
card.Card.ActiveCard(_battleProcess.PlayerCards, _battleProcess.EnemyCards);
            gameObject.SetActive(false);
            gameObject.transform.SetParent(null);
        }
    }

    transform.SetSiblingIndex(_tempCardGo.transform.GetSiblingIndex());
    _tempCardGo.transform.SetParent(GameObject.Find("BattleTableCanvas").transform);
    _tempCardGo.transform.localPosition = new Vector3 (2300, 0);

}

void CheckPosition()
{

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        int newIndex = DefaultTempCardParent.childCount;

        for(int i = 0; i < DefaultTempCardParent.childCount; i++)
        {

            if(transform.position.x <
DefaultTempCardParent.GetChild(i).position.x)
            {
                newIndex = i;

                if(_tempCardGo.transform.GetSiblingIndex() < newIndex)
                {
                    newIndex--;
                }

                break;
            }

        }

        _tempCardGo.transform.SetSiblingIndex(newIndex);
    }
}

```

Клас CardHandler використовує стандартні для юніті інтерфейси що дозволяють перетягувати карти за допомогою миші або пальців у випадку телефону, а саме:

IBeginDragHandler — Відповідає за початок процесу перетягування об'єкту.

IDragHandler — Відповідає за процес перетягування об'єкту.

IEndDragHandler — Відповідає за кінець процесу перетаскування об'єкту.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public enum FieldType
{
    SelfHand,
    EnemyPlace,
    PlayerPlace
}

public class DropPlace : MonoBehaviour, IDropHandler, IPointerEnterHandler,
IPointerExitHandler
{

    public FieldType Type;

    public void OnDrop(PointerEventData eventData)
    {
        var card = eventData.pointerDrag.GetComponent<CardHandler>();

        if (card)
        {
            if (card.IsEnoughMana)
            {
                card.DefaultParent = transform;
            }
        }
    }

    public void OnPointerEnter(PointerEventData eventData)
    {
        if (eventData.pointerDrag == null)
            return;

        var card = eventData.pointerDrag.GetComponent<CardHandler>();

        if (card && card.IsEnoughMana)
        {
            card.DefaultTempCardParent = transform;
        }
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        if (eventData.pointerDrag == null)
            return;

        var card = eventData.pointerDrag.GetComponent<CardHandler>();

        if (card && card.DefaultTempCardParent == transform &&
card.IsEnoughMana)
        {
            card.DefaultTempCardParent = card.DefaultParent;
        }
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас DropPlace використовує стандартні для юніті інтерфейси що дозволяють перевірити входження об'єкту в зону взаємодії та перевірити чи було відпущено об'єкт у цій зоні.

IDropHandler — Перевіряє чи відпустив гравець об'єкт який увійшов до зони взаємодії.

IPointerEnter — Перевіряє чи увійшов об'єкт до зони взаємодії.

IPointerExitHandler — Перевіряє чи вийшов об'єкт із зони взаємодії.

```
public class BattleProcess : MonoBehaviour
{
    public static BattleProcess Instance;

    [field: SerializeField] public CardDataBase CardDataBase { get; private set; }
    [field: SerializeField] public PlayerCards PlayerCards { get; private set; }
    [field: SerializeField] public EnemyCards EnemyCards { get; private set; }

    [SerializeField] private Transform _playerHand;

    [SerializeField] private int _turn;

    [SerializeField] private TextMeshProUGUI _manaPoolText;
    [SerializeField] private int maxMana = 5;
    [SerializeField] private int manaPool = 5;

    [SerializeField] private Button _endTurnButton;

    [SerializeField] private CardInform _cardInformPrefab;

    [SerializeField] private GameObject _tempCardGo;
    [SerializeField] private RectTransform _canvasRectTransform;
    private ObjectPool<CardInform> _cardPool;

    // [SerializeField] private GameObject CardPref;

    public bool IsPlayerTurn
    {
        get
        {
            return _turn % 2 == 0;
        }
    }

    private void Awake() {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
        }
    }
}
```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		41

```

        Destroy(gameObject);
    }
}

private void Start()
{
    _cardPool = new ObjectPool<CardInform>(_cardInformPrefab, 10, true);
    _turn = 0;

    int i = 0;
    while(i++ < 2)
    {
        GiveCard();
    }
    ResetMana();
    UpdateMana(maxMana);
    _endTurnButton.onClick.AddListener(ChangeTurn);
}

public void ChangeTurn()
{
    if (!IsPlayerTurn)
    {
        ChangeToPlayer();
        _turn++;
    }
    else
    {
        EnemyCards.Attack((delegate
        {
            ChangeToPlayer();
            _turn++;
            _turn++;
        }));
    }
}

private void ChangeToPlayer()
{
    int i = 0;
    while (i++ < 1)
    {
        GiveCard();
    }

    UpdateMana(3);
}

private void GiveCard()
{
    if(PlayerInventory.Instance.EquippedCards.All(x => x.IsUsed))
    {
        PlayerInventory.Instance.EquippedCards.ForEach(x => x.ResetCard());
    }

    Card card = PlayerInventory.Instance.EquippedCards[Random.Range(0,
    PlayerInventory.Instance.EquippedCards.Count)];
    while (card.IsUsed)
    {

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        card = PlayerInventory.Instance.EquippedCards[Random.Range(0,
PlayerInventory.Instance.EquippedCards.Count)];
    }

    var obj = _cardPool.GetFreeElement();
    obj.ShowCardInfo(card);
    obj.transform.SetParent(_playerHand);

obj.GetComponent<CardHandler>().SetReferences(_canvasRectTransform,_tempCardGo);
}

public void UpdateMana(int mana)
{
    manaPool += mana;
    _manaPoolText.text = manaPool.ToString();
}

private void ResetMana()
{
    manaPool = 0;
    _manaPoolText.text = manaPool.ToString();
}

public float GetMana()
{
    return manaPool;
}

public void WinBattle()
{
    SceneManager.LoadScene("SampleScene");
    GameProcess.Instance.SetIsWin(true);
    // GameProcess.Instance.GiveReward();
}

