

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:

**«Комп'ютерна гра у жанрі метроїдванія “Dark
Watcher” – діалогова система, система торгівлі»**

Виконав студент групи ІПЗ-19-1
КОЗАКЕВИЧ Наталія Олександрівна

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2023

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: **КОЗАКЕВИЧ Наталія Олександрівна**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Комп'ютерна гра у жанрі метроїдванія "Dark Watcher" - діалогова система, система торгівлі»,**

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

Вихідні дані для роботи: **Об'єктом дослідження є комп'ютерна гра "Dark Watcher" у жанрі метроїдванія. Предметом дослідження є системи діалогу та торгівлі, які використовуються у грі.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	14.02.2023	14.02.2023
2	Морозов А.В.	14.02.2023	14.02.2023
3	Морозов А.В.	14.02.2023	14.02.2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	02 березня 2023 – 05 квітня 2023	
2	Проектування структури системи	6 квітня 2023 – 12 квітня 2023	
3	Написання програмного коду	13 квітня 2023 – 26 квітня 2023	
4	Тестування	27 квітня 2023 – 04 травня 2023	
5	Підготовка пояснювальної записки	5 травня 2023 – 12 червня 2023	

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу гри у жанрі метроїдванія "Dark Watcher" і супроводжуючої пояснювальної записки. Пояснювальна записка включає 87 сторінок, 23 ілюстрації та 37 таблиць, а також один додаток.

Метою даної роботи є розробка гри у жанрі метроїдванія за допомогою рушія Unity. У роботі були визначені основні завдання щодо розробки системи, проведений аналіз аналогічних систем, наведені сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи та об'єктна структура системи (діаграма класів). Крім того, у роботі приведені результати тестування системи.

Ключові слова: "Dark Watcher", метроїдванія, 2D, діалог, ігрова пригода, торгівля.

					ІПЗ.КР.Б. – 121 – 23 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	Комп'ютерна гра в жанрі «метроїдванія» — діалогова система, система торгівлі. Пояснювальна записка		
Розроб.		Н.О. Козакевич					
Керівник		А.В. Морозов					
Н. контр.		О.П.Тичина					
Затвердив		А.В.Морозов			Літ. Арк. Аркушів		
					4 88		
					Житомирська політехніка, група ІПЗ-19-1		

ABSTRACT

The bachelor's thesis consists of the software complex of the game in the metroidvania genre "Dark Watcher" and the accompanying explanatory note. The explanatory note includes 87 pages, 23 illustrations and 37 tables, as well as one appendix.

The purpose of this work is to develop a game in the metroidvania genre using the Unity engine. In the work, the main tasks related to the development of the system were defined, the analysis of similar systems was carried out, the scenarios of the software complex, the general structure of the software complex, the description of the interaction algorithms of individual system modules and the object structure of the system (class diagram) were given. In addition, the work presents the results of system testing.

Keywords: Dark Watcher, Metroidvania, 2D, dialogue, game adventure, trade.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

2D – 2-dimentional

3D – 3-dimentional

ОС – операційна система

UES – Unity Event System

NPC – Non-Player Character

SEC – System-Entity-Component

UML – Unified Modeling Language

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				6
Змн.	Арк.	№ докум.	Підпис	Дата		

Зміст

ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР	12
1.1 Постановка задачі	12
1.2 Аналіз аналогів програмного продукту.....	12
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	29
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	29
2.2 Розробка подій та елементів гри	37
2.3 Проектування та реалізація алгоритмів роботи системи	42
2.4 Реалізація ігрового додатку.....	53
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ В ЖАНРІ МЕТРОЇДВАНІЯ “DARK WATNCER”	68
3.1 Порядок встановлення та налаштування системи	68
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з системою симуляції	69
3.3 Тестування роботи системи.....	73
ВИСНОВКИ	81
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	83
ДОДАТКИ.....	85
Додаток А. Відгуки на кваліфікаційну роботу	86

ВСТУП

Для уважного спостерігача, важко було б проігнорувати той факт, що комп'ютерні ігри в сучасному світі займають все більш важливе місце в повсякденному житті людини. За останні роки ігрова індустрія продемонструвала вражаючий ріст і розвиток, завоювавши місце в серцях мільйонів гравців по всьому світу. Сьогодні багато людей регулярно приділяють свій особистий час іграм, щоб насолодитися, розслабитися або викликати емоції, які можуть бути недоступні в реальному світі.

Серед різноманіття жанрів ігор одним з найпопулярніших і цікавих є метроїдванія. Цей жанр зародився в 1986 році з виходом культової гри «Metroid» на платформі Nintendo Entertainment System і в свій час став одним з найуспішніших ігрових трендів. Ключова ідея метроїдванії полягає в дослідженні гравцем ігрового світу, частини якого тісно пов'язані між собою, знаходженні нових зон, отриманні інструментів та навичок, що допоможуть вирішити недоступні раніше задачі на попередніх територіях.

У зв'язку з цим хочемо звернути увагу на комп'ютерну гру «Dark Watcher», яка відноситься до жанру метроїдванія. Цей ігровий застосунок забезпечує особливий інтерактивний досвід, який може утримувати увагу гравця з самого початку. Разом із інтригуючим сюжетом і приголомшливими візуальними ефектами, «Dark Watcher» також дозволяє гравцеві відчути, як це жити у вигаданому місці, долаючи різноманітні складні перешкоди.

Перевага «Dark Watcher» полягає в тому, що вона допомагає гравцям покращити когнітивні та моторні навички, стратегічне мислення та здатність швидко діяти під тиском. Гра сприяє розвитку творчих здібностей, адже гравці можуть експериментувати при взаємодії зі світом через неігрових персонажів і цим змінювати глобальний сюжет гри.

У зв'язку з тим, що комп'ютерні ігри стають все більш популярними і мають значний вплив на сучасну культуру, усім хто цікавиться цією темою

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				8
Змн.	Арк.	№ докум.	Підпис	Дата		

та хоче дізнатися більше про захоплюючий світ ігрової індустрії, важливо охопити особливості та переваги жанру метроїдванія, а також ігрового досвіду у «Dark Watcher».

Отож, у цьому проекті ми докладніше розглянемо системи діалогу та торгівлі, відстоюючи їх значення та реальне використання. Дана тема є актуальною для розуміння сучасних тенденцій розвитку ігрової індустрії та формування ігрового досвіду гравців.

Об'єктом та предметом дослідження є:

Об'єктом дослідження є відеогра у жанрі метроїдванія «Dark Watcher». Предметом дослідження є діалогова та торгівельна системи.

Специфіка системи діалогу, яка дозволяє гравцеві взаємодіяти з неігровими персонажами та просувати сюжет гри, буде розглянута через аналіз цього предмета дослідження. Можливість отримання та обміну предметів між головним героєм та неігровими персонажами буде обговорюватись у розгляді торгівельної системи.

Мета дослідження цих елементів — визначити, наскільки вони важливі та як вони впливають на ігровий процес «Dark Watcher». Розуміння поточних тенденцій у створенні комп'ютерних ігор і покращенні ігрового досвіду для гравців допоможе як з аналізу результатів досліджень розробників ігор, так і попереднім дослідженням у цій галузі.

Ключові особливості систем діалогу та торгівлі «Dark Watcher» обговорюватимуться з використанням таких методів дослідження, як аналіз, порівняння та моделювання, щоб визначити, як вони функціонують та взаємодіють з грою.

Для того, щоб розкрити специфіку діалогової та торгівельної системи у цій комп'ютерній грі, зрозуміти їх вплив на ігровий процес і покращити ігровий досвід гравців, буде вивчено об'єкт і предмет дослідження «Dark Watcher».

Методи аналізу та інформаційна база дослідження:

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Кваліфікаційну роботу бакалавра розроблено з використанням методів аналізу літературних джерел і досліджень, аналізу ігрового досвіду та експериментального моделювання.

Джерелом даних дослідження були наукові праці, книги, записи та коментарі розробників комп'ютерних ігор. Крім того, використовувалася інформація з офіційних сайтів і дискусійних форумів ігор, схожих на «Dark Watcher».

Натхненням для розробки кваліфікаційної роботи виступили такі ігри, як «Blasphemous», «Ori and the Blind Forest», «Hollow Knight» і «Fran Bow». Ці витвори ігрової індустрії надали натхнення та знання, при дослідженні яких, було виконано роботу над діалоговими та торгівельною системами.

Гра «Dark Watcher» створена на платформі Unity, яка пропонує зручний інтерфейс і широкий набір інструментів для створення ігор високого рівня. «Dark Watcher» використовує Unity для реалізації систем торгівлі та діалогу.

У результаті, під час розробки гри на платформі Unity «Dark Watcher», черпалося натхнення популярними іграми у жанрі метроїдванія, передбачене застосування методів аналізу, заснованих на літературних джерелах та ігровому досвіді, а також використання дослідницької бази даних що включає наукові джерела та досвід розробників ігор.

Практичне значення одержаних результатів:

Результати отримані у цьому дослідженні мають велике практичне значення для розробників комп'ютерних ігор і ігрової індустрії в цілому.

Вивчення діалогової та торгівельної систем Dark Watcher має змогу навчити розробників корисним стратегіям та вказівкам для включення цих функцій в інші ігри в жанрі метроїдванія. Це дозволяє створити збалансовану і захоплюючу гру, а також підвищити якість ігрового процесу.

Результати дослідження також застосовуються до відеогри "Dark Watcher", щоб підвищити її візуальну привабливість і рівень занурення.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Використовуються найкращі практики та розвиваються оригінальні ігрові взаємодії, завдяки знанням елементів ігрового процесу, які успішно використовувалися у відомих іграх.

Крім того, результати можуть стати відправною точкою для подальшого вивчення та просування індустрії комп'ютерних ігор. Вони сприяють створенню свіжих, передових ідей для діалогової та торгівельної систем, збагачуючи сферу навчання та створюючи нові можливості для експериментів і творчості.

Таким чином, удосконалення процесу створення комп'ютерних ігор, підвищення рівня ігрового процесу та створення більш захоплюючого та вражаючого ігрового досвіду для гравців, є практичними наслідками отриманих результатів.

Структура та обсяг проекту (роботи). Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 18 найменувань (2 сторінки), 1-м додатком, 37 таблицями та 23 ілюстраціями по тексту. Повний обсяг роботи становить 87 сторінок, з них 84 сторінок основного тексту.

		<i>Н.О.Козакевич</i>			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		<i>А.В. Морозов</i>				11
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР

1.1 Постановка задачі

Основним завданням дослідження є розробка та реалізація комп'ютерної гри в жанрі метроїдванія «Dark Watcher». Для досягнення поставленої мети необхідно задовільнити такі підзадачі:

1. Розробка діалогової системи:

- 1.1. Проектування архітектури інтерактивної системи для взаємодії гравця з неігровими персонажами.
- 1.2. Реалізація механізму обробки та відтворення діалогів.
- 1.3. Розробка інтерфейсу користувача для взаємодії з діалоговою системою.
- 1.4. Розробка системи інтерактивних завдань, що будуть сюжетно пов'язані з діалогами.

2. Розробка торгівельної системи:

- 2.1. Розробка системи що дозволяє гравцеві купувати предмети у грі.
- 2.2. Реалізація взаємодії з продавцем та обміну предметами.
- 2.3. Розробка інтерфейсу користувача для проведення торгівельних операцій.

Після успішного виконання цих підзадач комп'ютерна гра «Dark Watcher» буде готова до тестування та задоволення гравців. Основна мета — створити захоплюючий ігровий досвід у жанрі Metroidvania, надаючи гравцям цікаву графіку, відточену діалогову систему та цікаву систему торгівлі.

1.2 Аналіз аналогів програмного продукту

Виходячи з вимог до системи та враховуючи специфіку предметної області, проведемо аналіз. Результати цього аналізу стануть основою для

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				12
Змн.	Арк.	№ докум.	Підпис	Дата		

вибору платформи, технологій та середовища розробки програмного забезпечення. Для обґрунтування вибору використовуються методи прийняття рішень.

Hollow Knight – ця гра відзначається великою глибиною геймплею та складністю, а також вражаючою візуальною естетикою [16]. Вона приховує багато таємниць і заплутаних маршрутів, які варто досліджувати й розгадувати. Вас чекають зустрічі з різноманітними противниками й потужними босами, які вимагатимуть стратегічного мислення та вправності в боях. Додатково, захоплюючий саундтрек та атмосфера гри повністю поглинуть вас у цей вигаданий світ.

При створенні ігрового додатку гра Hollow Knight буде еталоном якісної розробки метроїдванії, враховуючи її популярність, пропрацьованість та захопливість, вона може служити натхненням та прикладом для інших розробників гейміндустрії. Гра успішно демонструє, що навіть у віці суворої конкуренції та швидкої зміни технологій, якість геймплею та мистецтво можуть піднімати гру на новий рівень.

Плюси:

- захопливий світ, що приваблює;
- глибока та емоційна історія;
- велике розмаїття супротивників;
- естетичний дизайн;
- різноманітні виклики.

Мінуси:

- можлива висока складність
- вимагає великих навичок та терпіння для успішного проходження;
- ризик втрати прогресу при отриманні значних ушкоджень;
- відсутність української локалізації.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				13
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.1. Hollow Knight

Blasphemous – гра, створена The Game Kitchen та випущена у 2019 році, представляє собою гібрид метроїдванії, платформера та екшену. Дія відбувається у чарівному світі під назвою Cystodia, де панує готична атмосфера та релігійні мотиви. Гравець приймає роль Міріам, молитовника що володіє мечем "Меч Переродження", чия мета — подолати ворогів, розв'язати головоломки та розкрити темні таємниці гри. Blasphemous привертає увагу своїм безжальним геймплеєм, ретро-пиксельною графікою та насиченою атмосферою.

Blasphemous отримала позитивні відгуки від геймерської спільноти та отримала визнання за свою якість та оригінальність.

Плюси:

- унікальність художнього супроводження;
- естетичність атмосфери;
- велике розмаїття супротивників;
- глибокий світ.

Мінуси:

- можлива висока складність;

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				14
Змн.	Арк.	№ докум.	Підпис	Дата		

- незручність системи керування;
- відсутність української локалізації.



Рис. 1.2. Blasphemous

A Castle Full of Cats – поєднує у собі такі жанри як метроїдванія та пошук предметів із унікальними результатами [17]. Гра захоплює нас пошуком котів у проклятому замку. Через прокляття всі коти були перетворені на спотворені версії самих себе. Щоб звільнити замок і котів, вам доведеться побродити всюди, щоб знайти понад 500 котів. Знайдіть таємні входи та проходи, щоб увійти в більше місць і знайти більше котів.

Плюси:

- унікальність художній стиль;
- захоплююча атмосфера;
- цікава концепція.

Мінуси:

- відсутність розмаїття ігрового процесу;
- відсутність української локалізації.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				15
Змн.	Арк.	№ докум.	Підпис	Дата		

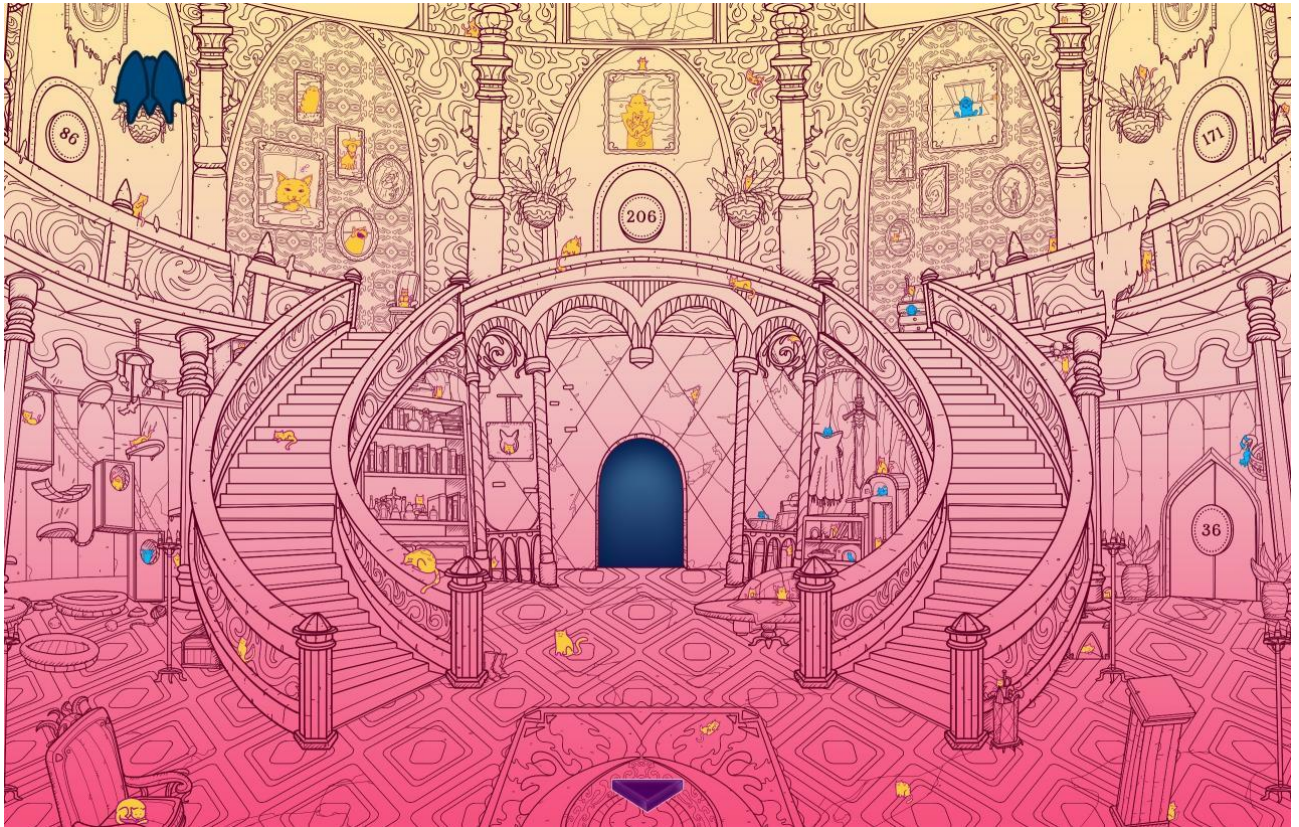


Рис. 1.3. A Castle Full of Cats

Ori and the Blind Forest – представлена гравцям у 2015 році Moon Studios, екшн-платформер, випущений Microsoft Studios. Гра відзначається неперевершеною візуальною презентацією та захоплюючим саундтреком. Головний герой — Орі, маленька біла істота з унікальними здібностями, яка прагне врятувати свій ліс від знищення. Гравець може насолоджуватися різноманітними режимами гри, розв'язувати головоломки та зустрічати ворогів. "Ori and the Blind Forest" отримала високі оцінки за свою візуальну привабливість, музику та геймплей.

Плюси:

- графічна презентація, яка захоплює;
- емоційно насичена історія;
- інноваційна механіка;
- гармонійне злиття різних жанрів;
- висока якість аудіо-супроводу.

Мінуси:

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				16
Змн.	Арк.	№ докум.	Підпис	Дата		

- висока складність в окремих моментах гри;
- обмежені можливості налаштування графіки та інтерфейсу;
- відсутність подальшого продовження після завершення гри;
- відсутність української локалізації.

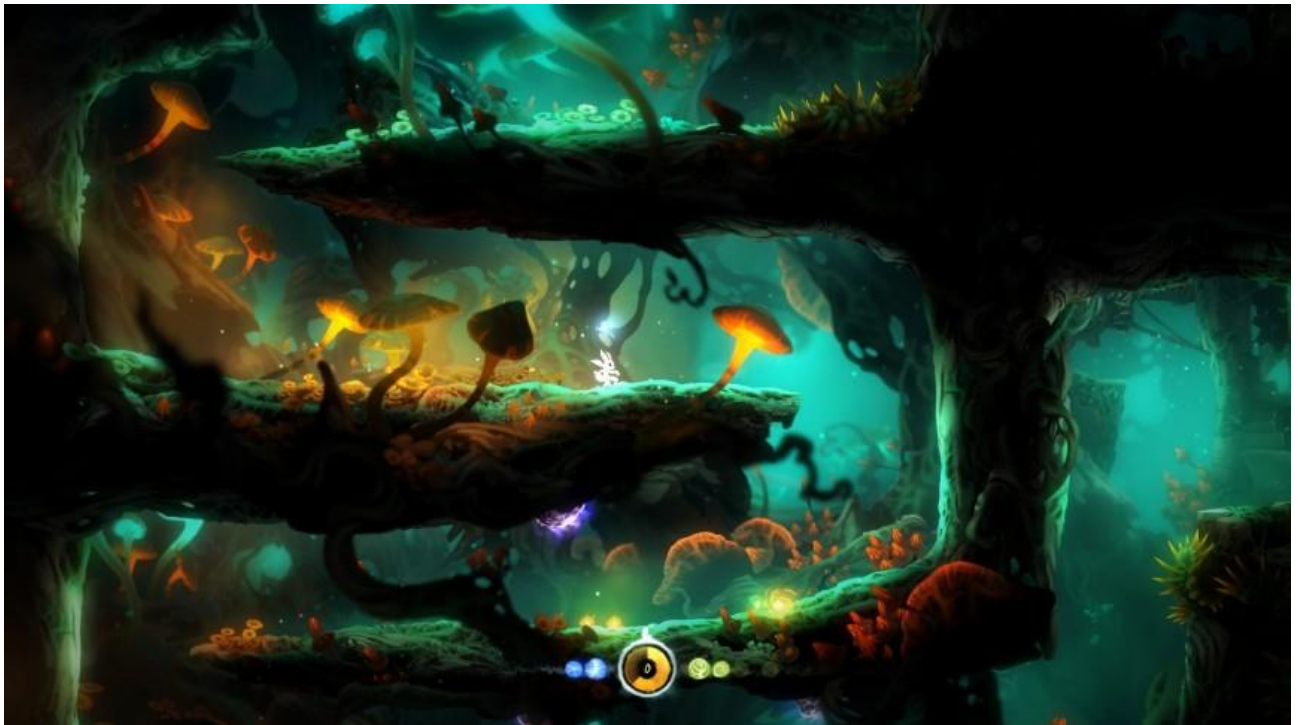


Рис. 1.4. Ori and the Blind Forest

Щоб усвідомити хід розробки подібних додатків та відповідально підійти до цієї роботи, необхідно сформуванати таблицю для відмінностей у розробці, а саме з якої кількості людей складалася команда та який рушій було використано при розробці.

Наведена нижче таблиця містить у собі інформацію про рушій та мову програмування [18] для кожної гри з наведених ігор, ця інформація є істотною з декількох причин:

Технічна направленість: знання рушію та мови програмування, які використовуються для створення гри, можуть бути корисними розробникам ігор або тим, хто цікавиться технічними аспектами розробки. Це дає уявлення про те, які інструменти та технології використовуються в процесі створення певних ігор.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Дослідження рушію розробників: знання рушію та мови програмування можуть допомогти вивчити підходи що застосовуються при розробці ігор. Ця інформація може бути особливо корисною для тих хто цікавиться інді-розробкою ігор або хоче зрозуміти, які технології використовуються в певних проектах.

Обрання технологій розробки: для розробки власної гри, знання движка і мови програмування може вплинути на вибір певної технології. Порівняння движків і мов допоможе зрозуміти їх переваги та недоліки, а також знайти найбільш підходящу технологію для конкретного проекту.

Відстеження популярних тенденцій: Порівняльна інформація про рушії та мови програмування, що використовуються в популярних іграх, може допомогти зрозуміти тенденції та напрямки розвитку індустрії розробки ігор. Це дозволяє бути в курсі нових технологій і допомагає розробникам приймати зважені рішення.

Таблиця 1.1. Стек конкурентних додатків

Гра	Рушій	Мова програм ування	Кількість розробників
Hollow Knight	Unity	C#	15-20
A Castle Full of Cats	Godot Engine	C#	6
Blasphemous	GameMaker Studio 2	GML	10-15
Ori and the Blind Forest	Unity	C#	20-25

Варто зазначити що кількість розробників у наведеній таблиці включає лише актуальних розробників, тобто тих хто приймав участь від альфа-версії до випуску гри.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибираючи мову розробки, варто звернути увагу на кількість доступної літератури, документації та спільноти, яка допоможе вирішити проблеми, якщо інші джерела не надають відповідей на неочевидні проблеми.

При виборі стеку було обрано мову програмування C#.

1.3.1. Мова програмування C#:

Значний обсяг використання: C# — одна з найпопулярніших мов програмування в розробці ігор, що широко використовується в основних ігрових движках, таких як Unity [7] та Unreal Engine, а також у розробці ігор для різноманітних платформ, включаючи консолі, мобільні пристрої та персональні комп'ютери. Що у висновку робить C# однією з основних мов програмування для розробки ігор.

Простота вивчення: C# має чистий і логічний синтаксис, який дружній до новачків у розробці ігор. Вона надає значну кількість готових бібліотек і інструментів, які спрощують процес розробки. Крім того, доступно багато матеріалів і ресурсів для вивчення, розбору та поглиблення у C#, а також розробки ігор, написаних цією мовою програмування, що робить її привабливою для початківців.

Взаємопроникнення з ігровими рушіями: C# має потужну інтеграцію з популярними ігровими рушіями, зокрема з Unity. Unity використовує C# як основну мову програмування, що робить її надійним вибором для створення ігор на цій платформі. Вибір цієї мови дозволяє розробникам продуктивно працювати з готовими ігровими рушіями та використовувати багатofункціональність, що вони надають.

Взаємодія і підтримка: C# має активну спільноту розробників, яка надає допомогу, підтримку та новітні інструменти. Наявність форумів, блогів, документації та інших ресурсів, де можна знайти відповіді на

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				19
Змн.	Арк.	№ докум.	Підпис	Дата		

запитання та отримати допомогу у вирішенні проблем, надає гнучку допомогу при розробці. Більше того, взаємопроникнення C# з іншими мовами і технологіями дозволяє легко розширювати можливості і функціональність розроблених ігор.

Вибір оптимальної мови програмування залежить від конкретної ситуації та вимог проекту. Тим не менш, C# має кілька важливих переваг, які роблять його привабливим варіантом для створення ігор у порівнянні з іншими мовами програмування:

Простота вивчення та використання: C# має простий і послідовний синтаксис, що є ідеальним для початківців у галузі розробки ігор. Вона також пропонує широкий вибір готових бібліотек та інструментів, які спрощують процес розробки. Більше того, існує велика кількість навчальних матеріалів, курсів і ресурсів, які допомагають новим розробникам швидко освоїти C# і розпочати роботу над своїм проектом.

Платформна незалежність: C# — це мова, яка підтримує кросплатформну розробку. Це означає, що код, написаний на C#, можна виконувати на різних платформах, включаючи консолі, мобільні пристрої та персональні комп'ютери.

Продуктивність: C# має високу продуктивність і швидкість. Це вкрай важливо для розробки великих ігрових проектів з високими вимогами до складності та продуктивності.

Загалом вибір мови програмування C# для розробки ігор виправданий її популярністю, простотою вивчення, інтеграцією з ігровими рушіями та наявністю активної екосистеми підтримки. Це один з основних інструментів для створення як невеликих інді-проектів, так і великих AAA-ігор.

1.3.2. Програмне забезпечення C# Mono

C# Mono (також відомий як Mono) — це програмне забезпечення з відкритим кодом, яке забезпечує інтерпретатор і середовище виконання для мови програмування C#. Воно було розроблене компанією Xamarin (пізніше

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				20
Змн.	Арк.	№ докум.	Підпис	Дата		

придбаною Microsoft) з метою створення міжплатформного середовища для розробки додатків C# [8].

Моно надає змогу розробникам писати програми на C# та запускати їх у різних операційних системах, таких як Windows, macOS та Linux. Це забезпечує кросплатформну сумісність, що дозволяє розгортати програми на різних пристроях і платформах без необхідності переписування коду.

Моно містить інтерпретатор і компілятор C#, які дозволяють запускати програми на C#. Воно також підтримує багату функціональність мови, враховуючи об'єктно-орієнтовану парадигму, обробку винятків, доступ до .NET Framework API та інший функціонал.

Одним з ключових аспектів Моно є його використання в розробці ігор. Ігровий рушій Unity, популярний у галузі розробки ігор, використовується для створення ігор на C# з використанням Моно. Це дає змогу розробникам створювати кросплатформні ігри, які можна запускати на різних пристроях, включаючи персональні комп'ютери, ігрові консолі та мобільні пристрої.

C# Моно — це наявність активної підтримки спільноти та розробників, регулярність випуску оновлень для покращення продуктивності, безпеки та функціональності Моно. Більше того, існує багато ресурсів і документації, які допоможуть розробникам працювати з C# Моно.

Загалом, C# Моно — це потужний інструмент для розробки кросплатформних додатків C#. Він забезпечує можливість розгортання програм на різних операційних системах і платформах, впроваджуючи гнучкість та широке охоплення аудиторії.

Саме тому було обрано цю мову для розробки гри.

Обравши мову програмування можемо обирати ігровий двигун, для мови програмування C# їх є декілька, розглянемо їх перелік за популярністю.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				21
Змн.	Арк.	№ докум.	Підпис	Дата		

– Unity — один з найбільш популярних ігрових рушіїв, використовує C# як основну мову програмування. Він забезпечує потужні засоби для розробки ігор, включаючи 2D і 3D графіку, фізику, анімацію, штучний інтелект та інше. Unity також має широку спільноту розробників, розширень і ресурсів, які полегшують процес розробки ігор;

– Godot Engine — безкоштовний та відкритий ігровий рушій, який також підтримує мову програмування C# для розробки ігор. Він пропонує простий у використанні інтерфейс та потужні інструменти для створення графіки, фізики, анімації та багато іншого. Godot Engine продовжує активно розвиватись і має живу спільноту розробників;

– Xenko — відкритий ігровий рушій, який використовує мову програмування C# для розробки ігор. Він надає розробникам інструменти для створення як 2D, так і 3D ігор, включаючи графіку, фізику, звук, штучний інтелект та інші компоненти. Крім того, Xenko має вбудовану підтримку віртуальної реальності;

– Wave Engine — ігровий рушій, який підтримує мову програмування C# для розробки ігор, надає розробникам потужні інструменти для роботи з графікою, фізикою, звуком, штучним інтелектом та іншими аспектами ігрової розробки. Він також підтримує кросплатформну розробку і може використовуватися для створення ігор для різних платформ.

Ігровим рушієм для розробки ігор обрано Unity з використанням мови C# та це має певне обґрунтування:

– популярність: Unity є одним з найпопулярніших ігрових рушіїв у сфері розробки ігор. З Unity вже працює велика кількість розробників, що сприяє наявності великої спільноти, ресурсів і документації. Це забезпечує доступ до підтримки, порад і усунення несправностей від інших розробників;

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				22
Змн.	Арк.	№ докум.	Підпис	Дата		

– використання C#: основною мовою програмування в Unity є C#, що робить розробку ігор ефективнішою для тих, хто вже має досвід роботи з цією мовою програмування або .NET-платформою. C# — це потужна сучасна мова з багатим функціоналом, яка дозволяє легко впроваджувати різні ігрові механізми та взаємодії;

– кросплатформність: Unity надає можливість розробляти ігри для різноманітних платформ, включаючи персональні комп'ютери, ігрові консолі, мобільні пристрої та віртуальну реальність. Це означає, можливість створення ігор, що легко запускаються на різних пристроях без необхідності повторного написання коду;

– поєднання інструментів: Unity пропонує вбудовані потужні інструменти для розробки графіки, фізики, звуку, анімації та багатьох інших аспектів розробки ігор. Він також забезпечує можливості візуального редагування, швидкого прототипування та налагодження, що спрощує процес розробки.

Загалом, вибір Unity як ігрового двигуна для розробки ігор з використанням мови C# дозволяє розробникам сфокусуватися на творчому процесі, надаючи їм потужні інструменти та підтримку, які спрощують розробку високоякісних ігрових проектів.

Unity Event System (UES) це набір інструментів та компонентів для розробки інтерфейсів всередині фреймворку Unity [9]. Є частиною середовища розробки, що дозволяє легко та ефективно взаємодіяти з внутрішньо ігровими елементами. UES дозволяє встановлювати слухачів на різні об'єкти гри які працюють за методом слухання Mutex. Події можуть відрізнятися одна від одної в залежності від написання окремого компоненту. UES має переваги та недоліки що зазначені нижче:

Переваги використання Unity Event System:

– легкість використання: має простий та зрозумілий інтерфейс що дозволяє розробникам легко налаштовувати та розробляти події;

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				23
Змн.	Арк.	№ докум.	Підпис	Дата		

- розширюваність: гнучкість системи дозволяє створювати власні спеціалізовані події та розширювати їх в залежності від потреб проекту;
- інтеграція з іншими системами: UES чудово інтегрується з іншими дочірніми бібліотеками Unity такими як Unity UI, Unity Physical Library, Unity Animation, а також зі сторонніми NuGet пакетами;
- відокремленість логіки: система подій дозволяє відокремлювати логіку обробки подій об'єктів, що робить код більш чистим та організованим.

Недоліки використання Unity Event System:

- відсутність вбудованої валідації параметрів: UES не має вбудованих засобів для перевірки аргументів наданих в обробник подій, що може призвести до помилок якщо параметри не відповідають заданому формату;
- відсутність підтримки зовнішніх сервісів: UES не включає в себе вбудовану підтримку зовнішніх сервісів, що означає, що для їх інтеграції може бути необхідний власний код або використання додаткових бібліотек;
- потреба у пам'яті: використання UES вимагає певної кількості пам'яті для зберігання даних про події та підписників. У проектах з великою кількістю об'єктів та подій це може призвести до збільшення обсягу використаної пам'яті.