public void LoseBattle()
{
    SceneManager.LoadScene(SceneManager.sceneCountInBuildSettings);
}
}

```

BattleProcess — клас що відповідає за правильність дій гри, чий зараз хід, гравця чи ворога, видача карт та відновлення мани і звертається до скрипта управління ворогом для виклику процесу атаки.

Висновки до другого розділу

У даному розділі було розглянуто основні функціональні можливості системи ігрової програми "MunchkinRogueLike", а також описано різні сценарії використання, які задовольняють потреби користувачів. Було представлено діаграму варіантів використання та діаграму класів проекту.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

Також в розділі було описано використання різних патернів проектування, таких як "Одинак", "Машина станів" та "Пул об'єктів". Ці патерни були використані для створення класів, які забезпечують керування ігровою сценою та об'єктами.

Для керування поведінкою об'єктів у грі використовувалися стандартні методи життєвого циклу класу MonoBehaviour, такі як "Awake()", "Start()", "Update()", "FixedUpdate()", "LateUpdate()", "OnEnable()" та "OnDisable()".

Нова система вводу в русії покращила роботу з подіями, що викликаються автоматично, та дозволяє працювати з ними швидше та ефективніше.

В цілому, проектування та реалізація додатку "MunchkinRogueLike" була проведена з використанням сучасних технологій та методів проектування, що дозволило створити якісну та функціональну гру для користувачів.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Порядок встановлення та налаштування системи.

Гра була написана на рушії Unity то ж у подальшому з легкістю запрацює на мобільних пристроях, а дякуючи новій системі вводу/виводу проблем з переносом на будь-які платформи включаючи консолі не буде.

Розроблена система має підтримувати стабільну та комфортну частоту кадрів на будь-яких девайсах у межах не старіше ніж 6-7 років.

3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з грою.

Після завантаження користувач опиняється у головному меню.

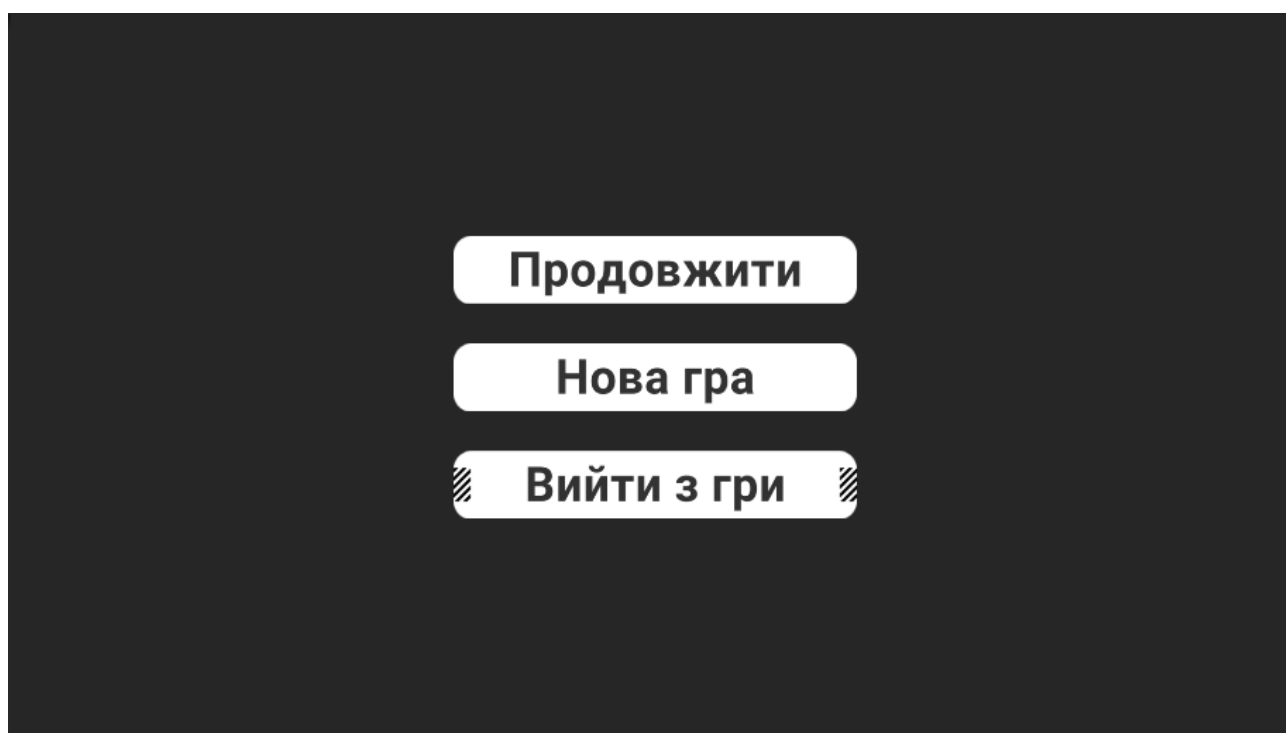


Рисунок 3.1. Головне меню

На екрані головного меню можна побачити 3 кнопки які відповідають:

«Продовжити» — за продовження сесії, якщо збережень немає то кнопка буде не доступна для натискання

«Нова гра» — за початок нової гри

«Вийти з гри» — вихід з гри

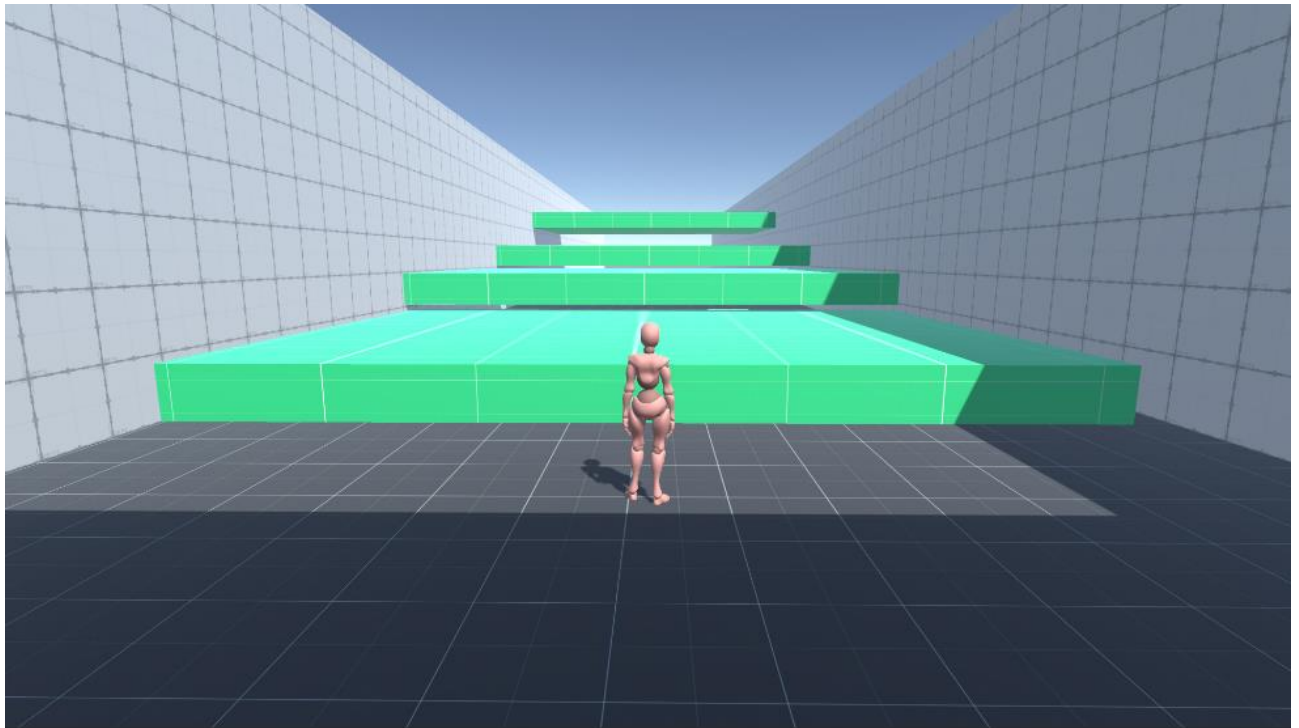


Рисунок 3.2. Початок гри

Для того щоб пересуватися користувач натискає клавіші W S A D, де:

- W — рух вперед;
- S — рух назад;
- A — рух вліво;
- D — рух вправо.

Для бігу користувач може натиснути на клавішу Shift.

За допомогою клавіші Ctrl можна переключитись на ходьбу.

Клавіша I відкриває інвентар де можна змінити колоду.

Клавіша Escape відкриє меню паузи.

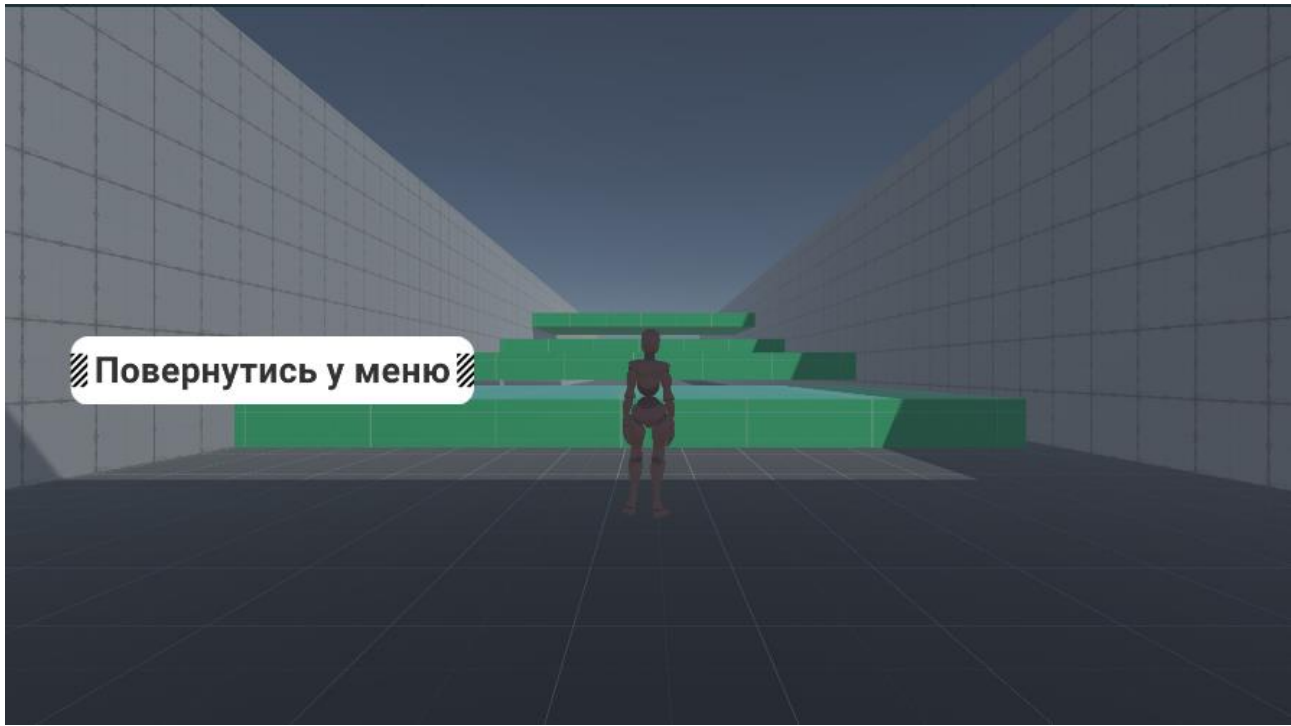


Рисунок 3.3. Меню паузи

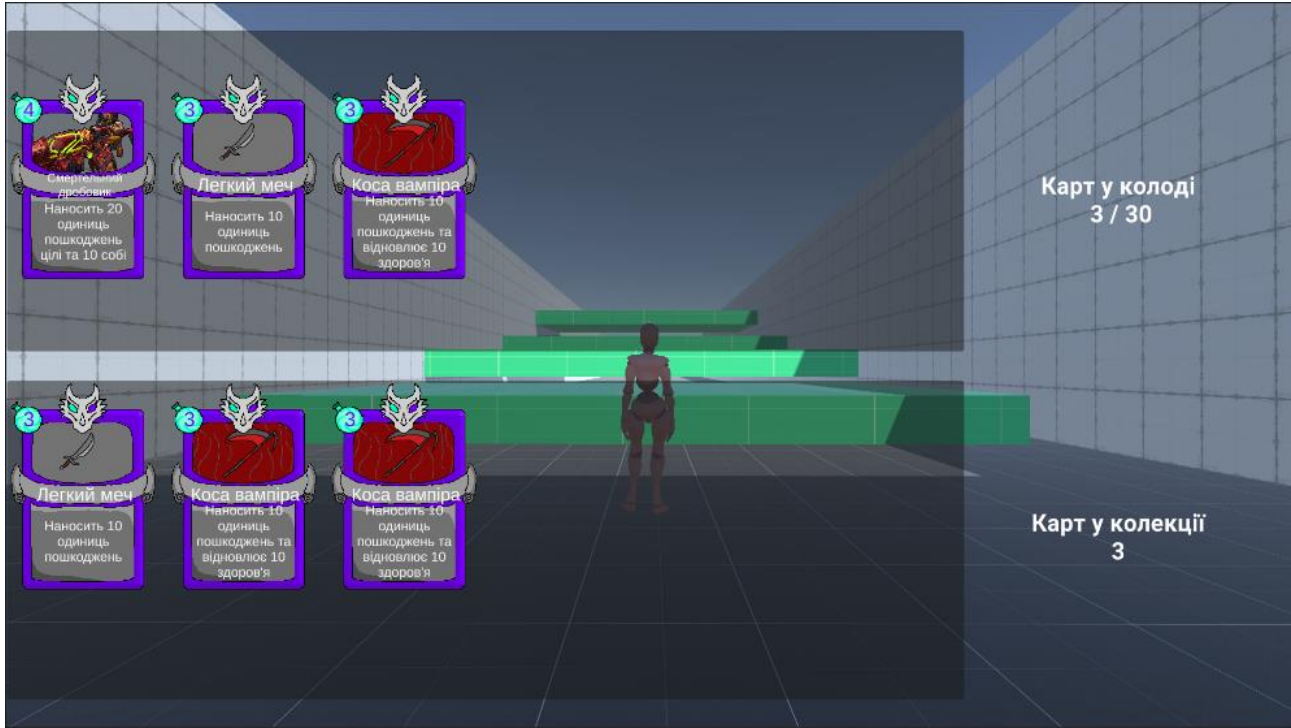


Рисунок 3.4. Інвентар

Щоб пройти платформи на них потрібно застрибнути.

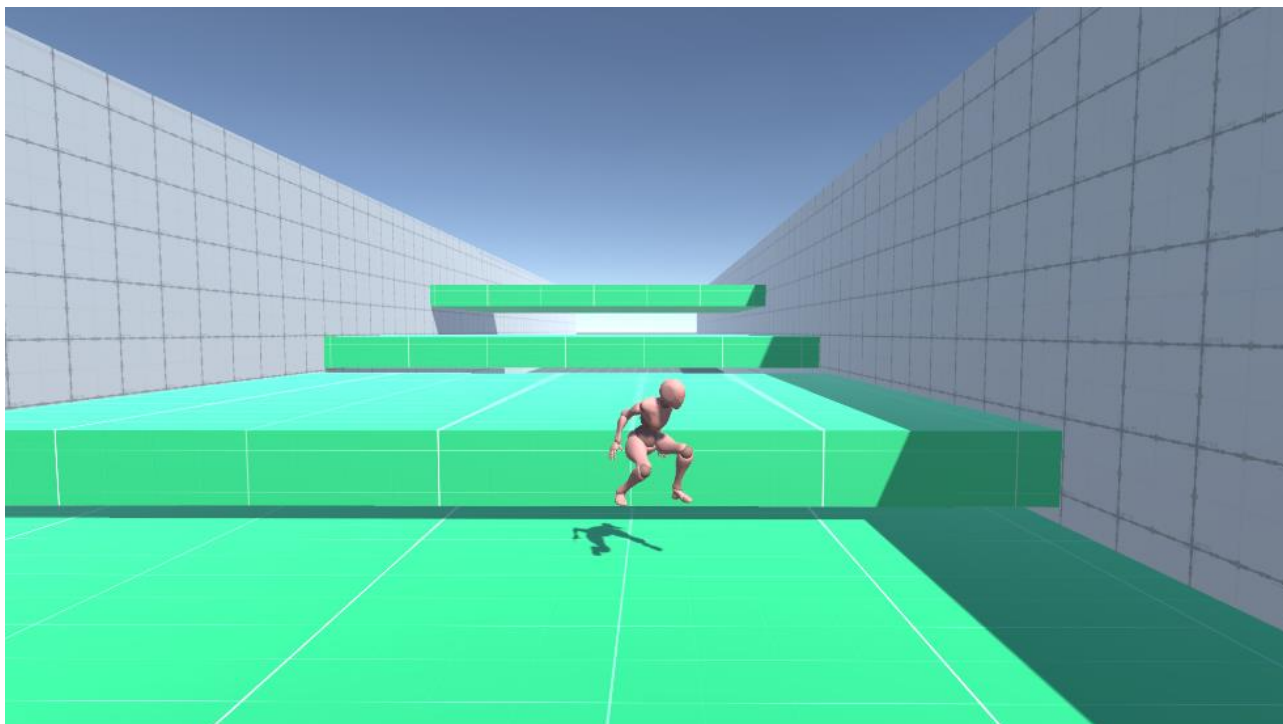


Рисунок 3.5. Стрибки гравця

При падінні в русі з великої висоти персонаж зробить перекат.

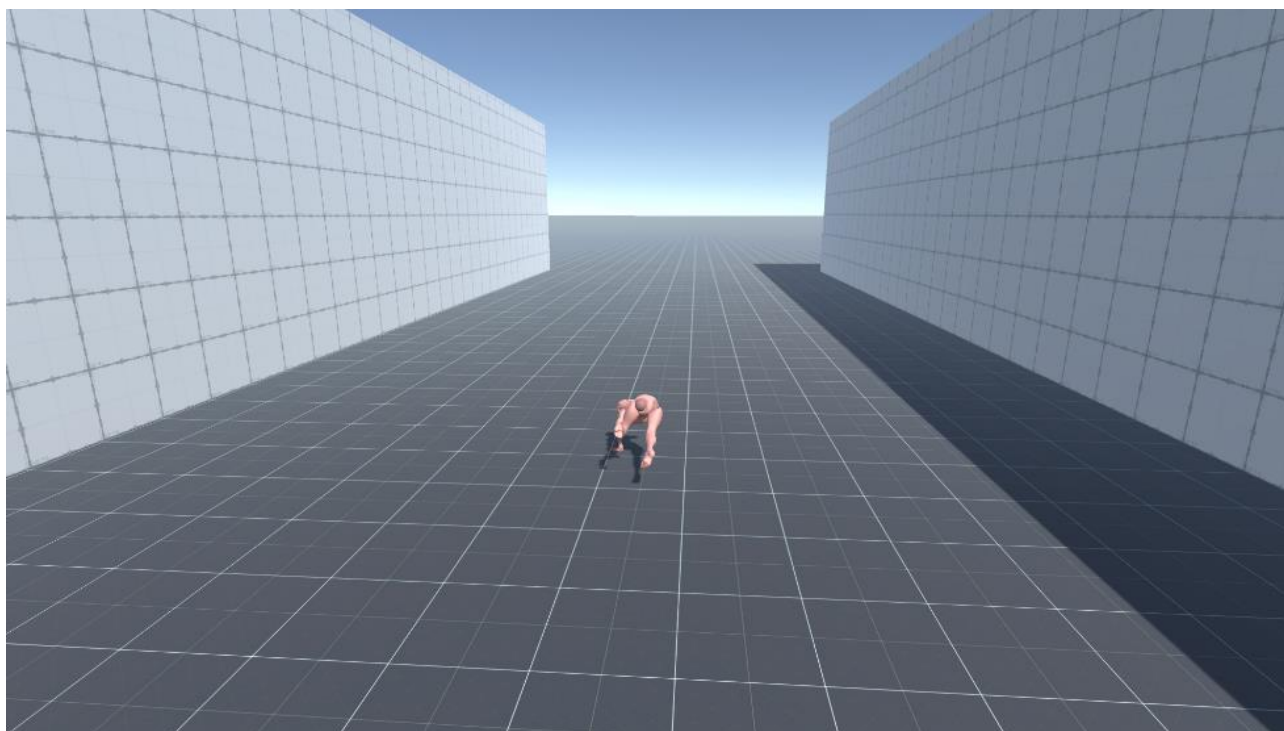


Рисунок 3.6. Перекат гравця

Якщо підійти близько до ворога почнеться бій.

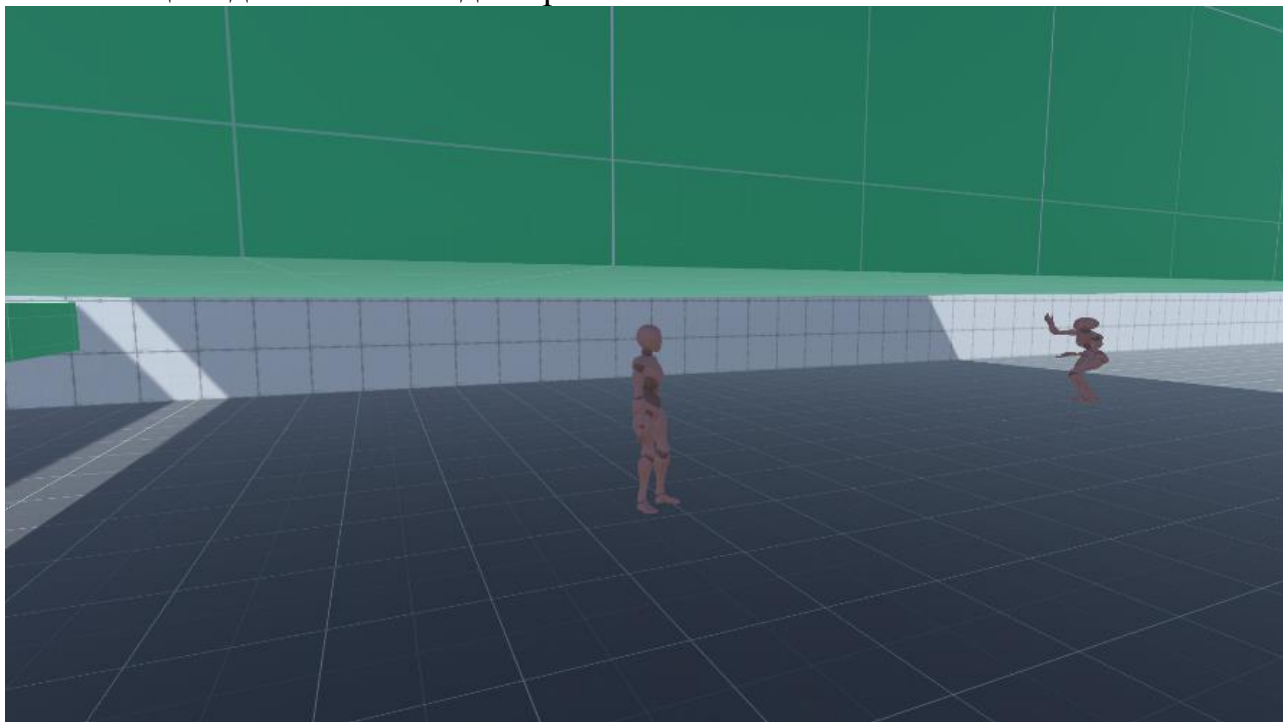


Рисунок 3.7. Ворог

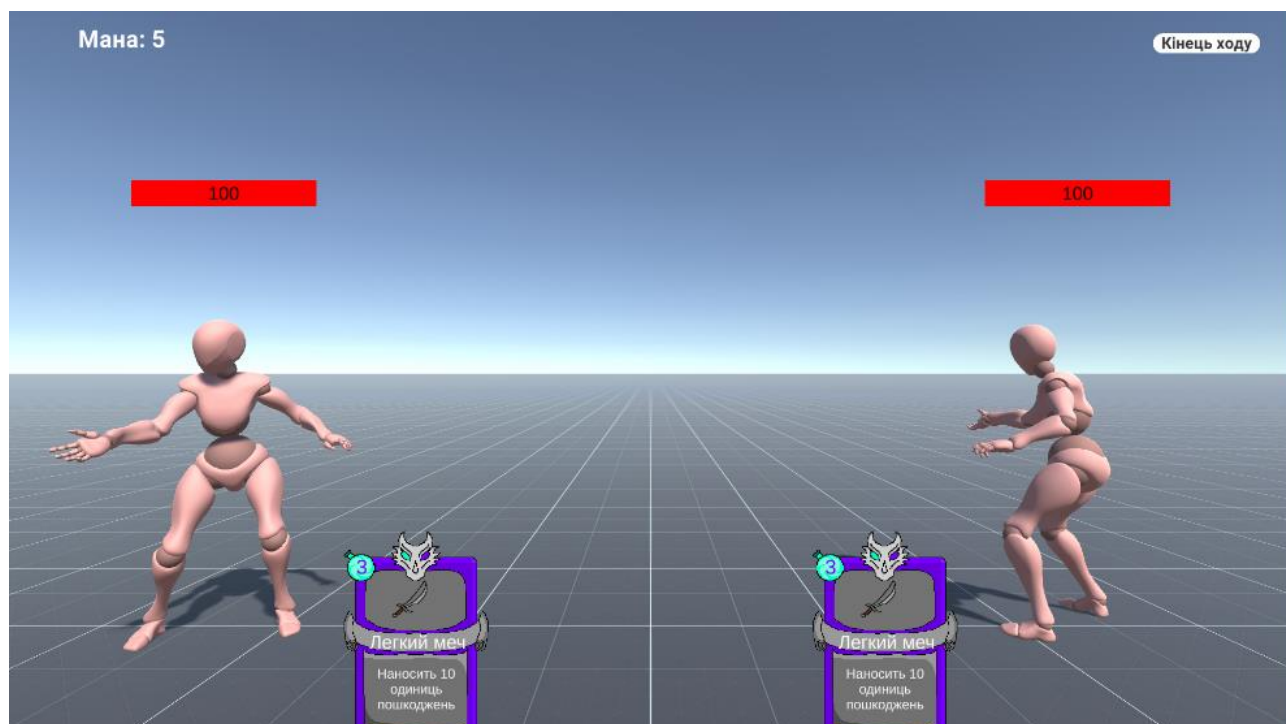


Рисунок 3.8. Сцена бою.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

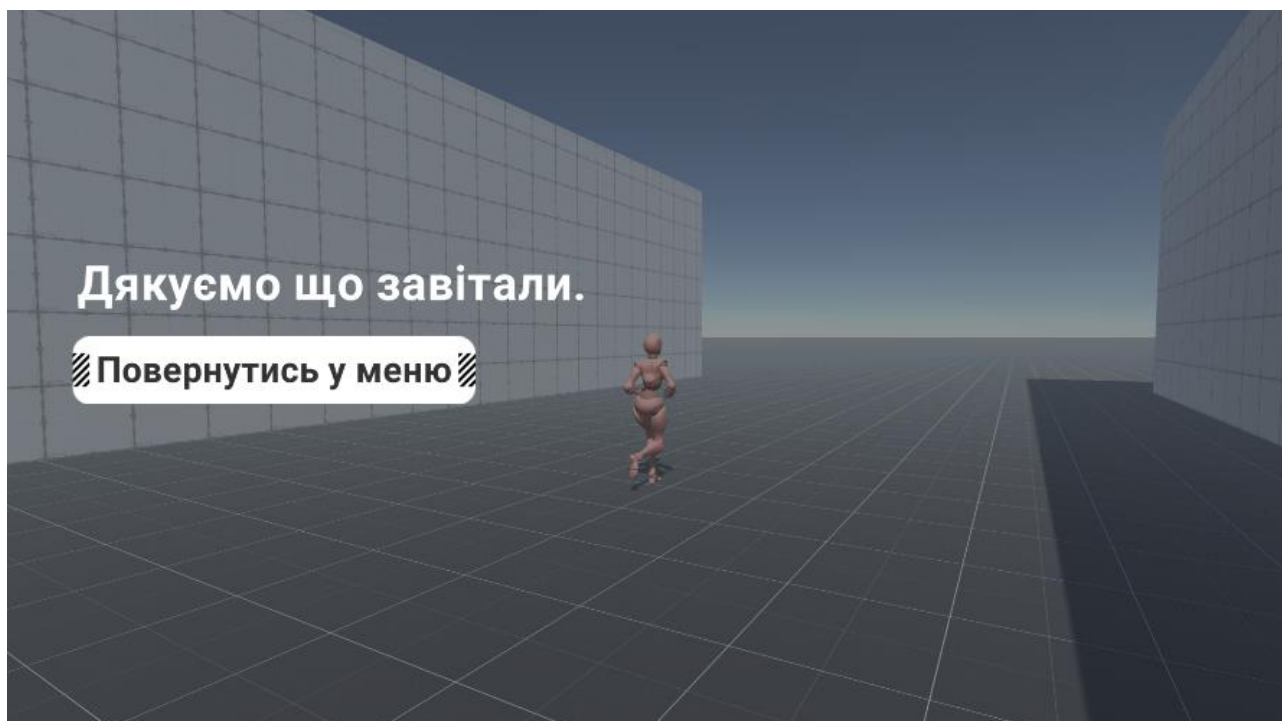


Рисунок 3.9. Кінцевий екран.

3.3. Тестування роботи системи

Проведено технічні дослідження для відповідності продукту стосовно контексту записки. Проведено тестування у середовищі Unity, що дозволило видалити помилки завдяки перевірці гри у реальному часі.

Було перевірено відповідність вимогам, правильна взаємодія для усіх можливих вхідних даних, виконання функцій за прийнятний час, практичність.

В ході тестування було отримано працездатну ігрову програму, яка працює у звичайному режимі [\[14,15\]](#).

Висновки до третього розділу