Скриптовані об'єкти — це тип об'єктів що дозволяє зберігати, організовувати та маніпулювати даними ігрових станів всередині гри [1]. Вони є контейнерами що містять в собі поля, властивості та методи схожі на класи, але їх можна створювати, налаштовувати безпосередньо в редакторі Unity без повторного написання коду [5].

Переваги використання Scriptable Objects:

- зручність створення та редагування: Scriptable Objects можуть бути легко створені та налаштовані безпосередньо в редакторі Unity,

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				24
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяючи змінювати їх значення без потреби редагування вихідного коду. Це значно спрощує процес розробки та налагодження гри;

- модульність та повторне використання: Scriptable Objects можуть бути використані як незалежні модулі [2], які зберігають дані та логіку, і можуть бути використані в різних частинах гри. Вони дозволяють повторно використовувати дані та логіку в різних сценах, рівнях або навіть в різних проектах;

- зручне зберігання та передача даних: забезпечення зручного способу зберігання та передачі даних. Scriptable Objects можуть бути легко серіалізовані, що дозволяє зберігати їх у файлах або передавати через мережу [3]. Це корисно, наприклад, при роботі зі збереженням/завантаженням стану гри або імпортом/експортом налаштувань;

- редагування в реальному часі: Scriptable Objects можуть бути редаговані в реальному часі під час виконання гри [4]. Це означає, можливість зміни їх значень в редакторі Unity під час гри, і застосовані зміни відобразяться без необхідності перезапуску. Це дає можливість налаштовувати параметри гри на льоту, що дозволяє зручно експериментувати та налаштовувати ігровий процес;

- підтримка версіонування та співпраці: Scriptable Objects ідеально підходять для спільної роботи декількох розробників над проектом. Вони можуть бути збережені в системі контролю версій і легко об'єднані, спрощуючи командну розробку.

Недоліки використання Scriptable Objects:

- велика кількість об'єктів: використання багатьох Scriptable Objects може призвести до збільшення кількості об'єктів у проекті. Це може вплинути на продуктивність та використання пам'яті;

- важкість налагодження: під час налагодження гри, редагування Scriptable Objects може бути більш складним, оскільки вони не мають

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				25
Змн.	Арк.	№ докум.	Підпис	Дата		

прямого доступу до інструментів для налагодження, які доступні для вихідного коду;

- потенційні проблеми з серіалізацією: Scriptable Objects можуть виявитися вразливими до проблем, пов'язаних з серіалізацією даних. Це може виникнути при зміні структури даних або введенні змін до коду;

- обмежена підтримка платформ: деякі платформи, особливо ті, які не підтримують Mono або .NET Framework, можуть мати обмежену або відсутню підтримку для Scriptable Objects та C#;

- висока складність: використання Scriptable Objects вимагає додаткового розуміння їх концепцій та специфіки роботи з ними. Це може збільшити складність проекту та вимагати додаткового зусилля від розробників.

Обґрунтування мінімальних системних вимог для 2D гри на Unity, які включають ОС Windows 7, процесор Intel Core 2 Duo E5200, 4 ГБ оперативної пам'яті, відеокарту GeForce 9800GTX+ з 1 ГБ пам'яті, DirectX 10 та 2 ГБ доступного місця на диску, можна сформулювати таким чином:

- сумісність: оскільки ОС Windows 7 є широко поширеною та користувачами підтримується, вимога до неї забезпечує сумісність гри з багатьма комп'ютерами, що працюють на цій операційній системі;

- розповсюдженість обладнання: процесор Intel Core 2 Duo E5200 є представником середнього класу, який був випущений багато років тому. Вимога до такого процесора враховує розповсюдженість середнього та старшого класу комп'ютерів, що дозволяє грі запускатися на більшості старіших систем;

- мінімальні ресурси: вимоги до оперативної пам'яті (4 ГБ) та доступного місця на диску (2 ГБ) відображають мінімальні потреби гри. Це гарантує стабільну роботу гри на системах з обмеженими ресурсами;

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				26
Змн.	Арк.	№ докум.	Підпис	Дата		

– графічні можливості: відеокарта GeForce 9800GTX+ є старішою, але все ще здатною відтворювати 2D графіку. Вимога до цієї відеокарти забезпечує підтримку відображення зображень та анімацій у грі;

– DirectX 10: вимога до DirectX 10 гарантує, що гра використовує сучасні можливості графічної бібліотеки DirectX, що забезпечує покращену візуальну якість та ефекти.

З урахуванням цих факторів, вимоги до системи були встановлені таким чином, щоб гра могла працювати на різних комп'ютерах, забезпечуючи оптимальну продуктивність та відтворення 2D графіки. Описані мінімальні системні вимоги дозволяють гравцям зі старішими або менш потужними комп'ютерами насолоджуватися грою без значних проблем з продуктивністю чи відтворенням графіки.

Висновки до першого розділу

Під час порівняння та аналізу аналогів різних варіантів мов програмування, рушіїв та обраних технологій, було прийнято рішення використовувати мову програмування C#, ігровий рушій Unity, систему Unity Event System та інтерфейс Scriptable Objects. Дане рішення складалося на основі таких висновків:

Мова програмування C#:

– одна з основних мов програмування для розробки ігор;
– чистий і логічний синтаксис;
– значна кількість готових бібліотек та інструментів;
– потужна інтеграція з популярними ігровими рушіями, зокрема з Unity;

– наявність форумів, блогів, документації та інших ресурсів.

Ігровий рушій Unity:

– популярність ігрового рушію;
– потужні засоби для розробки ігор;
– широка спільнота розробників.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Система Unity Event System:

- простий та зрозумілий інтерфейс;
- гнучкість системи;
- чудово інтегрується з іншими дочірніми бібліотеками Unity;
- відокремленість логіки для чистого та організованого коду.

Інтерфейс Scriptable Objects:

- зручність створення та редагування;
- модульність та повторне використання;
- зручне зберігання та передача даних;
- підтримка версіонування.

Після аналізу було встановлено, що найкращим варіантом для розробки є використання мови програмування C# разом з ігровим рушієм Unity, системою Unity Event System та інтерфейсом Scriptable Objects. Цей вибір надає широкий спектр можливостей для розробки ігор, гарантує високу продуктивність та ефективність роботи, а також дозволяє використовувати готові рішення та компоненти для прискорення процесу розробки.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				28
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Перед початком розробки проект поділяється на певні етапи загального проектування, які включають в себе важливі елементи, що допомагають при розробці програмного продукту. В цьому розділі виконується детальне планування та опис проекту, забезпечуючи його якісну і успішну реалізацію. Також виконано розподілення ролей у команді, при яких кожен її член виконує певну задачу наведену у таблиці 2.1.

У цьому розділі ми розглянемо такі елементи:

1. Технічне завдання: надається ретельний опис вимог і функціональності програмного продукту, що включає докладну інформацію про цільову аудиторію, функціональні вимоги, необхідні інтеграції та інші важливі деталі, спрямовані на точне сприйняття та розуміння проекту.
2. Сюжетна складова: використання математичних моделей для побудови сюжетної логіки. Включає в себе графові систему діалогів та часово-просторовий обмежений послідовний відрізок подій ігрових пригод.

Ці складові сприяють осмисленню та структуруванню всіх аспектів проектування програмного забезпечення, взаємодії між їх елементами, забезпечуючи ясне розуміння та ефективну комунікацію між розробниками та іншими зацікавленими сторонами.

Таблиця 2.1. Ролі команди

Учасник	Ролі
Харченко Віталій Андрійович	Team Leader, Lead Developer
Леут Назарій Віталійович	Developer, QA
Козакевич Наталія Олександрівна	Developer, Director
Буханевич Оксана Андріївна	Developer, Designer

2.1.1 Технічне завдання

Технічне завдання для ігрового проекту "Dark Watcher" відповідає за створення діалогової та торгівельної систем, що у свою чергу включають систему ігрових пригод.

1. Опис проекту

"Dark Watcher" — це комп'ютерна гра у жанрі метроїдванія з елементами екшену та пригод. Гравець виступає містичним героєм, що досліджує темний світ та бореться зі злом. Гра відкриває перед спостерігачем широкий світ, який включає елементи головоломок, збирання предметів та епічних бойових сцен.

2. Основні функціональні вимоги

2.1. Діалогова система

2.1.1. Можливість взаємодії з неперсонажними об'єктами через діалоги

2.1.2. Реалістичне моделювання діалогових сцен та варіативність вибору відповідей гравця

2.1.3. Здатність впливати на сюжетну лінію та розвиток персонажів через діалоги

2.1.4. Система ігрових пригод

2.1.4.1. Наявність основних та побічних ігрових пригод, що розкривають сюжет та світ гри

2.1.4.2. Можливість приймати, виконувати ігрові пригоди

2.1.4.3. Відстеження прогресу ігрових пригод та винагороди за їх успішне виконання

2.1.4.4. Логічні головоломки та завдання, пов'язані з ігровими пригодами

2.2. Торгівельна система

2.2.1. Наявність різних товарів та ресурсів для торгівлі

2.2.2. Механізми збуту та отримання товарів

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				30
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2.3. Місця торгівлі та можливість укладання угод з персонажами

3. Технічні вимоги

3.1. Операційна система: Windows 7

3.2. Процесор: Intel Core 2 Duo E5200

3.3. Оперативна пам'ять: 4 ГБ

3.4. Відеокарта: GeForce 9800GTX+ з 1 ГБ пам'яті

3.5. DirectX: версія 10

3.6. Місце на диску: 2 ГБ доступного місця

4. Завдання для розробника

4.1. Розробка діалогової системи з можливістю взаємодії з неперсонажними об'єктами та неігровими персонажами з подальшим впливом на сюжет та персонажів.

4.2. Реалізація системи ігрових пригод з основними та побічними ігровими пригодами, включаючи логічні головоломки та завдання.

4.3. Розробка торгівельної системи збуту та отримання товарів.

4.4. Тестування та налагодження систем діалогів, ігрових пригод та торгівлі з метою виявлення та усунення помилок.

5. Критерії успішності

5.1. Функціональна діалогова система з реалістичним моделюванням діалогових сцен та варіативністю вибору відповідей гравця.

5.2. Повноцінна система ігрових пригод з основними та побічними ігровими пригодами, логічними головоломками та відстеженням прогресу.

5.3. Розвинута торгівельна система з різноманітністю товарів.

5.4. Гра пройдена тестуванням, помилки усунені.

Це технічне завдання надає вихідну точку для розробки гри "Dark Watcher" та встановлює вимоги до її функціональності, апаратного забезпечення та розробницьких інструментів. Воно дозволяє забезпечити систематизацію, чіткість та розуміння командою розробників поставлених цілей, що є основою успішного втілення проекту.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				31
Змн.	Арк.	№ докум.	Підпис	Дата		

2.1.2 Сюжетна складова

Сюжетна складова є однією з ключових основ будь-якої гри, яка може надати гравцеві цікавий імерсивний досвід. Цей аспект гри визначається сюжетом, історією та наративом, що розгортається під час геймплею. Сюжет може бути простим або складним, лінійним або нелінійним, але його головна мета — залучити гравця до гри, надати йому відчуття присутності в уявному світі і зробити його досвід цікавим та значущим.

Ігрова сюжетна складова може включати такі елементи:

- Персонажі: головні та другорядні персонажі гри, які мають свою індивідуальну особистість, мету та мотивацію. Добре розроблені персонажі додають глибину занурення у атмосферу та емоційну прив'язку до сюжету.

- Світ: уявний світ, в якому розгортається гра. Він може бути фантастичним, реалістичним, науково-фантастичним або історичним, залежно від жанру гри. Світ може мати свою власну історію, культуру, політику та економіку, які впливають на сюжетні події та взаємодію персонажів.

- Події та сюжетні лінії: сюжет гри може містити різні події, конфлікти та завдання, які просувають історію вперед. Він може мати головну сюжетну лінію, яка зосереджена на основних подіях, а також побічні сюжетні лінії, які розширюють світ гри та розкривають додаткову інформацію про персонажів та події.

- Діалоги та візуальні оповіді: важливою частиною сюжету є діалоги між персонажами, які можуть розкривати їхні характери, взаємини та мотивацію. Візуальні оповіді (завчасно встановлені анімаційні послідовності) можуть використовуватися для надання ключових моментів сюжету, які неможливо відтворити в реальному часі геймплею.

- Вибір гравця: деякі ігри надають гравцеві можливість впливати на розвиток сюжету шляхом прийняття рішень. Ці рішення можуть мати вплив на подальші події, розвиток персонажів та кінцевий результат гри. Цей

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				32
Змн.	Арк.	№ докум.	Підпис	Дата		

аспект називається "геймплей зі свободою вибору" і створює унікальний досвід для кожного гравця.

Сюжетна складова гри впливає на загальний досвід гравця, надаючи йому стимул до дослідження світу гри, взаємодії з персонажами та розв'язання сюжетних загадок. Якщо сюжет є цікавим, захоплюючим та добре виконаним, він може зробити гру незабутньою та вражаючою для гравця.

Отримавши загальне уявлення — про сюжетну систему визначаємо, що основний важіль впливу на неї це діалогова система. Щоб структурувати діалогову систему буде використано графи з неозначеною вагою. Також для розширення цієї системи, а саме ігрових пригод, буде використано часово-просторовий обмежений послідовний відрізок подій.

Використання графів у діалоговій системі може значно поліпшити ефективність та якість взаємодії з користувачем. Графи можуть бути використані для моделювання різних сценаріїв діалогу, управління переходами між різними станами та надання системі гнучкості та адаптивності. Це все буде базуватися на теорії графів.

Теорія графів — це галузь математики, що вивчає властивості графів. Граф — це абстрактна структура, що складається з вершин (вузлів) і ребер (зв'язків між вершинами) [6]. Графи використовуються для моделювання різноманітних ситуацій, де об'єкти можна представити як вершини, а взаємодії між ними - як ребра.

Теорія графів досліджує різні аспекти графів, такі як їх властивості, структури, алгоритми та застосування. Вона включає в себе розгляд таких понять, як шляхи, цикли, зв'язність, кольорування, потоки, дерева, сітки та багато інших.

Теорія графів знаходить застосування в багатьох галузях, таких як комп'ютерні науки, логістика, транспортні системи, соціальні мережі, електричні мережі, телекомунікації, біологія, фінанси та багато інших. Вона

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				33
Змн.	Арк.	№ докум.	Підпис	Дата		

надає інструменти для аналізу та розв'язання задач, пов'язаних зі зв'язками між об'єктами.

Узагальнення теорії графів включають орієнтовані графи, ваговані графи, гіперграфи та інші структури, які розширюють базові концепції теорії графів для моделювання більш складних ситуацій.

Графи з неозначеною вагою, також відомі як невизначені графи, є графами, в яких ребра не мають конкретних числових значень або ваг. Замість цього, кожному ребру присвоюється неозначена або символічна вага.

Загалом, графи з неозначеною вагою є важливими для моделювання ситуацій, де точні числові значення ваг недоступні або невідомі, і вимагають спеціальних методів для їх аналізу та розв'язання.

Ось деякі способи використання графів у діалоговій системі:

- Моделювання сценаріїв: граф може представляти сценарій діалогу з різними вузлами, що відповідають окремим крокам або етапам розмови. Кожен вузол може мати певні умови або варіанти відповідей, які визначають, який вузол буде виконаний наступним. Це дозволяє системі керувати ходом діалогу і реагувати на вхідні дані користувача залежно від поточного стану.

- Управління контекстом: граф може використовуватися для збереження та оновлення контексту діалогу. Кожен вузол може містити інформацію про попередні запити та відповіді, що дозволяє системі зберігати стан діалогу та коректно відповідати на подальші запити, враховуючи контекст.

- Розпізнавання намірів та варіантів: кожен вузол може представляти певний намір або можливу відповідь, а ребра графа можуть вказувати на можливі переходи між ними в залежності від вхідних даних користувача.

- Підтримка множинних шляхів та варіацій: граф може включати різні шляхи та варіації для відповідей та реакцій системи. Це дозволяє

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				34
Змн.	Арк.	№ докум.	Підпис	Дата		

системі бути гнучкою та отримувати різні варіанти інформації що може продовжувати чи завершувати діалог в залежності від вхідних даних та контексту.

Графи в діалоговій системі можуть бути представлені у вигляді орієнтованого ациклічного графа (рис 2.1) , дерева або інших варіацій, залежно від вимог та складності системи. Вони можуть бути розроблені та керовані розробниками системи або вивчатися та підтримуватися з використанням машинного навчання та алгоритмів зворотного поширення.