Гра написана на рушії Unity, що дозволяє легко запускати її на мобільних пристроях, а також на консолях. Розроблена система забезпечує стабільну та комфортну частоту кадрів на пристроях не старше 6-7 років.

Після завантаження, користувач опиняється у головному меню, де може продовжити сесію, розпочати нову гру або вийти з гри. Для переміщення гравець використовує клавіші W, S, A, D, для бігу може натиснути Shift, для ходьби — Ctrl. Інвентар можна відкрити за допомогою клавіші I, а меню паузи — за

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

допомогою Escape. Щоб перескочити платформу, гравець застрибує на неї. При падінні з висоти гравець зробить переكات. Бій починається при підході до ворога.

Було проведено технічні дослідження та тестування у середовищі Unity, що дозволило виявити та виправити помилки. Гра працює у звичайному режимі, відповідає вимогам та забезпечує правильну взаємодію з користувачем.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Розробка гри "MunchkinRogueLike" має на меті створення захоплюючого та емоційного ігрового досвіду для гравця. Головною метою є розважити користувача та розвинути його навички логічного мислення за допомогою створення найефективнішої збірки персонажа. Проект реалізовується у вигляді гри в жанрі Колекційна карткова гра з елементами РПГ та Roguelike.

Бізнес логіка передбачає рекламу на сайтах з власних коштів, платні послуги та продукти та поетапне впровадження проекту.

Вимоги користувачів включають можливість битися з різноманітними монстрами, збирати колоду з обмеженої кількості карт, споряджати свого персонажа, робити покупки різноманітних товарів у магазині, що будуть зустрічатися на карті.

Геймплей гри повинен надавати цікавий ігровий досвід користувачеві за допомогою боївок, повної свободи дій та можливості збереження інформації про ігровий процес гравця.

В системних вимогах передбачено, що гра повинна працювати на комп'ютерах з 8 ГБ пам'яті та процесором з тактовою частотою 3.1 ГГц, операційна система Windows 10. Кросплатформеність передбачає роботу гри на інших платформах, таких як IOS та Android.

Не функціональні вимоги вказують на необхідність зрозумілого та інтуїтивного користувацького інтерфейсу, надійності гри та відсутності інформації, забороненої нормами моралі та законодавством.

Усі ці вимоги та аналіз дозволяють зробити висновок, що розробка гри "MunchkinRogueLike" має перспективи та може стати популярною серед користувачів.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Чистий код: Посібник з майстерності гнучкого програмного забезпечення", Роберт К. Мартін, Prentice Hall (2008) — 441 с.
2. Чистий код та рефакторинг [Електронний ресурс] — Режим доступу до ресурсу: <https://refactoring.guru/uk/refactoring>
3. Andrew Trolelsen, Philip Japikse Pro C# 7: With .NET and .NET Core 2017 — 1372 с.
4. Документація C# [Електронний ресурс] — Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/csharp/>
5. Фрімен Е. Паттерни проектування / Е. Фрімен, Е. Фрімен. — 2011. — 656 с.
6. Машина станів [Електронний ресурс] — Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns/state>
7. Одинак [Електронний ресурс] — Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns/singleton>
8. Пул об'єктів [Електронний ресурс] — Режим доступу до ресурсу: <https://betterprogramming.pub/object-pooling-in-unity-aac9ccbc5dbd>
9. Документація Unity [Електронний ресурс] — Режим доступу до ресурсу: <https://forum.unity.com/>
10. Документація Unity [Електронний ресурс] — Режим доступу до ресурсу: <https://docs.unity3d.com/>
11. "Clean Architecture: A Craftsman`s Guide to Software Structure and Design" — Роберт Мартін, 2017 — 432с.
12. Торн А. Mastering Unity Scripting / Алан Торн. — Бірмінгем: Packt Publishing, 2015. — 380 с.
13. Heather Brown "Visual Studio Code Tips and Tricks" — 150 с.
14. Glenford J. Myers, Corey Sandler, Tom Badgett "The Art of Software Testing" — 256 с.
15. Alan Page, Ken Johnston, and Bj Rollison "How We Test Software at Microsoft" — 405 с.

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас Bootstrap

```

public class Bootstrap : MonoBehaviour
{
    [SerializeField] private GameObject _gameProcessPrefab;
    [SerializeField] private GameObject _playerInventoryPrefab;
    private void Awake()
    {
        // DontDestroyOnLoad(this.gameObject);
        Init();
    }

    private void Init()
    {
        if (GameProcess.Instance == null)
        {
            Instantiate(_gameProcessPrefab);
        }
        if (PlayerInventory.Instance == null)
        {
            Instantiate(_playerInventoryPrefab);
        }
    }
}

```

Клас CameraZoom

```

public class CameraZoom : MonoBehaviour
{
    [SerializeField] [Range(0f, 10f)] private float defaultDistance = 6f;
    [SerializeField] [Range(0f, 10f)] private float minimumDistance = 1f;
    [SerializeField] [Range(0f, 10f)] private float maximumDistance = 7f;

    [SerializeField] [Range(0f, 10f)] private float smoothing = 4f;
    [SerializeField] [Range(0f, 10f)] private float zoomSensitivity = 1f;

    private CinemachineFramingTransposer _framingTransposer;
    private CinemachineInputProvider _inputProvider;

    private float _currentTargetDistance;
    private void Awake()
    {
        _framingTransposer = GetComponent<CinemachineVirtualCamera>()
            .GetCinemachineComponent<CinemachineFramingTransposer>();

        _inputProvider = GetComponent<CinemachineInputProvider>();

        _currentTargetDistance = defaultDistance;
    }

    private void Update()
    {
        Zoom();
    }

    private void Zoom()
    {
        float zoomValue = _inputProvider.GetAxisValue(2) * zoomSensitivity;

        _currentTargetDistance = Mathf.Clamp(_currentTargetDistance + zoomValue,
            minimumDistance, maximumDistance);
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        float currentDistance = _framingTransposer.m_CameraDistance;

        if(currentDistance == _currentTargetDistance) return;

        float lerpedZoomValue = Mathf.Lerp(currentDistance,
        _currentTargetDistance, smoothing * Time.deltaTime);

        _framingTransposer.m_CameraDistance = lerpedZoomValue;
    }
}

```

Клас Player

```

[RequireComponent(typeof(PlayerInput))]
public class Player : MonoBehaviour
{
    [field: Header("References")]
    [field: SerializeField] public PlayerSO Data { get; private set; }

    [field: Header("Collisions")]

    [field: SerializeField] public PlayerCapsuleColliderUtility ColliderUtility
    { get; private set; }

    [field: SerializeField] public PlayerAnimationData AnimationData { get;
    private set; }

    [field: Header("Animations")]
    [field: SerializeField] public PlayerLayerData LayerData { get; private set;
    }

    public Rigidbody Rigidbody { get; private set; }
    public Animator Animator { get; private set; }
    public Transform MainCameraTransform { get; private set; }
    public PlayerInput Input { get; private set; }

    private PlayerMovementStateMachine _movementStateMachine;

    private void Awake()
    {
        Rigidbody = GetComponent<Rigidbody>();
        Input = GetComponent<PlayerInput>();
        Animator = GetComponentInChildren<Animator>();

        if (Camera.main != null) MainCameraTransform = Camera.main.transform;

        AnimationData.Initialize();

        _movementStateMachine = new PlayerMovementStateMachine(this);
    }

    private void OnValidate()
    {
        ColliderUtility.Initialize(gameObject);
        ColliderUtility.CapsuleColliderDimensions();
    }

    private void Start()
    {
        _movementStateMachine.ChangeState(_movementStateMachine.IdlingState);
    }

    private void OnTriggerEnter(Collider collider)

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

```

{
    _movementStateMachine.OnTriggerEnter(collider);
}

private void OnTriggerExit(Collider collider)
{
    _movementStateMachine.OnTriggerExit(collider);
}

private void Update()
{
    _movementStateMachine.HandleInput();
    _movementStateMachine.Update();
}

private void FixedUpdate()
{
    _movementStateMachine.PhysicsUpdate();
}

public void OnMovementStateAnimationEnterEvent()
{
    _movementStateMachine.OnAnimationEnterEvent();
}

public void OnMovementStateAnimationExitEvent()
{
    _movementStateMachine.OnAnimationExitEvent();
}

public void OnMovementStateAnimationTransitionEvent()
{
    _movementStateMachine.OnAnimationTransitionEvent();
}
}

```

Клас PlayerMovementStateMachine

```

public class PlayerMovementStateMachine : StateMachine
{
    public Player Player { get; }
    public PlayerStateReusableData ReusableData { get; }
    public PlayerIdlingState IdlingState { get; }
    public PlayerDashingState DashingState { get; }
    public PlayerRunningState RunningState { get; }
    public PlayerWalkingState WalkingState { get; }
    public PlayerSprintingState SprintingState { get; }
    public PlayerLightStoppingState LightStoppingState { get; }
    public PlayerMediumStoppingState MediumStoppingState { get; }
    public PlayerHardStoppingState HardStoppingState { get; }

    public PlayerLightLandingState LightLandingState { get; }
    public PlayerRollingState RollingState { get; }
    public PlayerHardLandingState HardLandingState { get; }
    public PlayerJumpingState JumpingState { get; }
    public PlayerFallingState FallingState { get; }

    public PlayerMovementStateMachine(Player player)
    {
        Player = player;
        ReusableData = new PlayerStateReusableData();
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        IdlingState = new PlayerIdlingState(this);
        DashingState = new PlayerDashingState(this);

        WalkingState = new PlayerWalkingState(this);
        RunningState = new PlayerRunningState(this);
        SprintingState = new PlayerSprintingState(this);

        LightStoppingState = new PlayerLightStoppingState(this);
        MediumStoppingState = new PlayerMediumStoppingState(this);
        HardStoppingState = new PlayerHardStoppingState(this);

        LightLandingState = new PlayerLightLandingState(this);
        RollingState = new PlayerRollingState(this);
        HardLandingState = new PlayerHardLandingState(this);

        JumpingState = new PlayerJumpingState(this);
        FallingState = new PlayerFallingState(this);
    }
}