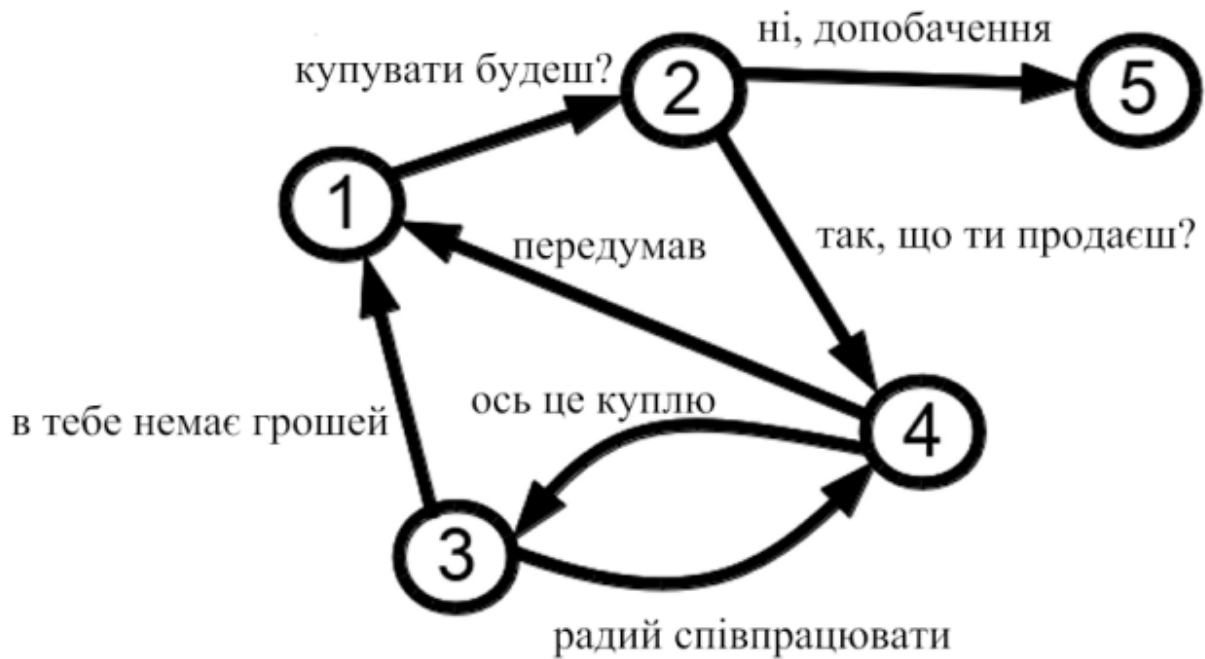


Рисунок 2.1 – Орієнтований ациклічний граф діалогу

Використання графів у діалоговій системі дозволяє забезпечити структуровану та гнучку взаємодію з користувачем, поліпшити розуміння його потреб та намірів і створити більш природний та ефективний діалоговий досвід.

Для розробки схеми ігрових пригод обрано часово-просторовий обмежений послідовний відрізок подій. Він використовується для опису

подій або подієвого ряду, які відбуваються в обмеженому просторі і тривають протягом певного періоду часу [10].

"Часово-просторовий" означає, що цей відрізок подій включає як просторовий, так і часовий аспекти. Він описує, де і коли відбуваються події, звертаючи увагу на їхню просторову розташованість та послідовність у часі.

"Обмежений" означає, що цей відрізок подій має чітко визначені межі або діапазон, в якому він відбувається. Він може бути обмеженим просторово, наприклад, в певному місці або на певній території, або обмеженим в часі, якщо події відбуваються протягом певного періоду.

"Послідовний" означає, що події відбуваються одна за одною в певному порядку. Вони можуть мати логічну послідовність або хронологічний порядок.

Таким чином, часово-просторовий обмежений послідовний відрізок подій, описує певну послідовність подій, які відбуваються в обмеженому просторі і тривають певний час.

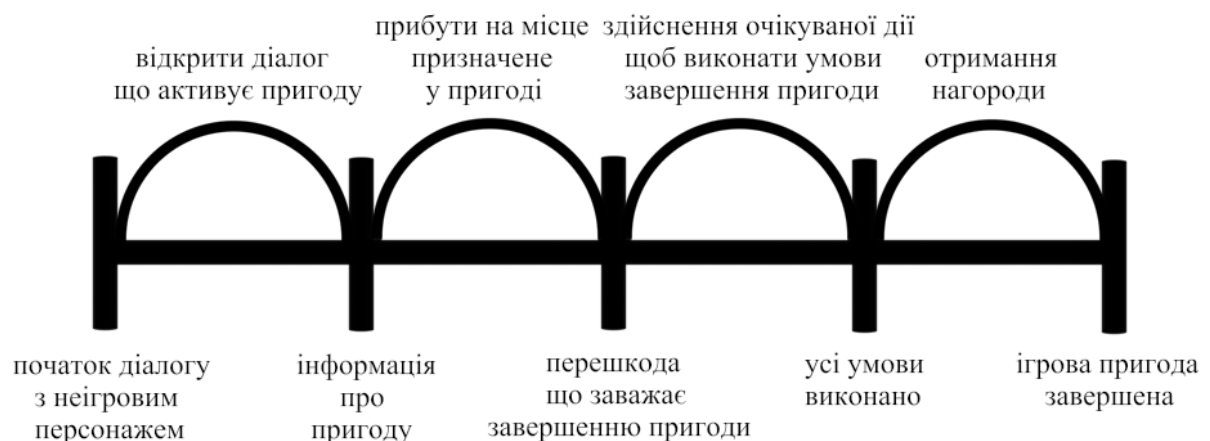


Рисунок 2.2 – Часово-просторовий обмежений послідовний відрізок ігрової пригоди

Підбиваючи підсумки визначаємо що розроблені схеми є актуальними та ефективними для розробки необхідного функціоналу.

2.2 Розробка подій та елементів гри

Під час розробки гри "Dark Watcher" використовувалися дві основні технології: UES та ScriptableObjects. Кожна частина роботи з цими технологіями дає унікальні можливості для розробки і спрощення розуміння шляху системи.

Unity Event System — це система подій, що використовується в Unity для реалізації взаємодії між об'єктами в ігровому світі. Вона дозволяє створювати та відстежувати події, які можуть спрацьовувати в різних контекстах і викликати відповідні дії.

У Unity Event System події визначаються за допомогою класу UnityEvent або унаслідованих від нього класів. Кожна подія може мати декілька підписників, які можуть реагувати на спрацьовування події викликом своїх функцій. Підписники можуть бути задані як публічні методи, що доступні для налаштування в Інспекторі Unity, або за допомогою скриптів на мові програмування C#.

Unity Event System дозволяє реалізувати різноманітні взаємодії, такі як кнопки, меню, анімації, фізичні події та багато інших. Вона спрощує комунікацію між об'єктами і дозволяє зв'язувати функціональність різних компонентів і скриптів без необхідності прямого посилання на них [11].

Unity Event System є потужним інструментом для розробки ігрових систем та інтерфейсів. Вона дозволяє створювати реактивні та інтерактивні елементи управління, що покращують досвід гравців і спрощують роботу розробників.

ScriptableObject — це клас у Unity, який дозволяє створювати розширювані об'єкти даних, які можна використовувати як контейнери для зберігання даних гри [12]. Вони використовуються для створення, збереження та перенесення конфігураційних даних, ресурсів, станів гри та іншої інформації.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				37
Змн.	Арк.	№ докум.	Підпис	Дата		

ScriptableObject є особливим типом об'єкта в Unity, який може бути створений та налаштований у редакторі Unity, а потім використовуватись під час виконання гри. Вони можуть містити будь-які типи даних, такі як числа, рядки, масиви, об'єкти та інші ScriptableObjects.

Основна перевага використання ScriptableObjects полягає в тому, що вони дозволяють зберігати дані в редакторі Unity і використовувати їх безпосередньо під час розробки, а також під час виконання гри. Вони можуть бути легко модифіковані та оновлені, не змінюючи сам код гри.

ScriptableObjects також дозволяють створювати зв'язки між об'єктами та компонентами гри. Вони можуть використовуватись як налаштування для ігрових об'єктів, дозволяючи швидко змінювати їхню функціональність, або як контейнери для спільного використання даних між різними об'єктами.

ScriptableObject є потужним інструментом для організації та управління даними гри в Unity. Використання ScriptableObjects дозволяє створювати модульні та переносні скрипти, що полегшує розробку і забезпечує більшу гнучкість у налаштуванні гри.

2.2.1 Розробка розгалуженої діалогової системи

Ця діалогова система заснована на взаємодії з Node об'єктами які надають можливість розгалуженої взаємодії, логіки до вибору подій та активації ігрових пригод.

Скриптований об'єкт Node:

- вершина графа що дозволяє переключатися між реченнями діалогу за допомогою системи переходів;
- Node може містити інформацію яка буде відображена користувачеві при доступі до варіантів шляху до наступної вершини;
- може містити логіку переходів та функціональності при активації вузла, наприклад при отриманні ігрової пригоди

- може містити умови та передумови, що визначають наступний вузол

Об'єкт Node містить посилання на об'єкт рядку тексту, а також може містити список компонентів та посилання на інші об'єкти вузлового типу.

Об'єкт рядку тексту Line:

- представляє собою рядок тексту який використовується об'єктами вузлового типу для передачі сюжетної частини діалогу

- кожен рядок тексту створюється через редактор для спрощення процесу розробки

- кожен рядок тексту має посилання на доповідача інформації

Об'єкт Line використовується об'єктами Node та містить посилання на доповідача інформації.

Доповідач інформації Actor:

- об'єкт містить ім'я доповідача який знаходиться у активному стані діалогу

- містить зображення завчасно зазначене у редакторі

- усі компоненти використовуються для відображення на діалоговому полотні

Об'єкт Actor використовується об'єктами Line та містить ім'я та зображення активного доповідача інформації.

У даному розділі було представлено опис об'єктів, які використовуються в розгалуженій діалоговій системі. Головною ідеєю системи є взаємодія з Node об'єктами, які забезпечують розгалужену взаємодію, логіку вибору подій та активацію ігрових пригод. Об'єкт Node є вершиною графа, що дозволяє переключатися між реченнями діалогу і містить інформацію, яка відображається користувачеві при доступі до варіантів шляху до наступної вершини. Він також може містити логіку переходів та функціональність при активації вузла.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				39
Змн.	Арк.	№ докум.	Підпис	Дата		

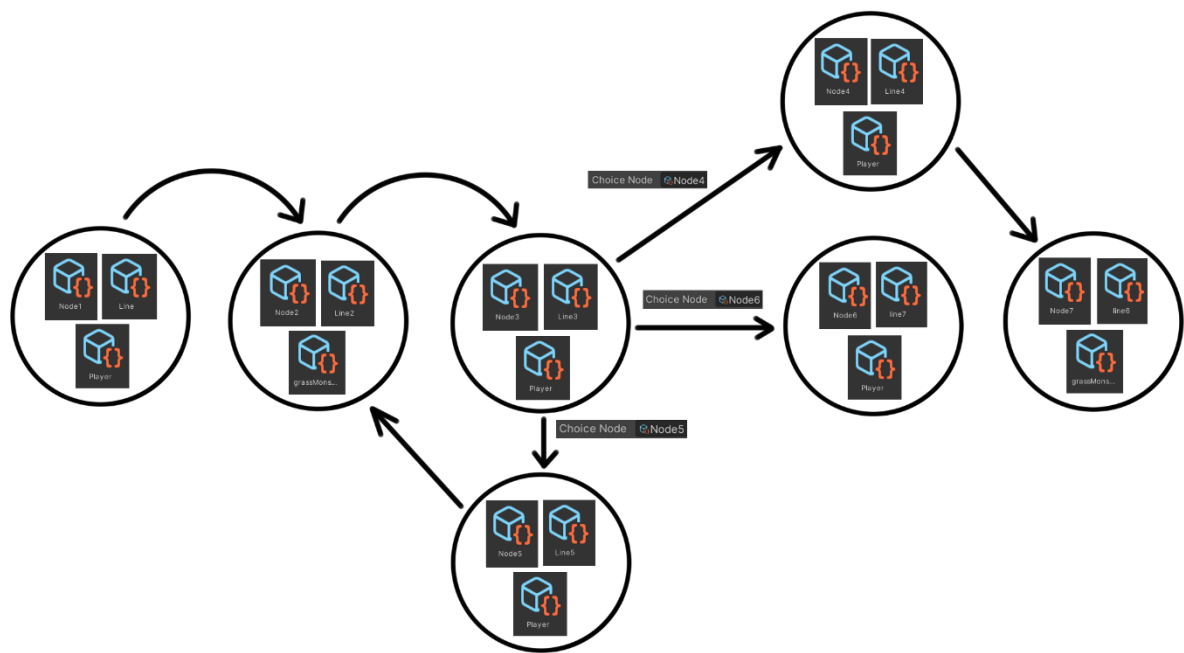


Рисунок 2.3 – Реалізація ігрової взаємодії між ScriptableObjects та UES

Об'єкт Node має посилання на об'єкт рядку тексту Line, який представляє собою сюжетну частину діалогу. Рядки тексту створюються через редактор для спрощення процесу розробки і містять посилання на доповідача інформації. Доповідач інформації, представлений об'єктом Actor, містить ім'я доповідача, зображення і використовується для відображення на діалоговому полотні.

Ці об'єкти створюють основу для розробки розгалуженої діалогової системи, що дозволяє створювати складні та цікаві діалоги з можливістю розгалуження та активації ігрових пригод.

2.2.2 Розробка лінійної діалогової системи

Ця діалогова система заснована на взаємодії з системою вводу та виводу вбудованої в мову програмування C#.

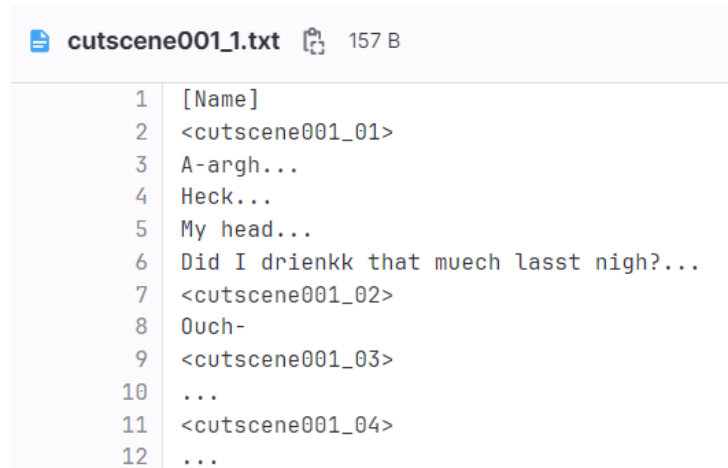
Що включає в себе ця система:

- картинку портрет персонажу
- ім'я персонажу
- репліку персонажу

Взаємодія з текстовим файлом заснована на власному форматуванні, де кожен рядок зчитується, декодується, переформатовується та використовується в залежності від вказаного символу початку рядка.

Символи на початку рядку:

- «[]»: ім'я мовця
- «<>»: портрет мовця
- «>»: репліка мовця



```

1 [Name]
2 <cutscene001_01>
3 A-argh...
4 Heck...
5 My head...
6 Did I drienkk that muech lasst nigh?...
7 <cutscene001_02>
8 Ouch-
9 <cutscene001_03>
10 ...
11 <cutscene001_04>
12 ...

```

Рисунок 2.4 – Приклад застосування власного форматування у діалозі.

Підбиваючи підсумки було представлено опис системи, яка базується на взаємодії з системою вводу та виводу, вбудованою в мову програмування С#. Взаємодія з текстовим файлом в цій системі базується на власному форматуванні, де кожен рядок зчитується, декодується, переформатовується та використовується в залежності від символу на початку рядка. Символи вказують на ім'я, портрет та репліку мовця відповідно.

Ця лінійна діалогова система є простою у реалізації та використанні. Вона може бути використана для створення простих діалогів з одним або декількома персонажами.

Однак важливо зазначити, що така лінійна структура діалогу обмежує гнучкість та можливості розгалуження в діалоговій системі, тому вона використовується лише у візуальних оповідях які не потребують різноманіття варіантів відповідей.

2.3 Проектування та реалізація алгоритмів роботи системи

Критично важливим етапом розробки є побудова ефективних алгоритмів роботи розроблюваної програмної системи. Алгоритми визначають логіку та поведінку яких буде дотримуватись система, вирішують поставлені задачі, виконують обробку даних та гарантують їх оптимальне виконання [13].

Діаграма активності використана для опису взаємодії користувача з неігровими персонажами або Non-Player Character (NPC) та використовується для опису діалогової системи та її функціональних можливостей. Основна мета цієї діаграми активності – опис функціональної системи діалогу, тобто як користувачі звертаються до цієї системи для отримання інформації або іншого результату.

На діаграмі «Взаємодія з неігровими персонажами» зображено інтерактивний процес що призводить до отримання певного результату, а саме ворожого чи дружнього.