```

Клас PlayerMovementState

```

public class PlayerMovementState : IState
{
    protected PlayerMovementStateMachine stateMachine;

    protected PlayerGoundedData movementData;

    protected PlayerAirborneData airborneData;

    public PlayerMovementState(PlayerMovementStateMachine
playerMovementStateMachine)
    {
        stateMachine = playerMovementStateMachine;

        movementData = stateMachine.Player.Data.GoundedData;
        airborneData = stateMachine.Player.Data.AirborneData;

        InitializeData();
    }

    private void InitializeData()
    {
        SetBaseRotationData();
    }

    #region IState Methods
    public virtual void Enter()
    {
        AddInputActionCallbacks();
    }
    public virtual void Exit()
    {
        RemoveInputActionCallbacks();
    }
    public virtual void HandleInput()
    {
        ReadMovementInput();
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public virtual void Update()
{
}

public virtual void PhysicsUpdate()
{
    Move();
}

public virtual void OnAnimationEnterEvent()
{
}

public virtual void OnAnimationExitEvent()
{
}

public virtual void OnAnimationTransitionEvent()
{
}

public virtual void OnTriggerEnter(Collider collider)
{
    if
(stateMachine.Player.LayerData.IsGroundLayer(collider.gameObject.layer))
    {
        OnContactWithGround(collider);

        return;
    }
}

public void OnTriggerExit(Collider collider)
{
    if
(stateMachine.Player.LayerData.IsGroundLayer(collider.gameObject.layer))
    {
        OnContactWithGroundExited(collider);
        return;
    }
}

#endregion

#region Main Methods
private void ReadMovementInput()
{
    stateMachine.ReusableData.MovementInput =
stateMachine.Player.Input.PlayerActions.Movement.ReadValue<Vector2>();
}

private void Move()
{
    if (stateMachine.ReusableData.MovementInput == Vector2.zero ||
stateMachine.ReusableData.MovementSpeedModifier == 0f)
    {
        return;
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Vector3 movementDirection = GetMovementInputDirection();

        float targetRotationYAngle = Rotate(movementDirection);

        Vector3 targetRotationDirection =
GetTargetRotationDirection(targetRotationYAngle);

        float movementSpeed = GetMovementSpeed();

        Vector3 currentPlayerHorizontalVelocity = GetPlayerHorizontalVelocity();

        stateMachine.Player.Rigidbody.AddForce(targetRotationDirection *
movementSpeed - currentPlayerHorizontalVelocity, ForceMode.VelocityChange);
    }

    private float Rotate(Vector3 direction)
    {
        float directionAngle = UpdateTargetRotation(direction);

        RotateTowardsTargetRotation();

        return directionAngle;
    }

    protected float UpdateTargetRotation(Vector3 direction, bool
shouldConsiderCameraRotation = true)
    {
        float directionAngle = GetDirectionAngle(direction);

        if (shouldConsiderCameraRotation) directionAngle =
AddCameraRotationToAngle(directionAngle);

        if (directionAngle != stateMachine.ReusableData.CurrentTargetRotation.y)
UpdateTargetRotationData(directionAngle);

        return directionAngle;
    }

    private float GetDirectionAngle(Vector3 direction)
    {
        float directionAngle = Mathf.Atan2(direction.x, direction.z) *
Mathf.Rad2Deg;
        if (directionAngle < 0f) directionAngle += 360f;
        return directionAngle;
    }

    private float AddCameraRotationToAngle(float angle)
    {
        angle += stateMachine.Player.MainCameraTransform.eulerAngles.y;

        if (angle > 360f) angle -= 360f;
        return angle;
    }

    private void UpdateTargetRotationData(float targetAngle)
    {
        stateMachine.ReusableData.CurrentTargetRotation.y = targetAngle;

        stateMachine.ReusableData.DampedTargetRotationPassedTime.y = 0f;
    }

    #endregion

    #region Reusable Methods

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

protected void StartAnimation(int animationHash)
{
    stateMachine.Player.Animator.SetBool(animationHash, true);
}
protected void StopAnimation(int animationHash)
{
    stateMachine.Player.Animator.SetBool(animationHash, false);
}
protected void SetBaseRotationData()
{
    stateMachine.ReusableData.RotationData = movementData.BaseRotationData;

    stateMachine.ReusableData.TimeToReachTargetRotation =
stateMachine.ReusableData.RotationData.TargetRotationReachTime;
}
protected Vector3 GetMovementInputDirection()
{
    return new Vector3(stateMachine.ReusableData.MovementInput.x, 0f,
stateMachine.ReusableData.MovementInput.y);
}

protected float GetMovementSpeed(bool shouldConsiderSlopes = true)
{
    float movementSpeed = movementData.BaseSpeed *
stateMachine.ReusableData.MovementSpeedModifier;

    if (shouldConsiderSlopes)
    {
        movementSpeed *=
stateMachine.ReusableData.MovementOnSlopesSpeedModifier;
    }
    return movementSpeed;
}

protected Vector3 GetPlayerHorizontalVelocity()
{
    Vector3 playerHorizontalVelocity =
stateMachine.Player.Rigidbody.velocity;

    playerHorizontalVelocity.y = 0f;

    return playerHorizontalVelocity;
}

protected Vector3 GetPlayerVerticalVelocity()
{
    return new Vector3(0f, stateMachine.Player.Rigidbody.velocity.y, 0f);
}
protected void RotateTowardsTargetRotation()
{
    float currentYAngle =
stateMachine.Player.Rigidbody.rotation.eulerAngles.y;

    if(currentYAngle == stateMachine.ReusableData.CurrentTargetRotation.y)
return;

    float smoothYAngle = Mathf.SmoothDampAngle(currentYAngle,
stateMachine.ReusableData.CurrentTargetRotation.y,
    ref stateMachine.ReusableData.DampedTargetRotationCurrentVelocity.y,
    stateMachine.ReusableData.TimeToReachTargetRotation.y -
stateMachine.ReusableData.DampedTargetRotationPassedTime.y);

    stateMachine.ReusableData.DampedTargetRotationPassedTime.y +=

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Time.deltaTime;

    Quaternion targetRotation = Quaternion.Euler(0f, smoothYAngle, 0f);

    stateMachine.Player.Rigidbody.MoveRotation(targetRotation);
}

protected Vector3 GetTargetRotationDirection(float targetAngle)
{
    return Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;
}

protected void ResetVelocity()
{
    stateMachine.Player.Rigidbody.velocity = Vector3.zero;
}

protected void ResetVerticalVelocity()
{
    var playerHorizontalVelocity = GetPlayerHorizontalVelocity();

    stateMachine.Player.Rigidbody.velocity = playerHorizontalVelocity;
}

protected virtual void AddInputActionCallbacks()
{
    stateMachine.Player.Input.PlayerActions.WalkToggle.started +=
OnWalkToggleStarted;
}

protected virtual void RemoveInputActionCallbacks()
{
    stateMachine.Player.Input.PlayerActions.WalkToggle.started -=
OnWalkToggleStarted;
}

protected void DecelerateHorizontally()
{
    var playerHorizontalVelocity = GetPlayerHorizontalVelocity();

    stateMachine.Player.Rigidbody.AddForce(-playerHorizontalVelocity *
stateMachine.ReusableData.MovementDecelerationForce, ForceMode.Acceleration);
}

protected void DecelerateVertically()
{
    var playerVerticalVelocity = GetPlayerVerticalVelocity();

    stateMachine.Player.Rigidbody.AddForce(-playerVerticalVelocity *
stateMachine.ReusableData.MovementDecelerationForce, ForceMode.Acceleration);
}

protected bool IsMovingHorizontally(float minimumMagnitude = .1f)
{
    var playerHorizontalVelocity = GetPlayerHorizontalVelocity();

    var playerHorizontalMovement = new Vector2(playerHorizontalVelocity.x,
playerHorizontalVelocity.z);

    return playerHorizontalMovement.magnitude > minimumMagnitude;
}

protected bool IsMovingUp(float minimumVelocity = .1f)

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    return GetPlayerVerticalVelocity().y > minimumVelocity;
}
protected bool IsMovingDown(float minimumVelocity = .1f)
{
    return GetPlayerVerticalVelocity().y < -minimumVelocity;
}
protected virtual void OnContactWithGround(Collider collider)
{
}

protected virtual void OnContactWithGroundExited(Collider collider)
{
}

#endregion

#region Input Methods
protected virtual void OnWalkToggleStarted(InputAction.CallbackContext context)
{
    stateMachine.ReusableData.ShouldWalk =
!stateMachine.ReusableData.ShouldWalk;
}
#endregion
}

```

Клас PlayerGroundedState

```

public class PlayerGroundedState : PlayerMovementState
{
    private SlopeData _slopeData;
    public PlayerGroundedState(PlayerMovementStateMachine
playerMovementStateMachine) : base(
    playerMovementStateMachine)
    {
        _slopeData = stateMachine.Player.ColliderUtility.SlopeData;
    }

    #region IStateMethods

    public override void Enter()
    {
        base.Enter();

        StartAnimation(stateMachine.Player.AnimationData.GroundedParameterHash);

        UpdateShouldSprintState();
    }

    public override void Exit()
    {
        base.Exit();

        StopAnimation(stateMachine.Player.AnimationData.GroundedParameterHash);
    }

    public override void PhysicsUpdate()
    {

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        base.PhysicsUpdate();

        Float();

    }
    #endregion

    #region MainMethods

    private void UpdateShouldSprintState()
    {
        if(!stateMachine.ReusableData.ShouldSprint) return;

        if(stateMachine.ReusableData.MovementInput != Vector2.zero) return;

        stateMachine.ReusableData.ShouldSprint = false;
    }
    private void Float()
    {
        Vector3 capsuleColliderCentreInWorldSpace =
stateMachine.Player.ColliderUtility.CapsuleColliderData.Collider.bounds.center;
        Ray downwardsFromCapsuleCentre = new
Ray(capsuleColliderCentreInWorldSpace, Vector3.down);

        if (Physics.Raycast(downwardsFromCapsuleCentre, out RaycastHit hit,
        _slopeData.FloatRayDistance,
            stateMachine.Player.LayerData.GroundLayer,
QueryTriggerInteraction.Ignore))
        {
            float groundAngle = Vector3.Angle(hit.normal, -
downwardsFromCapsuleCentre.direction);

            float slopeSpeedModifier =
SetSlopeSpeedModifierOnAngle(groundAngle);
            if(slopeSpeedModifier == 0) return;

            float distanceToFloatingPoint =
stateMachine.Player.ColliderUtility.CapsuleColliderData.ColliderCentreInLocalSpa
ce.y
                * stateMachine.Player.transform.localScale.y - hit.distance;

            if(distanceToFloatingPoint == 0f) return;

            float amountToLift = distanceToFloatingPoint *
        _slopeData.StepReachForce - GetPlayerVerticalVelocity().y;

            Vector3 liftForce = new Vector3(0f, amountToLift, 0f);

            stateMachine.Player.Rigidbody.AddForce(liftForce,
ForceMode.VelocityChange);
        }
    }

    private float SetSlopeSpeedModifierOnAngle(float angle)
    {
        float slopeSpeedModifier =
movementData.SlopeSpeedAngles.Evaluate(angle);

        stateMachine.ReusableData.MovementOnSlopesSpeedModifier =
slopeSpeedModifier;

        return slopeSpeedModifier;
    }

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 – ІІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    private bool IsThereGroundUnderneath()
    {
        var groundCheckCollider =
stateMachine.Player.ColliderUtility.TriggerColliderData.GroundCheckCollider;

        var groundColliderCenterInWorldSpace =
groundCheckCollider.bounds.center;

        Collider[] overlappedGroundColliders =
Physics.OverlapBox(groundColliderCenterInWorldSpace,

stateMachine.Player.ColliderUtility.TriggerColliderData.GroundCheckColliderExten
ts,
            groundCheckCollider.transform.rotation,
            stateMachine.Player.LayerData.GroundLayer,
            QueryTriggerInteraction.Ignore);

        return overlappedGroundColliders.Length > 0;
    }

#endregion

#region Reusable Methods

protected override void AddInputActionCallbacks()
{
    base.AddInputActionCallbacks();
    stateMachine.Player.Input.PlayerActions.Movement.canceled +=
OnMovementCanceled;

    stateMachine.Player.Input.PlayerActions.Dash.started += OnDashStarted;
    stateMachine.Player.Input.PlayerActions.Jump.started += OnJumpStarted;
}

protected override void RemoveInputActionCallbacks()
{
    base.RemoveInputActionCallbacks();
    stateMachine.Player.Input.PlayerActions.Movement.canceled -=
OnMovementCanceled;

    stateMachine.Player.Input.PlayerActions.Dash.started -= OnDashStarted;
    stateMachine.Player.Input.PlayerActions.Jump.started -= OnJumpStarted;
}

protected virtual void OnMove()
{
    if (stateMachine.ReusableData.ShouldSprint)
    {
        stateMachine.ChangeState(stateMachine.SprintingState);
        return;
    }

    if (stateMachine.ReusableData.ShouldWalk)
    {
        stateMachine.ChangeState(stateMachine.WalkingState);
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return;
    }

    stateMachine.ChangeState(stateMachine.RunningState);
}

protected override void OnContactWithGroundExited(Collider collider)
{
    base.OnContactWithGroundExited(collider);

    if (IsThereGroundUnderneath())
    {
        return;
    }

    var capsuleColliderCenterInWorldSpace =
stateMachine.Player.ColliderUtility.CapsuleColliderData.Collider.bounds.center;

    var downwardsRayFromCapsuleBottom = new
Ray(capsuleColliderCenterInWorldSpace
-
stateMachine.Player.ColliderUtility.CapsuleColliderData.ColliderVerticalExtents,
Vector3.down);

    if (!Physics.Raycast(downwardsRayFromCapsuleBottom, out _,
movementData.GroundToFallRayDistance,
stateMachine.Player.LayerData.GroundLayer,
QueryTriggerInteraction.Ignore))
    {
        OnFall();
    }
}

protected virtual void OnFall()
{
    stateMachine.ChangeState(stateMachine.FallingState);
}

#endregion

#region Input Methods
protected virtual void OnMovementCanceled(InputAction.CallbackContext
context)
{
    stateMachine.ChangeState(stateMachine.IdlingState);
}

protected virtual void OnDashStarted(InputAction.CallbackContext context)
{
    stateMachine.ChangeState(stateMachine.DashingState);
}

protected virtual void OnJumpStarted(InputAction.CallbackContext context)
{
    stateMachine.ChangeState(stateMachine.JumpingState);
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#endregion
}

```

Клас PlayerDashingState

```

public class PlayerDashingState : PlayerGroundedState
{
    private PlayerDashData _dashData;
    private float _startTime;
    private int _consecutiveDashesUsed;

    private bool _shouldKeepRotating;
    public PlayerDashingState (PlayerMovementStateMachine
playerMovementStateMachine) : base(playerMovementStateMachine)
    {
        _dashData = movementData.DashData;
    }

    #region IState Methods
    public override void Enter()
    {
        stateMachine.ReusableData.MovementSpeedModifier =
movementData.DashData.SpeedModifier;

        base.Enter();

        StartAnimation(stateMachine.Player.AnimationData.DashParameterHash);

        stateMachine.ReusableData.CurrentJumpForce =
airborneData.JumpData.StrongForce;

        stateMachine.ReusableData.RotationData = _dashData.RotationData;

        Dash();

        _shouldKeepRotating = stateMachine.ReusableData.MovementInput !=
Vector2.zero;

        UpdateConsecutiveDashes();

        _startTime = Time.time;
    }

    public override void Exit()
    {
        base.Exit();
        StopAnimation(stateMachine.Player.AnimationData.DashParameterHash);
        SetBaseRotationData();
    }

    public override void PhysicsUpdate()
    {
        base.PhysicsUpdate();

        if(!_shouldKeepRotating) return;

        RotateTowardsTargetRotation();
    }

    public override void OnAnimationTransitionEvent()
    {
        if (stateMachine.ReusableData.MovementInput == Vector2.zero)
        {

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        stateMachine.ChangeState(stateMachine.HardStoppingState);
        return;
    }
    stateMachine.ChangeState(stateMachine.SprintingState);
}

#endregion

#region Main Methods

private void Dash()
{
    Vector3 dashDirection = stateMachine.Player.transform.forward;

    dashDirection.y = 0f;

    UpdateTargetRotation(dashDirection, false);

    if(stateMachine.ReusableData.MovementInput != Vector2.zero)
    {
        UpdateTargetRotation(GetMovementInputDirection());

        dashDirection =
GetTargetRotationDirection(stateMachine.ReusableData.CurrentTargetRotation.y);
    }

    stateMachine.Player.Rigidbody.velocity = dashDirection *
GetMovementSpeed(false);
}

private void UpdateConsecutiveDashes()
{
    if (!IsConsecutive()) _consecutiveDashesUsed = 0;

    ++_consecutiveDashesUsed;

    if (_consecutiveDashesUsed == _dashData.ConsecutiveDashesLimitAmount)
    {
        _consecutiveDashesUsed = 0;
    }

    stateMachine.Player.Input.DisableActionFor(stateMachine.Player.Input.PlayerActions.Dash, _dashData.DashLimitReachedCooldown);
}

private bool IsConsecutive()
{
    return Time.time < _startTime + _dashData.TimeToBeConsideredConsecutive;
}

#endregion

#region Reusable Methods

protected override void AddInputActionCallbacks()
{
    base.AddInputActionCallbacks();

    stateMachine.Player.Input.PlayerActions.Movement.performed +=
OnMovementPerformed;
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    protected override void RemoveInputActionCallbacks()
    {
        base.RemoveInputActionCallbacks();
        stateMachine.Player.Input.PlayerActions.Movement.performed -=
OnMovementPerformed;
    }

    #endregion
    #region Input Methos
    private void OnMovementPerformed(InputAction.CallbackContext context)
    {
    }
    protected override void OnMovementCanceled(InputAction.CallbackContext
context)
    {
        _shouldKeepRotating = true;
    }

    protected override void OnDashStarted(InputAction.CallbackContext context)
    {
    }
    }

    #endregion
}

```

Клас CapsuleColliderUtility

```

public class CapsuleColliderUtility
{
    public CapsuleColliderData CapsuleColliderData { get; private set; }
    [field: SerializeField] public DefaultColliderData DefaultColliderData {
get; private set; }
    [field: SerializeField] public SlopeData SlopeData { get; private set; }

    public void Initialize(GameObject gameObject)
    {
        if(CapsuleColliderData != null) return;
        CapsuleColliderData = new CapsuleColliderData();
        CapsuleColliderData.Initialize(gameObject);

        OnInitialize();
    }

    protected virtual void OnInitialize()
    {
    }

    public void CapsuleColliderDimensions()
    {
        SetCapsuleColliderRadius(DefaultColliderData.Radius);

        SetCapsuleColliderHeight(DefaultColliderData.Height * (1f -
SlopeData.StepHeightPercentage));

        RecalculateCapsuleColliderCentre();

        float halfColliderHeight = CapsuleColliderData.Collider.height / 2f;

        if (halfColliderHeight < CapsuleColliderData.Collider.radius)

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        SetCapsuleColliderRadius(halfColliderHeight);
    }

    CapsuleColliderData.UpdateColliderData();
}
private void SetCapsuleColliderRadius(float radius)
{
    CapsuleColliderData.Collider.radius = radius;
}

private void SetCapsuleColliderHeight(float height)
{
    CapsuleColliderData.Collider.height = height;
}
public void RecalculateCapsuleColliderCentre()
{
    float colliderHeightDifference = DefaultColliderData.Height -
CapsuleColliderData.Collider.height;

    Vector3 newColliderCentre =
        new Vector3(0f, DefaultColliderData.CentreY +
(colliderHeightDifference / 2f), 0f);

    CapsuleColliderData.Collider.center = newColliderCentre;
}
}

```

Клас ObjectPool

```

public class ObjectPool<T> where T : MonoBehaviour
{
    private readonly T _prefab;
    private readonly bool _autoExpand;
    private readonly Transform _container;
    private List<T> _pool;

    public List<T> Pool {get => _pool;}

    public ObjectPool(T prefab, int count, bool autoExpand)
    {
        this._prefab = prefab;
        var container = new GameObject();
        _container = container.transform;
        container.name = prefab.name;
        this._autoExpand = autoExpand;
        this.CreatePool(count);
    }

    private void CreatePool(int count)
    {
        this._pool = new List<T>();
        for (var i = 0; i < count; i++)
        {
            this.CreateObject();
        }
    }

    private T CreateObject(bool isActive = false)
    {
        var createdObject = Object.Instantiate(this._prefab, this._container);
        createdObject.gameObject.SetActive(isActive);
        this._pool.Add(createdObject);
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return createdObject;
    }

    public T GetFreeElement()
    {
        foreach (var obj in _pool.Where(obj =>
            !obj.gameObject.activeInHierarchy))
        {
            obj.gameObject.SetActive(true);
            return obj;
        }

        return this._autoExpand ? this.CreateObject(true) : null;

        throw new Exception("No free elements");
    }

    public bool IsHaveFreeElements()
    {
        foreach (var obj in _pool.Where(obj =>
            !obj.gameObject.activeInHierarchy))
        {
            return true;
        }

        return false;
    }

    public void HideAllElements()
    {
        foreach (var obj in _pool.Where(obj =>
obj.gameObject.activeInHierarchy))
        {
            obj.gameObject.SetActive(false);
        }
    }

    public List<T> GetAllElements()
    {
        return _pool;
    }

    public void ResetActiveElements()
    {
        foreach (var obj in _pool.Where(obj =>
obj.gameObject.activeInHierarchy))
        {
            obj.gameObject.SetActive(false);
            obj.gameObject.SetActive(true);
        }
    }

    public void DestroyPool()
    {
        GameObject.Destroy(_container.gameObject);
    }

    public void SetAllElementsToOriginalParent()
    {
        foreach (var obj in _pool)
        {

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(obj.transform.parent != this._container)
            obj.transform.SetParent(this._container);
    }
}

```

Клас SaveManager

```

public static class SaveManager
{
    private static readonly string SaveName = "saveData.json";

    public static void SaveData(LocalData localData)
    {
        DeleteSave();
        Save(localData);
    }

    public static void Save<T>(T data)
    {
        string filePath = Path.Combine(Application.persistentDataPath,
SaveName);
        string jsonData = JsonUtility.ToJson(data, true);
        File.WriteAllText(filePath, jsonData);
    }

    public static T Load<T>()
    {
        string filePath = Path.Combine(Application.persistentDataPath,
SaveName);
        string jsonData = File.ReadAllText(filePath);
        var data = JsonUtility.FromJson<T>(jsonData);
        return data;
    }

    public static void DeleteSave()
    {
        string filePath = Path.Combine(Application.persistentDataPath,
SaveName);

        if (File.Exists(filePath))
        {
            File.Delete(filePath);
        }
    }

    public static bool CheckIfExist()
    {
        string filePath = Path.Combine(Application.persistentDataPath,
SaveName);

        if (File.Exists(filePath))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

```

Клас DataManager

```

public class DataManager : MonoBehaviour
{
    [SerializeField] private GameObject _player;
    [SerializeField] private List<GoToBattle> _battleList;

    private void Start()
    {
        if (SaveManager.CheckIfExist())
        {
            LoadData();
        }
    }

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.F5))
        {
            SaveData();
        }
        else if (Input.GetKeyDown(KeyCode.F9))
        {
            LoadData();
        }
    }

    public GoToBattle GetCurrentBattle()
    {
        return _battleList.Find(x => x.GetIsOnBattle() == true);
    }

    public void SaveData()
    {
        var battleDataList = new List<BattleData>();

        foreach (var battle in _battleList)
        {
            battleDataList.Add(new BattleData(battle.GetIsWin(),
battle.GetIsOnBattle()));
        }

        var data = new LocalData(_player.transform.position, battleDataList,
PlayerInventory.Instance.EquippedCards,
        PlayerInventory.Instance.AllCollectedCards);
        SaveManager.SaveData(data);
    }

    public void LoadData()
    {
        if(!SaveManager.CheckIfExist()) return;
        var data = SaveManager.Load<LocalData>();
        for (int i = 0; i < _battleList.Count; i++)
        {
            _battleList[i].SetIsWin(data.BattleList[i].IsWin);
            _battleList[i].SetIsOnBattle(data.BattleList[i].IsOnBattle);
        }

        PlayerInventory.Instance.SetInventory(data.EquippedCards, data.AllCollectedCards)
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

;

    _player.transform.position = data.PlayerPosition;
}

}

[Serializable]
public class LocalData
{
    [field: SerializeField] public Vector3 PlayerPosition { get; private set; }
    [field: SerializeField] public List<BattleData> BattleList { get; private set; }
    [field: SerializeField] public List<Card> EquippedCards { get; private set; }
}

    [field: SerializeField] public List<Card> AllCollectedCards { get; private set; }

    public LocalData(Vector3 playerPosition, List<BattleData> battleList,
List<Card> equippedCards, List<Card> allCollectedCards)
    {
        PlayerPosition = playerPosition;
        BattleList = battleList;
        EquippedCards = equippedCards;
        AllCollectedCards = allCollectedCards;
    }

}

[Serializable]
public struct BattleData
{
    [field: SerializeField] public bool IsWin { get; private set; }

    [field: SerializeField] public bool IsOnBattle { get; private set; }

    public BattleData(bool isWin, bool isOnBattle)
    {
        IsWin = isWin;
        IsOnBattle = isOnBattle;
    }
}

```

Клас Card

```

public enum AdditionalEffectsNames
{
    Vampire,
    SelfDamage,
    Poison
}

[Serializable]
public struct AdditionalEffects
{
    [field: SerializeField] public AdditionalEffectsNames AdditionalEffectName { get; private set; }
    [field: SerializeField] public float AdditionalEffectStrength { get; private set; }
}

// [Serializable]

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```
[CreateAssetMenu(fileName = "Card", menuName = "Custom/Cards/Card")]
public class Card : ScriptableObject
{
    [field: SerializeField] public string ID { get; private set; }
    [field: SerializeField] public bool IsUsed { get; private set; } = false;
    [field: SerializeField] public string Name { get; private set; }
    [field: SerializeField] public CardRare CardRare { get; private set; }
    [field: SerializeField, TextArea(5, 10)] public string Description { get;
private set; }
    [field: SerializeField] public int ManaCost { get; private set; }
    [field: SerializeField] public float Damage { get; private set; }
    [field: SerializeField] public Sprite Logo { get; private set; }

    [field: SerializeField] public List<AdditionalEffects> AdditionalEffectsList
{ get; private set; }

    public void ActiveCard(PlayerCards playerCards, EnemyCards enemyCards)
    {
        IsUsed = true;
        enemyCards.TakeDamage(Damage);
        if(AdditionalEffectsList.Contains(AdditionalEffectsList.Find(x =>
x.AdditionalEffectName == AdditionalEffectsNames.Vampire)))
            playerCards.TakeDamage(-AdditionalEffectsList.Find(x =>
x.AdditionalEffectName ==
AdditionalEffectsNames.Vampire).AdditionalEffectStrength);
        if(AdditionalEffectsList.Contains(AdditionalEffectsList.Find(x =>
x.AdditionalEffectName == AdditionalEffectsNames.SelfDamage)))
            playerCards.TakeDamage(AdditionalEffectsList.Find(x =>
x.AdditionalEffectName ==
AdditionalEffectsNames.SelfDamage).AdditionalEffectStrength);
    }

    public void ResetCard()
    {
        IsUsed = false;
    }
}
```