Роз'яснення діаграми «Взаємодія з неігровими персонажами»:

- ігровий персонаж входить в радіус взаємодії з неігровими персонажами;
- з'являється можливість співдії;
- після зчитування чи відбулася взаємодія розрізняється якою вона була;
- при дружній взаємодії (розмова, торгівля) відкривається діалогове вікно;
- при ворожій взаємодії (образа, атака) неігровий персонаж переходить у стан ворога;

відповідно до отриманих результатів взаємодії продовжується рух за сценарієм гри.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				42
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.5 – Діаграма активності «Взаємодія з неігровими персонажами»

Діаграма активності для діалогової системи зображає взаємодію гравця з діалогом та його інтерфейсом. Вона дозволяє зрозуміти як проходить ігровий діалог та можливі дії з ним.

Різновид дружньої взаємодії з неігровими персонажами, розмову між ними та ігровим персонажем, зображено на рисунку 2.6.

Роз'яснення діаграми «Діалог з неігровими персонажами»:

- після ініціювання взаємодії перед гравцем розгортається діалогове інформаційне вікно
- при діалозі у певний момент розгортається меню вибору відповідей що вплине на подальшу розмову з неігровим персонажем
- добір відповідей призводить до таких взаємодій як: обмін інформацією, торгівля та прийняття певних завдань
- при виборі специфічної відповіді діалог завершиться

– взаємодія продовжується до вибору специфічної відповіді або примусового виходу з діалогу



Рисунок 2.6 – Діаграма активності «Діалог з неігровими персонажами»

На діаграмі «Діалог з неігровими персонажами» зображено інтерактивний процес дружньої взаємодії з NPC, що призводить до отримання інформації через циклічну дію продовження діалогу, або ж його завершення.

Діаграма активності використана для опису взаємодії користувача з NPC для придбання предметів. Основна мета цієї діаграми активності – опис функціональної системи торгівлі, тобто як користувач звертаються до цієї системи для отримання ігрових предметів.

Роз’яснення діаграми «Торгівельна система»:

1. Користувач ініціює взаємодію з торговельною системою
2. відображається торговельне інтерфейсне вікно, яке містить інформацію про товари

3. Користувач має можливість переглянути асортимент товарів, їх ціни та властивості. Можливо, деякі товари будуть заблоковані або недоступні залежно від умов гри або навичок гравця.
4. Користувач може вибрати товар, який бажає придбати. Для цього він взаємодіє з відповідними елементами інтерфейсу, такими як кнопки
5. У разі успішної операції користувач отримує товар
6. Після завершення операції користувач може продовжити переглядати асортимент товарів і здійснювати інші торговельні операції, або вийти з торговельної системи.

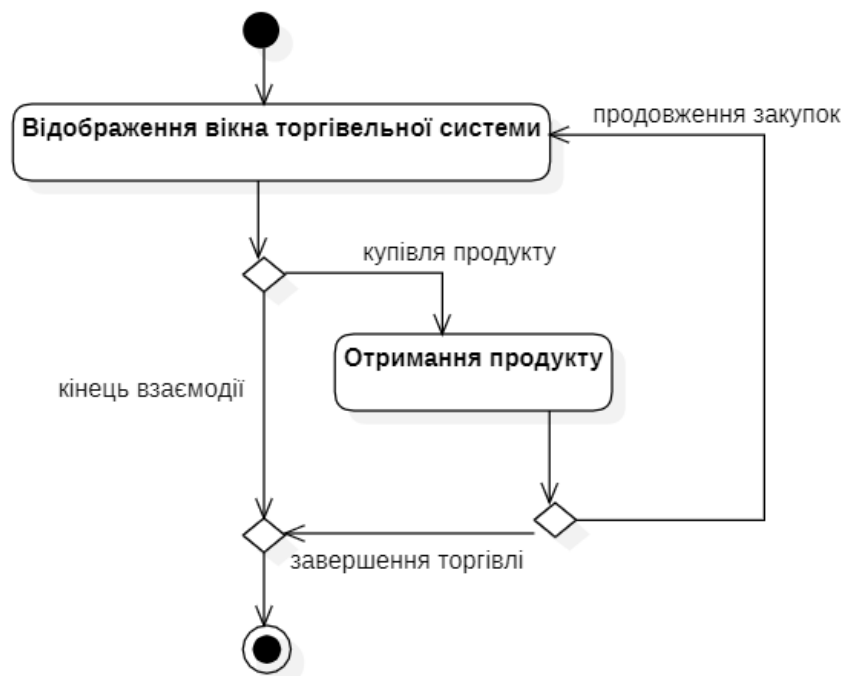


Рисунок 2.7 – Діаграма активності «Торгівельна система»

Торгівельна система надає гравцям доступ до різноманітного асортименту товарів, які можуть поліпшити їх гральний досвід. Торгівельна система може додати глибини до геймплею, оскільки вона створює додаткові механіки та виклики для гравця [14]. Вибір правильних товарів, відкриття рідкісних або унікальних предметів, розумне використання

економічних ресурсів - все це вимагає стратегічного мислення та прийняття рішень, що сприяє поглибленню імерсивності та цікавості гри.

Розглянемо діаграму класів. Діаграма класів — це тип діаграми в уніфікованій мові моделювання (UML), яка показує структуру системи шляхом зображення класів, атрибутів і зв'язків між об'єктами. Діаграма класів використовується для візуалізації структури класів та є одним з основних інструментів в процесі проектування програмного забезпечення і використовується для розробки архітектури програмного продукту.

Діаграми класів зазвичай використовуються в розробці програмного забезпечення для представлення структури системи та проектування нових систем. На діаграмі класів є три основні типи зв'язків, які можна зобразити між класами: асоціація, агрегація та успадкування.

Асоціація: асоціація — це зв'язок між двома класами, де один клас використовує інший. Це представлено на діаграмі класів лінією з відкритою стрілкою, що вказує від класу, який використовує інший клас, до класу, який використовується. Асоціація може бути односторонньою (односпрямованою) або двосторонньою (двонаправленою). В односторонній асоціації тільки один клас використовує інший клас, тоді як у двосторонній асоціації обидва класи використовують один одного.

Агрегація: агрегація – це зв'язок між двома класами, де один клас є частиною іншого. Це представлено на діаграмі класів лінією з ромбовидною формою на кінці, яка вказує від усього класу до частини класу. Агрегація — це спосіб представлення зв'язку між класами, де весь клас має частину класу як атрибут. Клас частини може існувати незалежно від цілого класу і може використовуватися кількома цілими класами.

Успадкування: успадкування – це зв'язок між класами, де один клас (підклас) успадковує характеристики від іншого класу (суперкласу). Це представлено на діаграмі класів лінією із замкнутою стрілкою, що вказує від підкласу до суперкласу. Спадкування — це спосіб створення нового класу,

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				46
Змн.	Арк.	№ докум.	Підпис	Дата		

який є модифікованою версією існуючого класу. Підклас успадковує всі атрибути та методи суперкласу, а також може мати власні додаткові атрибути та методи.

Існують також інші типи відносин, які можуть бути зображені на діаграмі класів, наприклад залежність і реалізація. Залежність — це зв'язок між двома класами, де один клас залежить від іншого для своєї реалізації. Реалізація — це зв'язок між класом та інтерфейсом, де клас реалізує інтерфейс.

Орієнтована діаграма класів – це інструмент який використовується в процесі розробки програмного продукту. Вона надає розробникам та команді розробки важливу інформацію про структуру системи, взаємозв'язки між класами, атрибути та методи кожного класу. Це дозволяє розробникам глибше зрозуміти логіку програми та спроектувати ефективну архітектуру.

Ця діаграма класів також дозволяє розробникам легше виявляти потенційні проблеми та здійснювати необхідні зміни протягом розробки. Якщо виникають нові вимоги або змінюються вимоги до програмного продукту, можна легко модифікувати структуру класів відповідно до нових вимог.

Крім того, орієнтована діаграма класів є основою для подальшого програмування, документування та спілкування між учасниками команди розробки. Вона надає зручний спосіб візуалізації системи, що полегшує спілкування та розуміння між розробниками. Діаграма може використовуватись для пояснення структури системи замовникам або іншим зацікавленим сторонам.

Загалом, орієнтована діаграма класів є вкрай корисним інструментом для розробників програмного продукту [15]. Вона допомагає зрозуміти структуру системи, виявити потенційні проблеми та зміни вимог, а також полегшує спілкування та розуміння між учасниками команди розробки.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				47
Змн.	Арк.	№ докум.	Підпис	Дата		

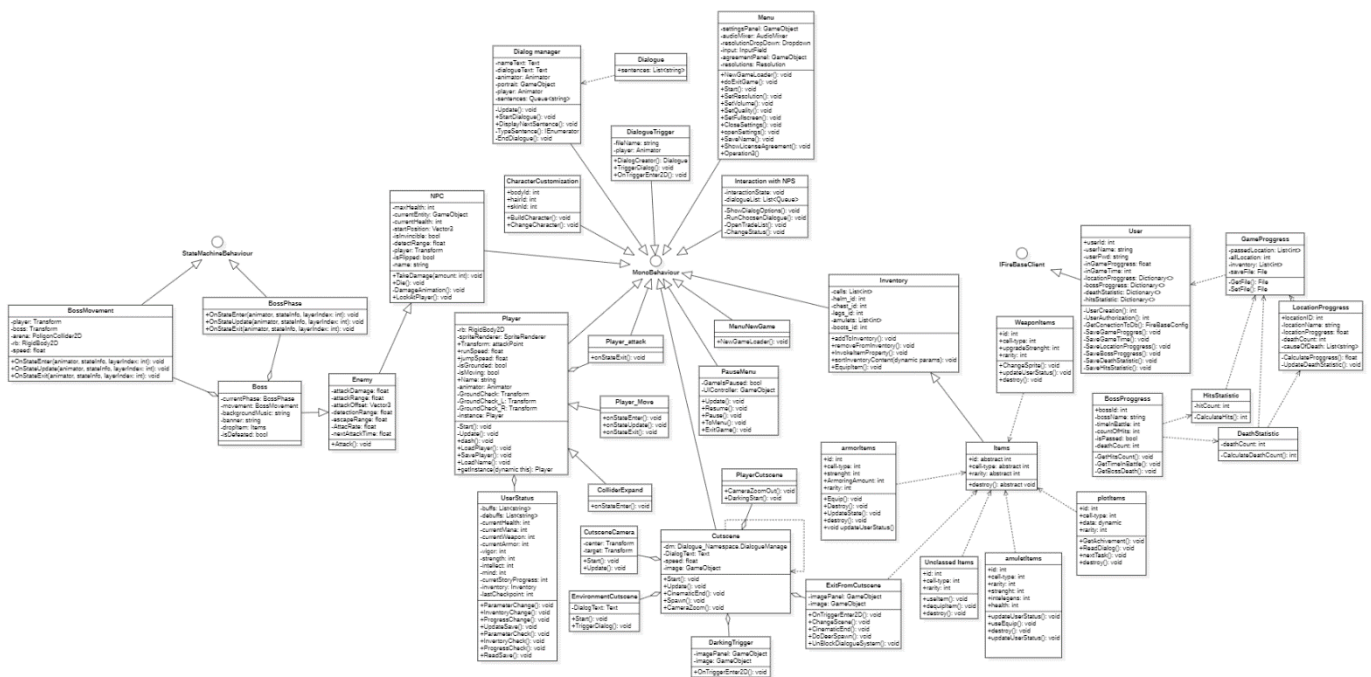


Рисунок 2.8 – Загальна командна діаграма класів

Діаграма послідовності — це тип діаграми, який показує взаємодію між об'єктами або компонентами протягом певного періоду часу. Він використовується для моделювання поведінки системи або для опису конкретної взаємодії між об'єктами в системі.

На діаграмі послідовності об'єкти представлені горизонтальними лініями, а взаємодії між об'єктами – вертикальними лініями, які називаються «лініями життя». Взаємодія між об'єктами відображається як повідомлення, які передаються між об'єктами. Повідомлення можуть бути синхронними (це означає, що відправник повідомлення чекає відповіді, перш ніж продовжити), або асинхронними (це означає, що відправник повідомлення не чекає відповіді, перш ніж продовжити).

Діаграми послідовності зазвичай використовуються для моделювання взаємодії між об'єктами в системі в контексті конкретного сценарію або випадку використання. Їх можна використовувати для моделювання взаємодії між об'єктами в системі на різних рівнях абстракції, від оглядів високого рівня до детальних описів низького рівня. Діаграма послідовності діалогової системи наведена на рисунку 2.9.

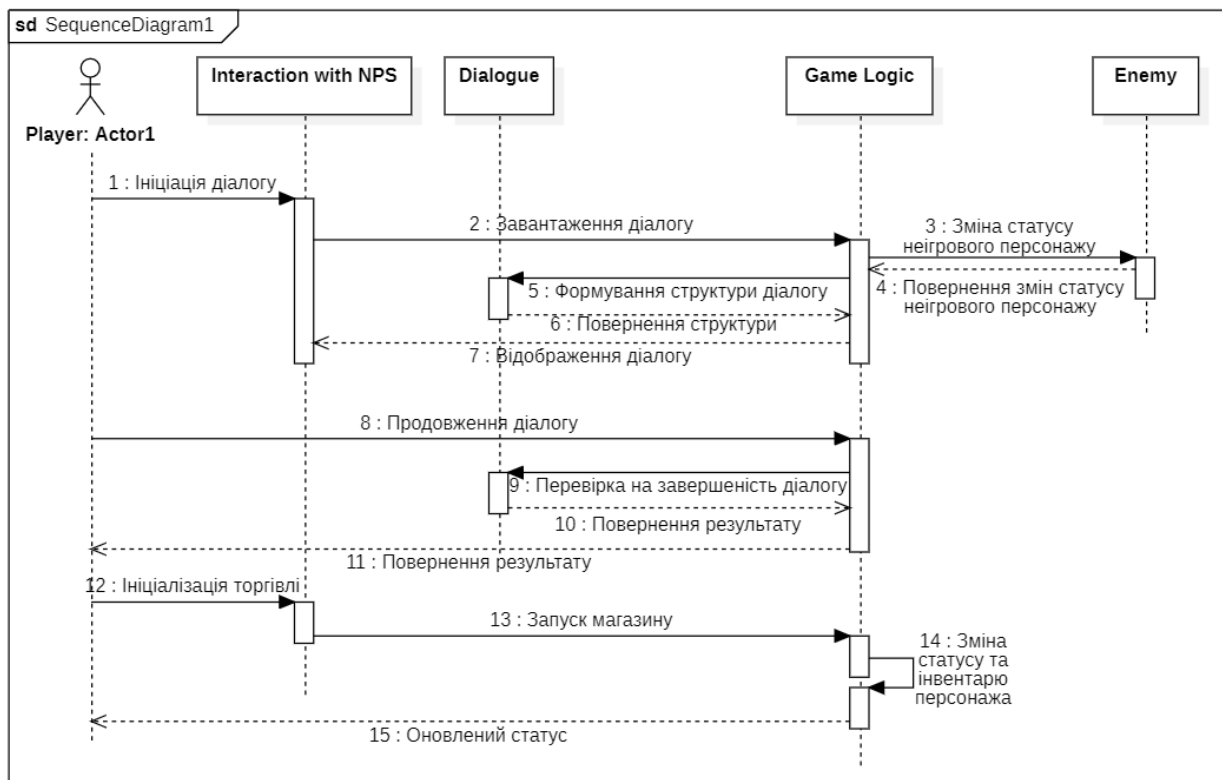


Рисунок 2.9 – Діаграма послідовності

Діаграми компонентів дозволяють визначити архітектуру системи, що розробляється. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси та залежності між ними.

Компонент:

Компонент (component) – термін який застосовується у UML для представлення фізичних об'єктів.

Компонент призначений для представлення фізичної організації елементів моделі, з якими він пов'язаний. Також компонент може мати текстовий стереотип і тегове значення, а деякі компоненти — власні графічні зображення.

Компонент може реалізувати певний набір інтерфейсів. Прямокутник і два менші прямокутники зліва використовуються для графічного представлення компонента. Ім'я компонента та додаткова інформація записуються усередині.

Інтерфейси:

Графічно вони зображуються колами, з'єднаними з компонентами лініями без стрілок, а коло пишеться назва інтерфейсу, яке рекомендовано писати з великої літери «I».

Крім того, інтерфейс на діаграмі компонентів може бути представлений у вигляді прямокутника класу зі стереотипом <<interface>> та розділом для операцій, що підтримуються. Як правило, ця нотація використовується для представлення внутрішньої структури інтерфейсів.

Під час розробки програмних систем інтерфейси забезпечують як сумісність між різними версіями, а й можливість вносити істотні зміни до однієї частини програми, залишаючи інші частини без змін. Характер використання інтерфейсу окремими компонентами може бути різним.