Клас Card

```
[System.Serializable]
public class CardHandler : MonoBehaviour, IBeginDragHandler, IDragHandler,
IEndDragHandler
{
    [SerializeField] private RectTransform _canvasRectTransform;
    [SerializeField] private GameObject _tempCardGo;

    private Vector3 _offset;

    public Camera MainCamera;

    public Transform DefaultParent, DefaultTempCardParent;

    public bool IsDraggable = true;
    public bool IsEnoughMana = true;
    public bool DestroyCard = false;

    public bool DestroyGGCard = false;
```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

BattleProcess _battleProcess;

void Awake() {
    MainCamera = Camera.main;
    _tempCardGo = GameObject.Find("TempCardGo");
    _battleProcess = FindObjectOfType<BattleProcess>();
}

// private void OnEnable()
// {
//     if(GetComponentInParent<Canvas>().GetComponent<RectTransform>())
//         _canvasRectTransform =
GetComponentInParent<Canvas>().GetComponent<RectTransform>();
// }

public void SetReferences(RectTransform canvasRectTransform, GameObject
tempCardGo)
{
    _canvasRectTransform = canvasRectTransform;
    _tempCardGo = tempCardGo;
}

public void OnBeginDrag(PointerEventData eventData)
{
    Cursor.visible = false;
    CardInform card = eventData.pointerDrag.GetComponent<CardInform>();

    _offset = transform.position -
MainCamera.ScreenToWorldPoint(eventData.position);

    if
(RectTransformUtility.ScreenPointToLocalPointInRectangle(_canvasRectTransform,
    Input.mousePosition, null, out Vector2 localPoint))
    {
        _offset = transform.position - (Vector3) localPoint;
    }

    DefaultParent = DefaultTempCardParent = transform.parent;

    //IsDraggable = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.SELF_HAND && GameManager.IsPlayerTurn;
    IsDraggable = _battleProcess.IsPlayerTurn;
    // IsDraggable = true;
    IsEnoughMana = _battleProcess.GetMana() >= card.Card.ManaCost;

    if (!IsDraggable)
    {
        return;
    }

    _tempCardGo.transform.SetParent(DefaultParent);
    _tempCardGo.transform.SetSiblingIndex(transform.GetSiblingIndex());

    transform.SetParent(DefaultParent.parent);
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

GetComponent<CanvasGroup>().blocksRaycasts = false;

}

public Vector3 GetWorldPositionOnPlane(Vector3 screenPosition, float z) {
    Ray ray = MainCamera.ScreenPointToRay(screenPosition);
    Plane xy = new Plane(Vector3.forward, new Vector3(0, 0, z));
    xy.Raycast(ray, out var distance);
    return ray.GetPoint(distance);
}

public void OnDrag(PointerEventData eventData)
{
    if (!IsDraggable)
    {
        return;
    }

    if
(RectTransformUtility.ScreenPointToLocalPointInRectangle(_canvasRectTransform,
    Input.mousePosition, null, out Vector2 localPoint))
    {
        // Use the localPoint as the mouse position within the canvas
        transform.position = (Vector3)localPoint + _offset;
    }

    CardInform card = eventData.pointerDrag.GetComponent<CardInform>();
    if (_battleProcess.GetMana() >= card.Card.ManaCost)
    {
        if (_tempCardGo.transform.parent != DefaultTempCardParent)
        {
            _tempCardGo.transform.SetParent(DefaultTempCardParent);
        }
    }

    CheckPosition();

}

public void OnEndDrag(PointerEventData eventData)
{
    Cursor.visible = true;
    if (!IsDraggable)
    {
        return;
    }
    transform.SetParent(DefaultParent);
    GetComponent<CanvasGroup>().blocksRaycasts = true;
    CardInform card = eventData.pointerDrag.GetComponent<CardInform>();

    if (DestroyCard = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.EnemyPlace)
    {
        if (DestroyCard && IsDraggable)
        {

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _battleProcess.UpdateMana(-card.Card.ManaCost);
card.Card.ActiveCard(_battleProcess.PlayerCards, _battleProcess.EnemyCards);
        gameObject.SetActive(false);
        gameObject.transform.SetParent(null);
    }
    }
    else if(DestroyCard = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.PlayerPlace)
    {
        if(DestroyCard && IsDraggable)
        {
            _battleProcess.UpdateMana(-card.Card.ManaCost);
card.Card.ActiveCard(_battleProcess.PlayerCards, _battleProcess.EnemyCards);
            gameObject.SetActive(false);
            gameObject.transform.SetParent(null);
        }
    }
    // DestroyCard = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.EnemyPlace;

    transform.SetSiblingIndex(_tempCardGo.transform.GetSiblingIndex());
    _tempCardGo.transform.SetParent(GameObject.Find("BattleTableCanvas").transform);
    _tempCardGo.transform.localPosition = new Vector3 (2300, 0);

}

void CheckPosition()
{
    int newIndex = DefaultTempCardParent.childCount;

    for(int i = 0; i < DefaultTempCardParent.childCount; i++)
    {

        if(transform.position.x <
DefaultTempCardParent.GetChild(i).position.x)
        {
            newIndex = i;

            if(_tempCardGo.transform.GetSiblingIndex() < newIndex)
            {
                newIndex--;
            }

            break;
        }
    }

    _tempCardGo.transform.SetSiblingIndex(newIndex);

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
}
```

Клас Card

```
public class CardInform : MonoBehaviour
{
    [field: SerializeField] public Card Card { get; private set; }

    // [field: SerializeField] private string ID;
    [field: SerializeField] private TextMeshProUGUI _name;

    [field: SerializeField] private TextMeshProUGUI _description;
    [field: SerializeField] private TextMeshProUGUI _manaCost;
    [field: SerializeField] private float _damage;
    [field: SerializeField] private Image _logo;
    [field: SerializeField] private List<AdditionalEffects> _additionalEffects;

    public void HideCartInfo(Card card)
    {
        Card = card;

        _logo.sprite = null;

        _name.text = "";
        _description.text = "";
        _manaCost.text = "";
    }

    public void ShowCardInfo(Card card)
    {
        Card = card;

        _logo.sprite = card.Logo;

        _name.text = card.Name;
        _description.text = card.Description;
        _manaCost.text = card.ManaCost.ToString();
    }
}
```

Клас CardsInventoryHandler

```
public class CardsInventoryHandler : MonoBehaviour, IBeginDragHandler,
IDragHandler, IEndDragHandler
{
    [SerializeField] private GameObject _tempCardGo;

    [SerializeField] private RectTransform _canvasRectTransform;

    public Camera MainCamera;

    public Transform DefaultParent, DefaultTempCardParent;

    public bool IsDraggable = true;

    private BattleProcess _battleProcess;
```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private Vector3 _offset;

void Awake()
{
    MainCamera = Camera.main;
    // _tempCardGo = GameObject.Find("TempCardGo");
    _battleProcess = FindObjectOfType<BattleProcess>();
    SceneManager.sceneLoaded += OnSceneChanged;
}

private void OnEnable()
{
    if(GetComponentInParent<Canvas>())
        _canvasRectTransform =
GetComponentInParent<Canvas>().GetComponent<RectTransform>();
}

private void OnDestroy()
{
    SceneManager.sceneLoaded -= OnSceneChanged;
}

private void OnSceneChanged(Scene arg0, LoadSceneMode arg1)
{
    MainCamera = Camera.main;
}

public void OnBeginDrag(PointerEventData eventData)
{
    Cursor.visible = false;
    CardInform card = eventData.pointerDrag.GetComponent<CardInform>();

    if
(RectTransformUtility.ScreenPointToLocalPointInRectangle(_canvasRectTransform,
    Input.mousePosition, null, out Vector2 localPoint))
    {
        _offset = transform.position - (Vector3) localPoint;
    }

    DefaultParent = DefaultTempCardParent = transform.parent;

    //IsDraggable = DefaultParent.GetComponent<DropPlace>().Type ==
FieldType.SELF_HAND && GameManager.IsPlayerTurn;
    // IsDraggable = _gameProcess.IsPlayerTurn;
    // IsDraggable = true;

    if (!IsDraggable)
    {
        return;
    }

    _tempCardGo.transform.SetParent(DefaultParent);
    _tempCardGo.transform.SetSiblingIndex(transform.GetSiblingIndex());
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        transform.SetParent(DefaultParent.parent.parent);

        GetComponent<CanvasGroup>().blocksRaycasts = false;

    }

    public void OnDrag(PointerEventData eventData)
    {
        if (!IsDraggable)
        {
            return;
        }

        if (_tempCardGo.transform.parent != DefaultTempCardParent)
        {
            _tempCardGo.transform.SetParent(DefaultTempCardParent);
        }

        if (RectTransformUtility.ScreenPointToLocalPointInRectangle(_canvasRectTransform,
            Input.mousePosition, null, out Vector2 localPoint))
        {
            // Use the localPoint as the mouse position within the canvas
            transform.position = (Vector3)localPoint + _offset;
        }

        CheckPosition();

    }

    public void OnEndDrag(PointerEventData eventData)
    {
        Cursor.visible = true;
        // CardInform card = eventData.pointerDrag.GetComponent<CardInform>();
        Card card = eventData.pointerDrag.GetComponent<CardInform>().Card;

        if (DefaultParent.parent.parent.GetComponent<DropPlaceInventory>().Type
== FieldInventoryTypeType.EquippedCards)
        {
            PlayerInventory.Instance.AddNewCardToEquip(card);
        }
        else
        if (DefaultParent.parent.parent.GetComponent<DropPlaceInventory>().Type ==
FieldInventoryTypeType.AllCollectedCards)
        {
            PlayerInventory.Instance.AddNewCardToCollectedCards(card);
        }

        if (!IsDraggable)
        {
            return;
        }

        transform.SetParent(DefaultParent);
        GetComponent<CanvasGroup>().blocksRaycasts = true;
    }

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        transform.SetSiblingIndex(_tempCardGo.transform.GetSiblingIndex());
        _tempCardGo.transform.SetParent(DefaultParent.parent.parent.parent);
        _tempCardGo.transform.localPosition = new Vector3(2300, 0);

    }

    void CheckPosition()
    {

        int newIndex = DefaultTempCardParent.childCount;

        for (int i = 0; i < DefaultTempCardParent.childCount; i++)
        {

            if (transform.position.x <
DefaultTempCardParent.GetChild(i).position.x)
            {
                newIndex = i;

                if (_tempCardGo.transform.GetSiblingIndex() < newIndex)
                {
                    newIndex--;
                }

                break;
            }

        }

        _tempCardGo.transform.SetSiblingIndex(newIndex);

    }

    public void SetTempCardGo(GameObject tempCardGo)
    {
        _tempCardGo = tempCardGo;
    }

}

```

Додаток М

Клас PlayerInventory

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using ProjectF.Pool;
using TMPPro;
using UnityEngine;

namespace ProjectF
{
    [Serializable]
    public class PlayerInventory : MonoBehaviour
    {
        public static PlayerInventory Instance;
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// [field: SerializeField] public Player Player { get; private set; }
[field: SerializeField] public CardDataBase CardDataBase { get; private
set; }
[field: SerializeField] public float MaxEquippedCardsAmount { get;
private set; } = 30;
[field: SerializeField] public List<Card> EquippedCards { get; private
set; } = new List<Card>();
[field: SerializeField] public List<Card> AllCollectedCards { get;
private set; } = new List<Card>();

[SerializeField] private Transform _equippedCardsField;
[SerializeField] private GameObject _equippedCardsObject;
[SerializeField] private Transform _allCollectedCardsField;
[SerializeField] private GameObject _allCollectedCardsObject;

[SerializeField] private GameObject _tempCardGo;

[SerializeField] private CardInform _cardInformPrefab;

[SerializeField] private TextMeshProUGUI _equippedCardsText;
[SerializeField] private TextMeshProUGUI _allCollectedCardsText;
private ObjectPool<CardInform> _cardPool;

private void Awake() {
    if (Instance == null)
    {
        Instance = this;
    }
    else
    {
        Destroy(gameObject);
    }
    DontDestroyOnLoad(this.gameObject);
}

private void Start()
{
    _cardPool = new ObjectPool<CardInform>(_cardInformPrefab, 10, true);
    InventoryCheck();
    ThrowAboutCards();

    InventoryCheck();
}

public bool CheckIfExist(Card card)
{
    if(card == EquippedCards.Any() || card == AllCollectedCards.Any())
        return true;
    else
    {
        return false;
    }
}

private void UpdateText()
{
    _equippedCardsText.text = $"Карт у колоді \n {EquippedCards.Count} /
{MaxEquippedCardsAmount}";
    _allCollectedCardsText.text = $"Карт у колекції
\n{AllCollectedCards.Count.ToString()}";
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public void AddNewCardToEquip(Card card)
{
    UpdateText();
    if(AllCollectedCards.Contains(card))
        RemoveCardFromCollectedCards(card);
    EquippedCards.Add(card);
}

public void RemoveCardFromEquip(Card card)
{
    UpdateText();
    EquippedCards.Remove(card);
}

public void AddNewCardToCollectedCards(Card card)
{
    UpdateText();
    if(EquippedCards.Contains(card))
        RemoveCardFromEquip(card);
    AllCollectedCards.Add(card);
}

public void RemoveCardFromCollectedCards(Card card)
{
    UpdateText();
    AllCollectedCards.Remove(card);
}

private void ThrowAboutCards()
{
    _cardPool.HideAllElements();
    foreach (var card in AllCollectedCards)
    {
        var obj = _cardPool.GetFreeElement();
        obj.ShowCardInfo(card);
        obj.transform.SetParent(_allCollectedCardsField);
    }

    foreach (var card in EquippedCards)
    {
        var obj = _cardPool.GetFreeElement();
        obj.ShowCardInfo(card);
        obj.transform.SetParent(_equippedCardsField);
    }
    foreach (var card in _cardPool.Pool)
    {
        card.GetComponent<CardsInventoryHandler>().SetTempCardGo(_tempCardGo);
    }
    UpdateText();
}

public bool InventoryCheck()
{
    if (_allCollectedCardsObject.gameObject.activeInHierarchy &&
        _equippedCardsObject.gameObject.activeInHierarchy)
    {
        _allCollectedCardsObject.gameObject.SetActive(false);
        _equippedCardsObject.gameObject.SetActive(false);
        _equippedCardsText.gameObject.SetActive(false);
        _allCollectedCardsText.gameObject.SetActive(false);
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return false;
    }
    else
    {
        ThrowAboutCards();
        _allCollectedCardsObject.gameObject.SetActive(true);
        _equippedCardsObject.gameObject.SetActive(true);
        _equippedCardsText.gameObject.SetActive(true);
        _allCollectedCardsText.gameObject.SetActive(true);
        return true;
    }
}

public void CloseInventory()
{
    _allCollectedCardsObject.gameObject.SetActive(false);
    _equippedCardsObject.gameObject.SetActive(false);
    _equippedCardsText.gameObject.SetActive(false);
    _allCollectedCardsText.gameObject.SetActive(false);
}

public void SetInventory(List<Card> equippedCards, List<Card>
allCollectedCards)
{
    EquippedCards.Clear();
    AllCollectedCards.Clear();
    EquippedCards.AddRange(equippedCards);
    AllCollectedCards.AddRange(allCollectedCards);
    InventoryCheck();
    ThrowAboutCards();
    InventoryCheck();
}

// public void SetPlayerPosition(Vector3 position)
// {
//     Player.transform.position = position;
// }

}

```

Клас BattleProcess

```

public class BattleProcess : MonoBehaviour
{
    public static BattleProcess Instance;

    [field: SerializeField] public CardDataBase CardDataBase { get; private set; }

    [field: SerializeField] public PlayerCards PlayerCards { get; private set; }
    [field: SerializeField] public EnemyCards EnemyCards { get; private set; }

    [SerializeField] private Transform _playerHand;

    [SerializeField] private int _turn;

    [SerializeField] private TextMeshProUGUI _manaPoolText;
    [SerializeField] private int maxMana = 5;
    [SerializeField] private int manaPool = 5;

    [SerializeField] private Button _endTurnButton;
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[SerializeField] private CardInform _cardInformPrefab;

[SerializeField] private GameObject _tempCardGo;
[SerializeField] private RectTransform _canvasRectTransform;
private ObjectPool<CardInform> _cardPool;

// [SerializeField] private GameObject CardPref;

public bool IsPlayerTurn
{
    get
    {
        return _turn % 2 == 0;
    }
}

private void Awake() {
    if (Instance == null)
    {
        Instance = this;
    }
    else
    {
        Destroy(gameObject);
    }
}

private void Start()
{
    _cardPool = new ObjectPool<CardInform>(_cardInformPrefab, 10, true);
    _turn = 0;

    int i = 0;
    while(i++ < 2)
    {
        GiveCard();
    }
    ResetMana();
    UpdateMana(maxMana);
    _endTurnButton.onClick.AddListener(ChangeTurn);
}

public void ChangeTurn()
{
    if (!IsPlayerTurn)
    {
        ChangeToPlayer();
        _turn++;
    }
    else
    {
        EnemyCards.Attack((delegate
        {
            ChangeToPlayer();
            _turn++;
            _turn++;
        }
    }

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        }));
    }

}

private void ChangeToPlayer()
{
    int i = 0;
    while (i++ < 1)
    {
        GiveCard();
    }

    UpdateMana(3);
}

private void GiveCard()
{
    if(PlayerInventory.Instance.EquippedCards.All(x => x.IsUsed))
    {
        PlayerInventory.Instance.EquippedCards.ForEach(x => x.ResetCard());
    }

    Card card = PlayerInventory.Instance.EquippedCards[Random.Range(0,
PlayerInventory.Instance.EquippedCards.Count)];
    while (card.IsUsed)
    {
        card = PlayerInventory.Instance.EquippedCards[Random.Range(0,
PlayerInventory.Instance.EquippedCards.Count)];
    }

    var obj = _cardPool.GetFreeElement();
    obj.ShowCardInfo(card);
    obj.transform.SetParent(_playerHand);

obj.GetComponent<CardHandler>().SetReferences(_canvasRectTransform, _tempCardGo);
}

public void UpdateMana(int mana)
{
    manaPool += mana;
    _manaPoolText.text = $"Ма́на: {manaPool}";
}

private void ResetMana()
{
    manaPool = 0;
    _manaPoolText.text = $"Ма́на: {manaPool}";
}

public float GetMana()
{
    return manaPool;
}

public void WinBattle()
{
    SceneManager.LoadScene("SampleScene");
    GameProcess.Instance.SetIsWin(true);
    // GameProcess.Instance.GiveReward();
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public void LoseBattle()
{
    SceneManager.LoadScene(SceneManager.sceneCountInBuildSettings);
}

```

Клас DropPlace

```

public enum FieldType
{
    SelfHand,
    EnemyPlace,
    PlayerPlace
}

public class DropPlace : MonoBehaviour, IDropHandler, IPointerEnterHandler,
IPointerExitHandler
{

    public FieldType Type;

    public void OnDrop(PointerEventData eventData)
    {
        var card = eventData.pointerDrag.GetComponent<CardHandler>();

        if (card)
        {
            if (card.IsEnoughMana)
            {
                card.DefaultParent = transform;
            }
        }
    }

    public void OnPointerEnter(PointerEventData eventData)
    {
        if (eventData.pointerDrag == null)
            return;

        var card = eventData.pointerDrag.GetComponent<CardHandler>();

        if (card && card.IsEnoughMana)
        {
            card.DefaultTempCardParent = transform;
        }
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        if (eventData.pointerDrag == null)
            return;

        var card = eventData.pointerDrag.GetComponent<CardHandler>();

        if (card && card.DefaultTempCardParent == transform &&
card.IsEnoughMana)
        {
            card.DefaultTempCardParent = card.DefaultParent;
        }
    }
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
}
```

Клас GameProcess

```
public class GameProcess : MonoBehaviour
{
    public static GameProcess Instance;
    public PlayerInputActions InputActions { get; private set; }
    public PlayerInputActions.PlayerActions PlayerActions { get; private set; }

    [SerializeField] private PlayerInput _player;
    [SerializeField] private GameObject _followCamera;
    [SerializeField] private PlayerInventory _playerInventory;

    [SerializeField] private List<Card> _rewardList;
    [SerializeField] private DataManager _dataManager;

    [SerializeField] private bool _isWin = false;

    [SerializeField] private GameObject _pausePanel;
    [SerializeField] private Button _menuButton;

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            Destroy(gameObject);
        }
        DontDestroyOnLoad(this.gameObject);
        InputActions = new PlayerInputActions();

        PlayerActions = InputActions.Player;

        PlayerActions.InventoryOpen.started +=
delegate(InputAction.CallbackContext context) { OpenInventory(); };
        PlayerActions.Pause.started += delegate(InputAction.CallbackContext
context) { OpenPause(); };
        _menuButton.onClick.AddListener(delegate { ToMenu(); });
        _pausePanel.SetActive(false);
    }

    private void ToMenu()
    {
        SceneManager.LoadScene("Menu");
    }

    private void OpenPause()
    {
        if (_playerInventory.InventoryCheck())
        {
            _player.enabled = true;
            _followCamera.SetActive(true);
            _playerInventory.CloseInventory();
        }

        if(_pausePanel.activeInHierarchy)
```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _pausePanel.SetActive(false);
    else
        _pausePanel.SetActive(true);

}

private void Start()
{
    _playerInventory = PlayerInventory.Instance;
    // _dataManager.LoadData();
}

private void OpenInventory()
{
    if (_playerInventory.InventoryCheck())
    {
        _player.enabled = false;
        _followCamera.SetActive(false);
    }
    else
    {
        _player.enabled = true;
        _followCamera.SetActive(true);
    }

}

private void OnEnable()
{
    InputActions.Enable();
    SceneManager.sceneLoaded += OnSceneChanged;
}

private void OnDisable()
{
    InputActions.Disable();
    SceneManager.sceneLoaded -= OnSceneChanged;
}

private void OnSceneChanged(Scene arg0, LoadSceneMode arg1)
{
    if(FindObjectOfType<DataManager>())
        _dataManager = FindObjectOfType<DataManager>();
    if(FindObjectOfType<PlayerInput>())
        _player = FindObjectOfType<PlayerInput>();
    if(FindObjectOfType<CameraZoom>())
        _followCamera = FindObjectOfType<CameraZoom>().gameObject;
    if(!_isWin)
        GiveReward();
}

public void SetRewardList(List<Card> rewardCards)
{
    _rewardList.AddRange(rewardCards);
}

public void SetIsWin(bool isWin)
{
    _isWin = isWin;
}

```

		М.П. Коптяєв			ІІЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```
public void GiveReward()
{
    if(_rewardList.Count <= 0) return;
    _dataManager.LoadData();
    foreach (var card in _rewardList)
    {
        _playerInventory.AddNewCardToCollectedCards(card);
    }
    _rewardList.Clear();

    _dataManager.GetCurrentBattle().SetIsWin(true);
    _dataManager.SaveData();
    _isWin = false;
    // _currentBattle = null;
}
}
```

Клас GoToBattle

```
public class GoToBattle : MonoBehaviour
{
    [SerializeField] private List<Card> _rewardList;
    [SerializeField] private bool _isWin = false;
    [SerializeField] private bool _isOnBattle = false;
    [SerializeField] private DataManager _dataManager;

    private void OnTriggerEnter(Collider other)
    {
        // _dataManager.LoadData();

        if(_isWin || _isOnBattle) return;
        // return;
        _isOnBattle = true;
        _dataManager.SaveData();
        GameProcess.Instance.SetRewardList(_rewardList);
        SceneManager.LoadScene("BattleScene");
        // StartCoroutine(LoadCoroutine());
        // SceneManager.LoadScene("BattleScene", LoadSceneMode.Additive);
    }

    private IEnumerator LoadCoroutine()
    {
        yield return new WaitForSeconds(.2f);
    }

    public void SetIsWin(bool isWin)
    {
        _isWin = isWin;
        if (_isWin)
        {
            gameObject.SetActive(false);
            _isOnBattle = false;
        }
        // if(_isWin)
        //     _trigger.SetActive(false);
        // else
        //     _trigger.SetActive(false);
    }

    public void SetIsOnBattle(bool isOnBattle)
```

```
{
    _isOnBattle = isOnBattle;
}

public bool GetIsWin()
{
    return _isWin;
}
public bool GetIsOnBattle()
{
    return _isOnBattle;
}
}
```

		М.П. Коптяєв			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		