Існує два способи підключення інтерфейсів та компонентів. Якщо компонент реалізує інтерфейс, цей компонент надає його послугу іншим компонентам.

Залежності між компонентами:

Залежно від діаграм компонентів представлені пунктирними лініями зі стрілками, що вказують від залежного об'єкта до незалежного об'єкта.

Залежно можуть представляти файлові відносини вашої програми під час компіляції. Залежності можуть також вказувати на існування в незалежних компонентах описів класів, що використовуються в залежних компонентах.

На діаграмі компонентів залежності дозволяють зв'язати разом компоненти та інтерфейси, які використовуються конкретними компонентами та компонентами різних типів. У цьому випадку ви малюєте стрілку від компонента клієнта до вашого інтерфейсу. Така стрілка означає, що компонент не реалізує інтерфейс, а використовує під час виконання. При цьому на тій самій діаграмі може існувати інший компонент, що реалізує цей інтерфейс. Відносини реалізації інтерфейсу показані на діаграмах компонентів звичайними лініями без стрілок.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Побудовану діаграму основних компонентів проектованої системи можна побачити на рисунку 2.10.

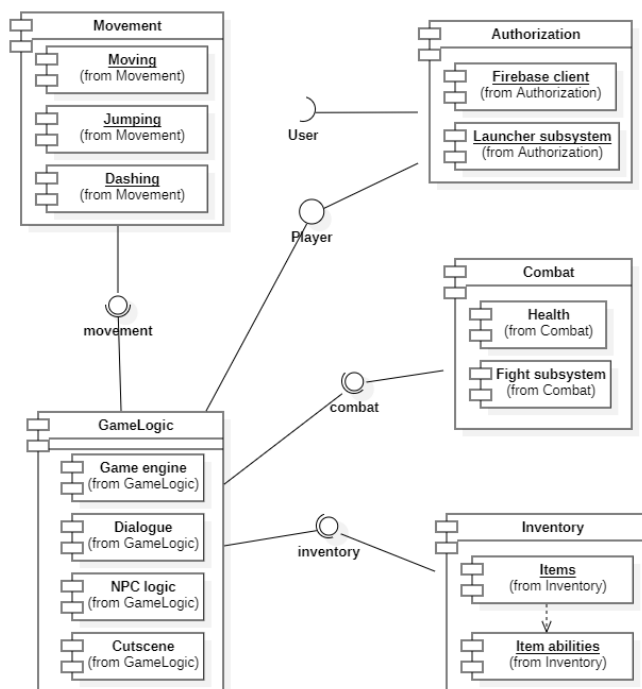


Рисунок 2.10 – Діаграма компонентів

Діаграми розгортання створюються в UML, щоб показати конфігурацію та топологію програмної системи. По суті створення діаграми розгортання — це останній крок у визначенні моделі програмної системи.

Діаграми розгортання використовуються для представлення елементів та компонентів програми, які існують лише на етапі виконання (runtime).

Діаграма розгортання містить графічне уявлення пристроїв, процесів та взаємозв'язків між ними. Створення схеми розгортання супроводжується наступними цілями:

- Розподілити компоненти системи по відповідних фізичних вузлах;
- Зобразити фізичні зв'язки між всіма вузлами системи на етапі виконання;
- Пере-конфігурувати топологію для досягнення вищої продуктивності, якщо це потрібно.

Основним елементом діаграми є вузол. Вузол — це фізично присутній елемент системи з певним обчислювальним ресурсом. Останні версії мови UML розширюють концепцію вузлів, включаючи не тільки обчислювальні пристрої, але й інші механічні або електронні пристрої, такі як принтери, датчики, цифрові камери, модеми, сканери і т.д.

Діаграма розгортання для модельованого програмного комплексу зображено на рисунку 2.11.

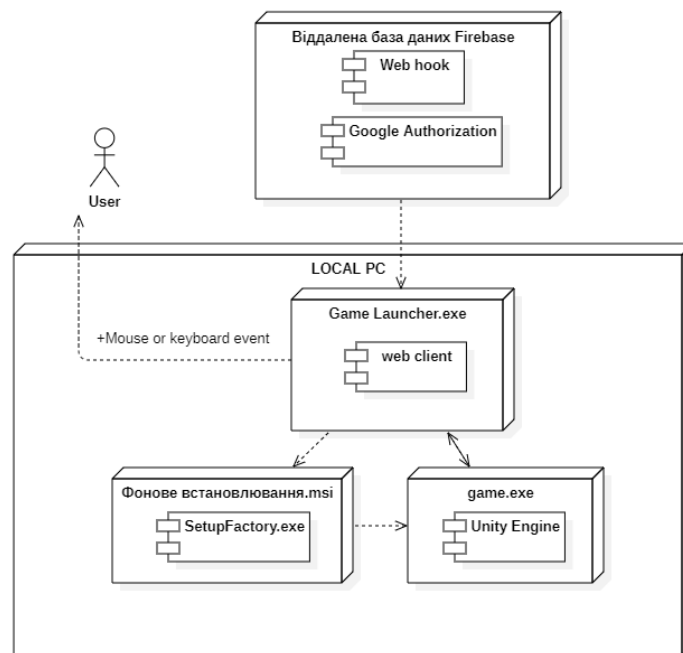


Рисунок 2.11 – Діаграма розгортання

Необхідно зазначити що гра працює за правилом SEC (System-Entity-Component).

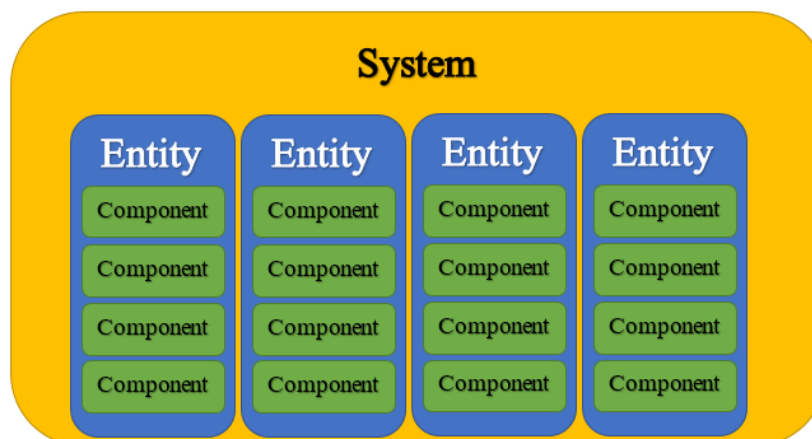


Рисунок 2.12 – SEC-паттерн проектування

2.4 Реалізація ігрового додатку

Розробка гри — це творчий процес створення відеогри від самої ідеї до готового продукту. Цей захоплюючий процес включає в себе кілька етапів, що поступово розкривають потенціал гри та надають їй життя.

На початку стоїть концепція гри. Це момент, коли розробники визначають загальну ідею гри та її основні характеристики. Вирішується, який жанр гри буде використовуватись, які механіки вона буде мати, а також які цілі і завдання ставляться перед гравцем.

Після концепції розпочинається етап проектування. Тут розробники створюють детальні плани і концепції гри. Вони проектують ігрові механіки, системи, рівні, персонажів, інтерфейс користувача та інші аспекти гри. Часто цей етап включає створення документації, схем, макетів та інших матеріалів для подальшої розробки.

Потім настає фаза розробки, коли починається створення самої гри. Це включає роботу з графікою, звуковими ефектами, програмним кодом та іншими компонентами гри. Розробники використовують різні інструменти і програмні платформи для створення візуальних та функціональних елементів гри.

Після завершення розробки гра проходить в етап тестування. Тестувальники перевіряють гру на наявність помилок, недоліків, проблем з ефективністю та іншими проблемами.

За умов успішного проходження тестування гра готова до випуску. Вона може бути поширена через різні канали, такі як цифрові магазини додатків або фізичні носії. Після випуску розробники можуть продовжувати підтримувати гру, випускаючи патчі, оновлення та додатковий контент, що розширює геймплей і забезпечує задоволення гравцям.

Ключовим способом отримання ігрового досвіду та занурення у фантастичний світ, є взаємодія гравця з навколишнім середовищем всередині гри. Системи діалогу та торгівлі доповнюють історію гри та дають

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				53
Змн.	Арк.	№ докум.	Підпис	Дата		

можливість гравцям насолоджуватися багатогранністю та різноманіттям геймплею.

В першу чергу, вони сприяють збільшенню імерсії гравця. Взаємодія з неігровими персонажами через систему діалогу дозволяє створити віртуальний світ, в якому гравець відчуває себе частиною історії. Завдяки ігровим пригодам гравець може виконувати різні місії та завдання, що розвивають сюжет та надають відчуття прогресу. Торгівля дозволяє гравцеві обмінюватися ресурсами, предметами та отримувати нові можливості. Це допомагає гравцеві відчути глибину гри та створює більш насичений геймплей.

Друга причина полягає в розвитку сюжету гри. Системи діалогу та ігрових пригод відкривають нові шляхи для розкриття сюжету, взаємодії з персонажами та розкриття секретів гри. Вони створюють можливість отримати важливу інформацію, вирішити загадки та розкрити додатковий контент. Це підтримує цікавість гравця та мотивує його досліджувати гру.

Крім того, системи діалогу, ігрових пригод та торгівлі надають гнучкість геймплею. Гравцю надається можливість обирати свій власний шлях та варіанти вирішення проблем. Це дозволяє гравцю самостійно впливати на розвиток гри, вибирати різні рольові характеристики та навички. Рішення, прийняті в системі діалогу або ігровій пригоді, можуть мати вплив на подальший сюжет гри, що робить геймплей більш особистим та унікальним для кожного гравця.

Використання потоків у діалоговій системі надає можливість визначити початок або завершення взаємодії з діалогом, переключати стани гравця наприклад «у діалозі» та «в гри» і в залежності від стану запускати наступний скрипт.

Таблиця 2.2. Основні методи класу FlowChannel

Назва методу	Відповідальність методу
RaiseFlowStateRequest(FlowState state)	Реакція на зміну стану у потоці ігрових подій через UES

RaiseFlowStateChanged(FlowState state)	Дозволяє іншим об'єктам підписуватися на події зміни стану потоку, реагувати на них та взаємодіяти з ними в контексті ігрового середовища. Використання цього класу дозволяє зручно керувати і спілкуватися між різними компонентами та системами у вашій відеогрі.
--	---

Коректне відображення діалогового вікна, є важливим аспектом візуального сприйняття, тому відслідковування зміни станів – відповідає за перевірку чи вже було запущено діалог.

Таблиця 2.3. Основні методи класу FlowListener

Назва методу	Відповідальність методу
Awake()	Отримує завдання слідкування за станами, ініціалізауючи об'єкт
OnDestroy()	Припинення слідкування за станами, знищення об'єкту
OnFlowStateChanged (FlowState state)	Перевірка стану з можливістю запуску діалогу

Для коректного відображення діалогів, а саме запобіганню небажаного або випадкового запуску діалогу створена машина логічних станів, яка тримає в собі поточний стан, вона змінює, встановлює та зберігає поточні стани. На цей клас посилаються всі класи які взаємодіють зі станами діалогу або гри.

Таблиця 2.4. Основні методи класу FlowStateMachine

Назва методу	Відповідальність методу
CurrentStateSetter	Встановлення поточного стану
InstanceSetter	Встановлення поточного екземпляру
Awake()	При створенні компоненту відбувається запит поточного стану та встановлення його як поточного

Start()	При запуску встановлюємо поточний стан як стан запуску
OnDestroy()	При знищенні об'єкту видаляємо стани та встановлюємо поточний екземпляр як відсутній
SetFlowState (FlowState state)	Встановлення стану в залежності від переданого параметру

Наведений клас CharacterMovement додає додаткову систему руху для діалогів та катсцен. Він вимикає можливість руху в залежності від потокового стану, при відображенні діалогів або катсцен.

Таблиця 2.5. Основні методи класу CharacterMovement

Назва методу	Відповідальність методу
Start()	При запуску отримуємо зону управління персонажем
Update()	Управління додатковими неігровими персонажами під час діалогу

Важливо визначити чи присутня можливість взаємодії з неігровими об'єктами. У випадку відсутності у даного об'єкту діалогу, компонент з цим класом автоматично вимикається. Інакше, якщо об'єкт має діалог, або ігрову пригоду, то на полотні відобразиться підказка для запуску діалогу.

Таблиця 2.6. Основні методи класу Interactable

Назва методу	Відповідальність методу
DoInteraction()	Виклик стану взаємодії
OnTriggerEnter2D()	Виклик діалогу при зіткненні з персонажем та перевірка на наявність ігрової пригоди, що видає цей персонаж

Система здатна перевіряти, чи персонаж перебуває в зоні активного діалогу. Коли натискається кнопка, система автоматично встановлює відповідний стан та розпочинає діалог. Протягом цього процесу, наступна

репліка блокується, і персонаж не може продовжувати діалог доти, поки не зробить свій вибір або не надасть відповідь.

Таблиця 2.7. Основні методи класу InteractionInstigator

Назва методу	Відповідальність методу
HasNearbyInteractables()	Зчитує чи персонаж знаходиться в зоні активації діалогу
Update()	Перевіряє чи виконано дію для активації діалогу при знаходженні у відповідній зоні
OnTriggerEnter()	Активация напису при вході в зону активації діалогу з конкретним персонажем і додавання до списку доступних діалогів
OnTriggerExit()	Деактивация напису при виході з зони активації діалогу з конкретним персонажем і видалення зі списку доступних діалогів

За допомогою цього скриптованого об'єкта можна визначити різні характеристики мовців, такі як їх портрет та ім'я. Це створює широкі можливості для створення багатогранних інтерактивних сценаріїв.

Таблиця 2.8. Основні методи класу NarrationCharacter

Назва методу	Відповідальність методу
CharacterNameSetter	Встановлення імені мовця
SpriteSetter	Встановлення аватара мовця

Створюється скриптований об'єкт, розроблений з метою автоматизації процесу створення діалогового контенту. Він забезпечує можливість задавати наповнення діалогу. Такий скриптований об'єкт є потужним інструментом підвищення рівня імерсії та взаємодії з віртуальним світом.

Таблиця 2.9. Основні методи класу NarrationLine

Назва методу	Відповідальність методу
SpeakerSetter	Встановлення мовця
TextSetter	Встановлення тексту діалогу

Заради підтримки зв'язку між вузлами діалогу, було створено канал діалогу. Його функціональність забезпечує безперервну комунікацію між

вузлами, щоб уникнути втрати зв'язку. Цей канал слугує метою збереження потоку інформації та підтримки неперервної взаємодії між учасниками діалогу. Він дозволяє передавати повідомлення, сигнали або дані між вузлами, забезпечуючи відповідний обмін інформацією. Такий канал діалогу є необхідним компонентом для забезпечення ефективного та безперебійного ходу діалогічного процесу та підтримки його неперервності.

Таблиця 2.10. Основні методи класу DialogueChannel

Назва методу	Відповідальність методу
RaiseRequestDialogue(Dialogue dialogue)	Виконує запит на отримання діалогу з зашифрованого об'єкта
RaiseDialogueStart(Dialogue dialogue)	Викликає, зчитує і відтворює репліку з першого вузла діалогу
RaiseDialogueEnd(Dialogue dialogue)	Вихід з фінального вузла діалогу
RaiseRequestDialogueNode(DialogueNode node)	Отримання наступного вузла діалогу
RaiseDialogueNodeStart(DialogueNode node)	Запуск рядку діалогу з поточного вузла діалогу
RaiseDialogueNodeEnd(DialogueNode node)	Перехід від поточного вузла діалогу до наступного

При старті діалогу відбувається присвоєння поточного діалогу до діалогового каналу. Це включає прикріплення слухачів подій до всього, що відбувається в цьому діалозі, а також розпізнавання іншими класами, що вони знаходяться в контексті діалогу. Це забезпечує можливість сприйняти та реагувати на всі елементи, що відбуваються в діалозі. Шляхом використання діалогового каналу, різні компоненти можуть ефективно спілкуватись між собою та взаємодіяти з контекстом діалогу, забезпечуючи гармонійний хід комунікації.

Таблиця 2.11. Основні методи класу DialogueInstigator

Назва методу	Відповідальність методу
Awake()	Присвоєння подій для різних станів діалогу
OnDestroy()	Видалення подій для різних станів діалогу

OnDialogueStart(Dialogue dialogue)	Запуск конкретного діалогу у об'єкті DialogueChannel
OnDialogueEnd(Dialogue dialogue)	Вилучення конкретного діалогу у об'єкті DialogueChannel

Створюється об'єкт діалогу, який передається в діалоговий канал з метою відображення вузлів та рядків діалогу. Цей об'єкт, виступаючи як клас контейнера, забезпечує зручний збір та організацію всіх елементів, пов'язаних з діалогом. Він призначений для збереження та управління даними, які включають в себе вузли, які представляють кроки чи етапи діалогу, а також рядки діалогу, які відображають текстові повідомлення. Цей об'єкт діалогу дозволяє зручно структурувати та керувати динамікою діалогового процесу, забезпечуючи зручну маніпуляцію та відображення всіх необхідних елементів діалогу в межах діалогового каналу.

Таблиця 2.12. Основні методи класу Dialogue

Назва методу	Відповідальність методу
FirstNodeSetter	Встановлення початкового вузла діалогу

Створюється клас контролера мовців, призначений для керування взаємодією та поведінкою мовців. Цей клас відповідає за надання можливості управляти різними аспектами мовців. Клас контролера мовців дозволяє програмувати їх реакції на події та взаємодію з іншими об'єктами. Клас контролера мовців є важливим елементом для забезпечення ефективного та гнучкого управління мовцями та створення живих та автентичних діалогових сценаріїв.

Таблиця 2.13. Основні методи інтерфейсу DialogueNodeVisitor

Назва методу	Відповідальність методу
Visit(BasicDialogueNode node)	Встановлення актору з вузла без вибору варіантів відповідей
Visit(ChoiceDialogueNode node)	Встановлення актору з вузла з вибором варіантів відповідей

Створюється вузол діалогу, який виконує ряд функцій. Він встановлює наступний вузол, який слідує за поточним, та забезпечує можливість переходу до цього наступного вузла. При цьому вузол перевіряє, чи не є він останнім у послідовності вузлів діалогу. У разі, якщо вузол не є останнім, відбувається перехід до наступного вузла та його активація. Таким чином, вузол діалогу забезпечує правильну послідовність вузлів та управління проходженням по ним. Це важливий механізм для керування ходом діалогового процесу та активації відповідних етапів чи кроків.

Таблиця 2.14. Основні методи класу BasicDialogueNode

Назва методу	Відповідальність методу
NextNodeSetter	Встановлює наступний вузол до поточного
CanBeFollowedByNode(DialogueNode node)	Встановлює можливість переходу до наступного вузла і перевіряє чи не є він останнім
Accept(DialogueNodeVisitor visitor)	Перехід до наступного вузла та його активація

Створюється об'єкт контейнера, призначений для вибору, що включає в себе відображення однієї кнопки. Цей об'єкт контейнера забезпечує можливість представлення користувачу варіантів вибору шляхом відображення кнопки. Він створює зручну візуальну інтерфейсну одиницю, яка дозволяє користувачеві взаємодіяти з системою та вибрати певну опцію, натискаючи на кнопку. Об'єкт контейнера для вибору є елементом інтерфейсу, який забезпечує інтуїтивно зрозумілу та легку навігацію для користувачів, спрощуючи процес прийняття рішень та вибору відповідних дій.

Таблиця 2.15. Основні методи класу DialogueChoice

Назва методу	Відповідальність методу
ChoicePreviewSetter	Встановлення тексту який буде відображений при всередині кнопки вибору

ChoiceNodeSetter	Встановлення наступного вузла який буде слідувати після натискання на кнопку з ChoicePreviewSetter
------------------	--

Створюється об'єкт контейнера, що забезпечує можливість відображення та управління багатьма кнопками одночасно. Він надає можливість представлення користувачу різних варіантів або опцій шляхом відображення кількох кнопок, з якими можна взаємодіяти. Такий об'єкт контейнера дозволяє легко навігувати та взаємодіяти з системою шляхом вибору конкретної кнопки, що відповідає певному варіанту або функції. Він створює зручну та ефективну інтерфейсну одиницю, яка спрощує процес вибору та дозволяє користувачеві швидко та зручно здійснювати вибір серед доступних опцій.

Таблиця 2.16. Основні методи класу ChoiceDialogueNode

Назва методу	Відповідальність методу
CanBeFollowedByNode(DialogueNode node)	Відповідає за перевірку можливості переходу до наступного вузла
Accept(DialogueNodeVisitor visitor)	Перехід до наступного вузла та його активація

Створюється абстрактний клас-батьківський для вузлів, який визначає загальні властивості та методи. Він надає основу для групування та обробки вузлів. Абстрактний клас виступає як шаблон, від якого можна отримати конкретні класи вузлів, які реалізують специфічні функціональність та додаткові властивості. Це забезпечує єдність та структурованість коду, а також полегшує розширення та зміни функціональності шляхом перевизначення методів у похідних класах.

Таблиця 2.17. Основні методи класу DialogueNode

Назва методу	Відповідальність методу
DialogueLineSetter	Встановлює рядок діалогу, який буде відображено у діалоговому вікні
Accept(DialogueNodeVisitor visitor)	Абстрактний метод що успадковується від дочірніх класів

CanBeFollowedByNode(DialogueNode node)	Абстрактний метод що успадковується від дочірніх класів
--	---

Створюється контролер порядку вузлів, який відповідає за керування послідовності вузлів діалогу. Він забезпечує правильну послідовність вузлів та управління проходженням по ним, враховуючи логіку та умови, встановлені в діалоговому сценарії. Він здійснює необхідні переключення між вузлами, враховуючи умови та результати попередніх дій користувача або системи. Таким чином, контролер порядку вузлів забезпечує ефективне та правильне керування послідовністю вузлів діалогу для досягнення потрібного розгортання та розвитку діалогового процесу.

Таблиця 2.18. Основні методи класу DialogueSequencer

Назва методу	Відповідальність методу
StartDialogue(Dialogue dialogue)	Початок порядку діалогу
EndDialogue(Dialogue dialogue)	Завершення діалогу
CanStartNode(DialogueNode node)	Перевіряє чи існує наступний вузол діалогу
StartDialogueNode(DialogueNode node)	Встановлює початок діалогового вузла для поточної послідовності
StopDialogueNode(DialogueNode node)	Очищає діалогові вузли з пам'яті

Створюється механізм для виведення тексту, який дозволяє відображати повідомлення або інформацію на екрані. Цей механізм забезпечує можливість виведення текстового контенту безособовим способом, не прив'язуючись до конкретного виконавця.

Таблиця 2.19. Основні методи класу UIInteractionTextController

Назва методу	Відповідальність методу
Update()	Виведення тексту

Після натискання кнопки виконуються певні дії для обробки події. При натисканні кнопки активуються відповідні процеси, які забезпечують виконання необхідних дій, які заповнюють всі необхідні поля відповідного класу, забезпечуючи правильну обробку та збереження введеної інформації. Крім того, спрацьовує подія, пов'язана з натисканням кнопки, що дозволяє

сповістити інші частини системи або класи про виникнення події та викликати відповідну реакцію.

Таблиця 2.20. Основні методи класу UIDialogueChoiceController

Назва методу	Відповідальність методу
Start()	Заповнює всі необхідні поля класу
OnClick()	Викликає подію при натисканні кнопки

Створюється контролер візуального інтерфейсу діалогу, який відповідає за відображення текстового контенту. Цей контролер забезпечує можливість відображення тексту безособовим способом, незалежно від конкретного виконавця. Він виступає посередником між діалоговими об'єктами та візуальним інтерфейсом, забезпечуючи відображення текстової інформації на екрані.

Таблиця 2.21. Основні методи класу UIDialogueTextBoxController

Назва методу	Відповідальність методу
Awake()	Ініціалізація візуального інтерфейсу
OnDestroy()	Очищення оперативної пам'яті
Update()	При натисканні кнопки запускається наступний вузол діалогу
OnDialogueNodeStart(DialogueNode node)	Вмикається поле тексту та заповнюється текстом
OnDialogueNodeEnd(DialogueNode node)	Очищаються поля діалогу та вимикається діалоговий інтерфейс
Visit(BasicDialogueNode node)	Встановлення очікування натискання наступної клавіші
Visit(ChoiceDialogueNode node)	Встановлення очікування натискання опції

Реалізується читання всього діалогу з текстового файлу, забезпечуючи доступ до всіх його рядків. Запрограмовано виведення наступного рядка діалогу після відповідного виклику. Реалізується можливість закриття діалогу за допомогою відповідної функціональності. Крім того, розробляється механізм анімаційного виведення тексту, що дозволяє ефектно відображати його на екрані. Усі ці функції розроблені з

метою забезпечення зручної та ефективної взаємодії з діалоговою інформацією, забезпечуючи користувачам зручний та зрозумілий спосіб сприйняття текстового контенту.

Таблиця 2.22. Основні методи класу DialogueManager

Назва методу	Відповідальність методу
Update()	Очікування кнопки відображення інформації з діалогового файлу
StartDialogue(Dialogue dialogue)	Читання з текстового файлу усього діалогу
DisplayNextSentence()	Вивід наступного рядка діалогу
TypeSentence(string sentence)	Анімаційний виводу тексту
EndDialogue()	Закриття діалогу

Розроблений функціонал забезпечує ефективне та контрольоване виконання діалогових процесів. Він забезпечує правильну послідовність та взаємодію з об'єктами діалогів. Механізм очікування зіткнення з об'єктом сигналізує про готовність до запуску діалогу. Процес знаходить менеджера та передає йому чергу діалогів. З'являється можливість запуску наступного діалогового процесу при зіткненні з відповідним об'єктом.

Таблиця 2.23. Основні методи класу DialogueTrigger

Назва методу	Відповідальність методу
DialogCreator()	Створення черги для діалогів
TriggerDialog()	Знаходження менеджера та передача йому черги
OnTriggerEnter2D(Collider2D collider)	Очікування зіткнення з об'єктом для запуску діалогу

Визначається, чи було досягнуто цілей ігрової пригоди. Перевіряється факт досягнення її цільових показників. Виконується оцінка результатів та порівняння їх з очікуваними цілями поточної пригоди для встановлення ступеня досягнення цілей.

Таблиця 2.24. Основні методи класу Quest

Назва методу	Відповідальність методу
IsReached()	Перевірка на досягнення цілей ігрової пригоди

Реалізовано функціонал для відображення сповіщення про отримання нової ігрової пригоди. Передбачається механізм, що автоматично виводить на екран повідомлення про наявність нової ігрової пригоди. Розробляється система, яка здатна самостійно сприймати та обробляти прийняття ігрової пригоди, у тому числі реєструвати його відповідно до ігрового процесу. Всі необхідні дії для відображення сповіщення та прийняття ігрової пригоди розробляються з метою забезпечення плавного та зрозумілого ігрового досвіду для користувачів.

Таблиця 2.25. Основні методи класу QuestGiver

Назва методу	Відповідальність методу
OpenQuestNotificationWindow()	Відображення сповіщення про отримання нової ігрової пригоди
AcceptQuest()	Прийняття ігрової пригоди

Виконується перевірка досягнення конкретної цілі ігрової пригоди. Наприклад факт вбивства ворога для визначення досягнення даної цілі. Проводиться перевірка досягнення конкретної цілі ігрової пригоди. Здійснюється перевірка факту убивства ворога або факту проведеного діалогу з певною особою з метою визначення виконання цільового завдання.

Таблиця 2.26. Основні методи класу QuestGoal

Назва методу	Відповідальність методу
isReached()	Перевірка досягнення конкретної цілі ігрової пригоди
EnemyKilled()	Перевірка чи вбито ворога
talked(GameObject speaker)	Перевірка чи проведено діалог з необхідним об'єктом

У цьому класі здійснюється активація інтерфейсу діалогового торгового вікна з метою його відображення та використання. Забезпечується відображення всіх доступних товарів на візуальному полотні діалогового вікна. Здійснюється очищення полотна від усіх товарів з метою їх прибирання та підготовки до нового відображення. Очікується виявлення

натискання клавіші виходу для подальшої обробки події. Здійснюється вимкнення візуального порту полотна, що припиняє його відображення. Проводиться оновлення списку товарів у продавця після здійснення продажу для підтримки актуальності інформації. Здійснюється отримання шляху до заготовки об'єкту для подальшого використання.

Таблиця 2.27. Основні методи класу shop

Назва методу	Відповідальність методу
showShopWindow(Dictionary<Item, int> other, string file)	Активація інтерфейсу діалогового торгового вікна
renderAllGoods()	Відображення всіх товарів на полотні
destroyAllGoods()	Прибирання усіх товарів з полотна
Update()	Очікування натискання клавіш виходу
Start()	Вимкнення з візуального порту полотна
SaveGoods()	Оновлення товарів у продавця після продажу
GetResourcePath(Item obj)	Отримання шляху до заготовки об'єкту

Здійснюється підготовка кнопки купівлі до можливого натискання з метою активування процесу покупки. Забезпечується відображення інформації про обраний товар на візуальному полотні з метою надання користувачу необхідних деталей. Здійснюється виконання операції віднімання необхідної суми грошей від рахунку користувача та надання обраного товару в обмін. Всі ці дії проводяться автоматизованою системою для забезпечення ефективності та точності процесу.

Таблиця 2.28. Основні методи класу ShopGood

Назва методу	Відповідальність методу
Start()	Підготовка кнопки купівлі до натискання
showChosenGood()	Відображення на полотні інформації про обраний товар
Purchasing()	Віднімання грошей та надання товару

Висновки до другого розділу

В розділі було проведено аналіз та розробку ключових аспектів системи відеогри. Визначені варіанти використання гри та розроблена об'єктно-орієнтована структура системи. Для цього було враховано технічні вимоги, викладені у технічному завданні, а також сюжетну складову гри, яка визначила основні аспекти історії та геймплею.

Проведено розробку логіки різних типів діалогових систем. Розгалужена діалогова система була спроектована для надання гравцям великої кількості варіантів взаємодії та вирішення завдань. Лінійна діалогова система, натомість, пропонує гравцям фіксовану послідовність діалогів для розкриття сюжету гри.

Описано процес розробки алгоритмів, які забезпечують роботу системи гри, що дозволяє ефективно керувати діалогами, подіями та іншими елементами гри. Цей етап гарантує її ефективну та плавну роботу, зокрема при керуванні діалогами, подіями та іншими елементами гри.

Загалом, у другому розділі визначено ключові компоненти та принципи які забезпечують фундаментальну основу для успішної розробки відеогри, включаючи варіанти використання, об'єктно-орієнтовану структуру, діалогові системи, алгоритми та реалізацію ігрового додатку.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				67
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ В ЖАНРІ МЕТРОЇДВАННЯ “DARK WATCHER”

3.1 Порядок встановлення та налаштування системи

"Dark Watcher" — це ігровий додаток, розроблений на платформі Unity 2D. Після створення релізної версії, цю систему можна встановити на будь-який пристрій з підтримуваними платформами. Основні платформи, для яких розроблялася гра, — Windows 11 та MacOS. Незалежно від операційної системи, встановлення додатку відбувається однаковим способом.

Для того, щоб встановити гру, користувач повинен перейти на сторінку інтернет-магазину додатків "STEAM" та у крамниці знайти назву гри. Після вдалого пошуку, користувач може придбати цифрову копію ліцензії, що надає доступ до геймплею. Після придбання, користувач отримує так званий "ключ", який дозволяє додаток з'явитися у бібліотеці ігор користувача. З цієї бібліотеки користувач може завантажити гру на свій пристрій.

1. Приклад оптимальної системи з операційною системою Windows для досягнення мінімум 120 кадрів в секунду (fps):
2. Процесор (CPU): високочастотний процесор Intel Core i7 або AMD Ryzen 7.
3. Графічна карта (GPU): продуктивна NVIDIA GeForce GTX 1660 Super або AMD Radeon RX 5700 з великим обсягом відеопам'яті.
4. Оперативна пам'ять (RAM): мінімум 8 гігабайт (ГБ) оперативної пам'яті.
5. Жорсткий диск (HDD) або твердотільний накопичувач (SSD): рекомендовано використовувати SSD для швидкого завантаження ігор та ефективного доступу до даних.
6. Монітор: роздільна здатність Full HD (1920x1080) з частотою оновлення 120 Гц або вище для відтворення 120 fps.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				68
Змн.	Арк.	№ докум.	Підпис	Дата		

7. Операційна система: Windows 11 64-біт.
8. Живлення: з урахуванням споживання енергії від компонентів варто обрати достатньо потужний блок живлення для живлення всієї системи.

Варто враховувати що це лише рекомендації щодо оптимального апаратного забезпечення Комп'ютера, а мінімальні вимоги були зазначені раніше на етапі проектування.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з системою симуляції

Варто відзначити, що цей підрозділ присвячений конкретній частині гри, яка визначена у темі кваліфікаційної роботи. Інші частини гри можна знайти у кваліфікаційних роботах інших учасників команди.

При входженні в зону взаємодії з об'єктом який є ключовим для сюжету, автоматично запускається діалогове вікно та блокується система переміщення. При натисканні на клавішу «Enter» здійснюється перехід на наступну репліку.

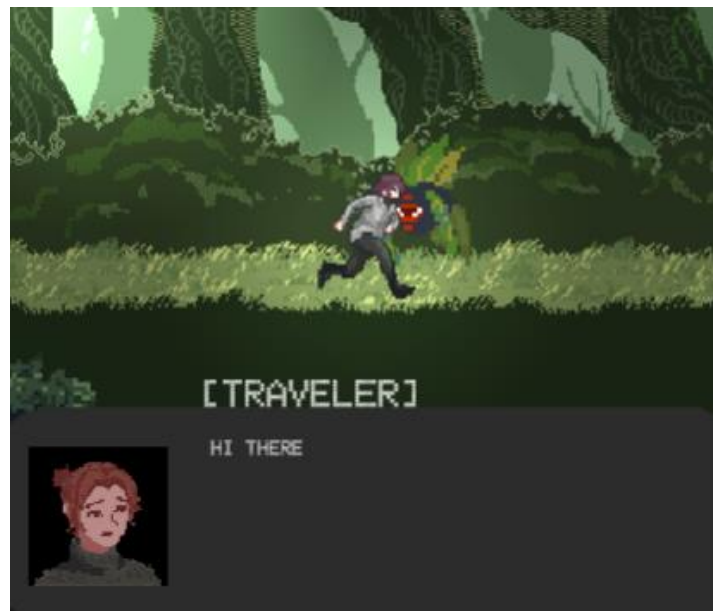


Рисунок 3.1 – Приклад рядка діалогу без можливості вибору (нелінійна діалогова система)

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				69
Змн.	Арк.	№ докум.	Підпис	Дата		

У випадку присутності у діалозі розгалужень гравцеві надається можливість вибору відповіді. В залежності від обраного варіанту отриманий результат буде відрізнятися.

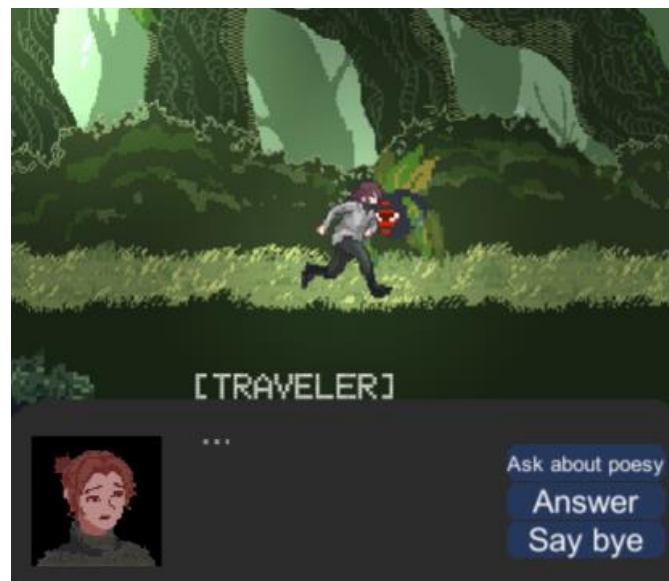


Рисунок 3.2 – Приклад рядка діалогу з можливістю вибору (нелінійна діалогова система)

При взаємодії з неігровим персонажем що відповідає за систему торгівлі, відкривається вікно внутрішньоігрового магазину з доступом до товарів конкретного продавця. Користувач має змогу купити запропоновані позиції в обмеженій кількості за визначеною ціною.

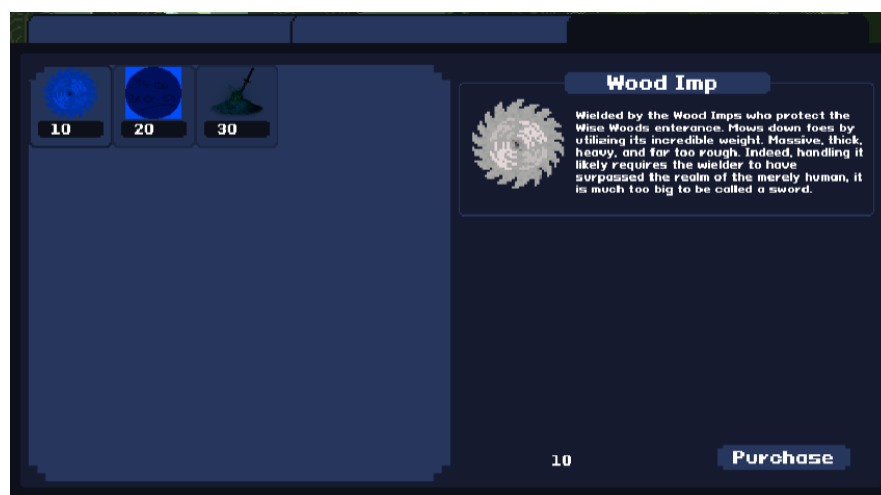


Рисунок 3.3 – Інтерфейс торгівельної системи

При відтворенні анімаційної послідовності автоматично запускається лінійна діалогова система.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				70
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 3.4 – Приклад лінійної діалогової системи з використанням у анімаційних послідовностях



Рисунок 3.5 – Приклад лінійної діалогової системи з використанням у анімаційних послідовностях

Натиснувши на кнопку «N» при запуску гри відобразиться пустий вікно списку ігрових пригод, які при отриманні будуть додані всередину нього. Також можна побачити наявну кількість ігрової валюти.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				71
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 3.6 – Приклад вікна відображення ігрових пригод (пригоди відсутні)

Аналогічно, натиснувши на кнопку «N», відобразиться вікно списку ігрових пригод, які доступні на даний момент. На інтерфейсі відображається Коротка та повна назви ігрової пригоди та її опис.

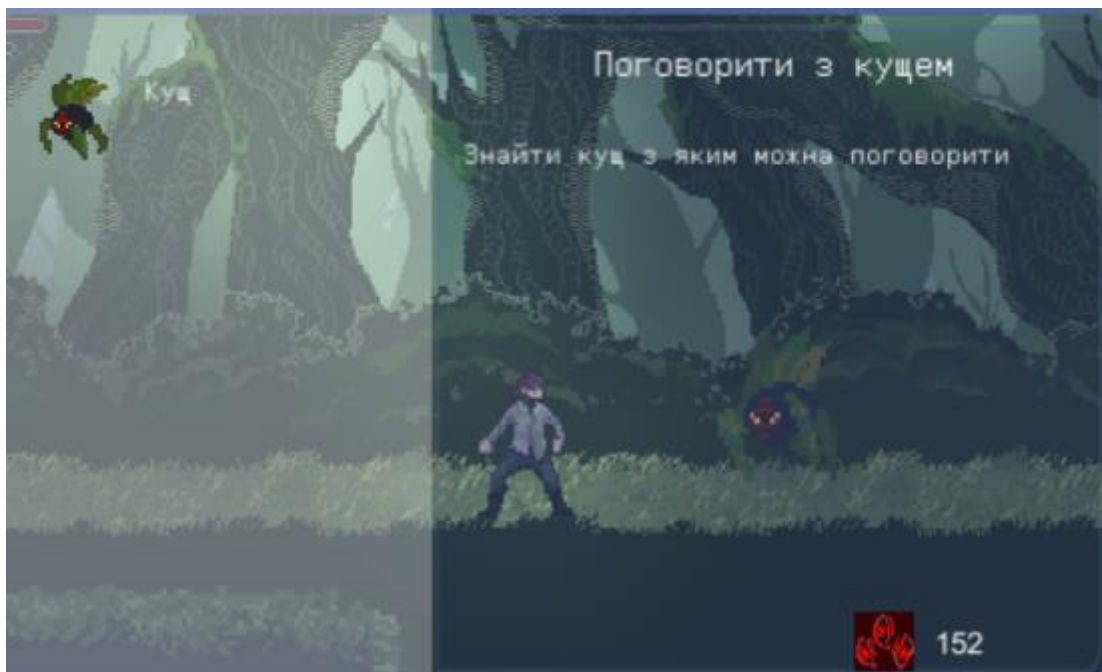


Рисунок 3.7 – Приклад вікна відображення ігрових пригод (доступна пригода)

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				72
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Тестування роботи системи

Тестування системи гри є необхідною складовою під час розробки програмного коду, оскільки допомагає перевірити працездатність системи. Воно дозволяє виявити й усунути помилки, що можуть вплинути на геймплей, логіку гри та інші характеристики. Необхідно забезпечити стабільність роботи системи та задоволення користувачів, а також зберегти репутацію розробника чи видавця гри.

Ось деякі причини, чому тестування є важливим:

- виявлення помилок
- підтвердження відповідності вимогам
- забезпечення якості
- тестування функціональності

Тестування анімації виводу тексту:

Таблиця 3.1. Тестування анімації виводу тексту

Test name: Text printing animation (Issue id #7)			
Status	Passed		
Created date	2023-06-04	Tested at	2023-06-06
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripion	Expected results	
1	Launch the Unity dialog system in the editor	The Unity dialog system should open without any errors	
2	Spawn a new dialog box in the scene	A blank dialog box should appear in the scene	
3	Display a text component to the dialog box	The text component should be displayed within the dialog box	
4	Play the dialog box	The text should start animating according to the defined properties	
5	Verify the text animation	The text should be printed with the expected animation effect	

Тестування роботи виборів у розгалуженій системі діалогу:

Таблиця 3.2. Тестування роботи виборів у розгалуженій системі діалогу

Test name: Check choises in dialog (Issue id #8)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripriion	Expected results	
1	Launch the Unity application.	The Unity application opens successfully.	
2	Navigate to the dialog system.	Access the dialog system feature or module in the Unity application.	
3	Open the dialog with existing choices.	The dialog box with pre-defined choices is displayed on the screen.	
4	Verify the available choices in the dialog.	The dialog box shows the expected choices accurately, matching the predefined options.	
5	Select one of the existing choices.	The selected choice is highlighted or marked as chosen within the dialog box.	
6	Verify the action performed by selecting the choice.	The chosen option triggers the intended action or response in the Unity application as specified.	

Тестування виходу з діалогу:

Таблиця 3.3. Тестування отримання досягнень

Test name: Check exit form dialog (Issue id #9)			
Status	Passed		
Created date	2023-06-07	Tested at	2023-06-08
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripriion	Expected results	

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				74
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження таблиці 3.3

1	Launch the Unity application.	The Unity application opens successfully.
2	Navigate to the dialog system.	Access the dialog system feature or module in the Unity application.
3	Open the dialog that allows exiting.	The dialog box with an exit option is displayed on the screen.
4	Select the exit option in the dialog.	The dialog box closes or transitions out of the view, indicating the exit from the dialog.
5	Verify that the dialog has been exited.	The previous screen or context is restored, and the dialog is no longer visible on the screen.
6	Confirm that any changes made in the dialog are discarded.	Any changes or selections made in the dialog are not saved or applied, and the application retains its previous state.

Тестування купівлі за наявності коштів:

Таблиця 3.4. Тестування купівлі за наявності коштів

Test name: Check ability to buy item with money (Issue id #10)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-09
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripion	Expected results	
1	Launch the application.	The application is opened.	
2	Navigate to the item store.	Access the store or marketplace section within the application.	
3	Locate the desired item for purchase.	The item is displayed with its price and relevant details in the store.	
4	Select the item for purchase.	The item is added to the shopping cart or marked for purchase.	

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження таблиці 3.4

5	Proceed to checkout and payment.	The application prompts for payment and displays the appropriate payment options.
6	Verify successful purchase.	After completing the payment process, the item is added to the user's inventory or account, and the appropriate confirmation or success message is displayed.

Тестування купівлі за відсутності коштів:

Таблиця 3.5. Тестування купівлі за відсутності коштів

Test name: Check disability to buy item without money (Issue id #11)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-09
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripion	Expected results	
1	Launch the application.	The application is opened.	
2	Navigate to the item store.	Access the store or marketplace section within the application.	
3	Locate the desired item for purchase.	The item is displayed with its price and relevant details in the store.	
4	Attempt to select the item for purchase.	The application prevents the user from proceeding with the purchase due to insufficient funds.	
5	Display an error or warning message.	The application displays an appropriate error message indicating the inability to make the purchase due to a lack of.	
6	Prevent further steps in the purchasing process.	The application blocks any further actions related to the purchase, ensuring that the user cannot proceed without the necessary funds.	

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				76
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестування автоматичного приймання ігрових пригод:

Таблиця 3.6. Тестування автоматичного приймання ігрових пригод

Test name: Auto acceptance of the game adventure (Issue id #12)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripriion	Expected results	
1	Launch the application.	The application opens successfully.	
2	Navigate to the quest log or quest interface.	Access the section within the application that displays available quests.	
3	Verify the presence of auto-accept functionality.	The application provides an option or setting for enabling auto-acceptance of quests.	
4	Wait for a new quest to become available.	The application automatically accepts a new quest without user intervention when it becomes available.	
5	Verify the successful auto-acceptance of the quest.	The newly available quest is automatically accepted and added to the player's quest log or active quests list.	

Тестування отримання сюжетно важливого об'єкту:

Таблиця 3.7. Тестування отримання сюжетно важливого об'єкту

Test name: Obtaining an object important to the plot (Issue id #13)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripriion	Expected results	

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				77
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження таблиці 3.7

1	Launch the application.	The application opens successfully.
2	Progress to a specific point in the storyline.	Follow the necessary steps or complete the required quests/missions to reach the point where the plot item is supposed to be received.
3	Trigger the event or interaction to receive the plot item.	Interact with an NPC, object, or trigger in the game that is designed to grant the plot item.
4	Confirm the successful receipt of the plot item.	The application displays a visual indication, message, or notification confirming the acquisition of the plot item.
5	Check the player's inventory or item list.	The plot item is added to the player's inventory or item list, and it is properly categorized or labeled as a plot item.
6	Verify the functionality of the plot item.	The plot item can be used or interacted with in the appropriate context within the game, progressing the storyline or unlocking new content as intended.

Тестування виконання цілі ігрової пригоди:

Таблиця 3.8. Тестування виконання цілі ігрової пригоди

Test name: Completing the objective of the game adventure (Issue id #14)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Nataliia Kozakevych
Pre Conditions: Authorization/Registration, game started			
Steps	Descripriion	Expected results	
1	Launch the application.	The application opens successfully.	

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				78
Змн.	Арк.	№ докум.	Підпис	Дата		

2	Accept and activate the quest.	The quest is accepted and activated in the player's quest log or active quests list.
3	Review the quest goal or objective.	The application displays the specific goal or objective that needs to be completed for the quest.
4	Perform the required actions to meet the quest goal.	The player takes the necessary actions, such as defeating enemies, collecting items, or interacting with specific objects, to fulfill the quest goal.
5	Verify the completion of the quest goal.	The application recognizes and indicates that the quest goal has been successfully achieved or completed.
6	Check for quest progress or rewards.	The application updates the quest status, providing progress information and rewarding the player with experience, items, or other in-game benefits if applicable.

Було проведено тестування геймплею, включаючи основні механіки діалогової та торгівельної систем. Також було забезпечено повне занурення в світ гри. Під час тестування доступного функціоналу не було виявлено жодних помилок чи недоліків. Були виявлені певні проблеми під час виконання окремих тест-кейсів, що були своєчасно виправлені.

Висновки до третього розділу

У третьому розділі було проведено інтенсивне тестування розробленої гри з метою забезпечення оптимального функціонування. Процес тестування дозволив виявити та усунути різноманітні помилки, недоліки та проблеми, що виникали під час гри. Тестування охоплювало широкий спектр функціональних аспектів гри, забезпечуючи відповідність

специфікаціям та вимогам проекту, а також поліпшення загального користувацького досвіду.

Внаслідок проведеного тестування були виявлені та виправлені певні технічні проблеми, включаючи баги, логічні помилки, проблеми з продуктивністю та оптимізацією. Крім того, особлива увага була приділена вірності відображення графічного інтерфейсу, балансу та різноманітності геймплею.

Встановлений порядок роботи з грою включає запуск гри, виконання завдань та досягнення цілей, відтворення та взаємодію з діалогами, а також придбання необхідних товарів. Оптимальний порядок роботи гарантує, що гравці зможуть насолоджуватися грою без зайвих перешкод та проблем.

Завдяки інтенсивному тестуванню та встановленому порядку роботи з грою було досягнуто високої якості геймплею та задоволення користувачів. Розроблена гра виявилася функціональною, стабільною та захоплюючою, надаючи користувачам незабутній досвід та задоволення від гри.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				80
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

За період роботи над кваліфікаційною роботою бакалавра було спроектовано ігровий додаток «Dark Watcher» у жанрі метроїдванія. Початок роботи був спрямований на вибір найкращих інструментів для її розробки.

Після аналізу було встановлено, що найкращим варіантом для розробки є використання мови програмування C# разом з ігровим рушієм Unity, системою Unity Event System та інтерфейсом Scriptable Objects. Цей вибір надає широкий спектр можливостей для розробки ігор, гарантує високу продуктивність та ефективність роботи, а також дозволяє використовувати готові рішення та компоненти для прискорення процесу розробки.

Наступним кроком став збір усієї необхідної інформації для вірної організації робочого процесу у майбутньому. Було проведено аналіз та розробку ключових аспектів системи відеогри. Визначені варіанти використання гри та розроблена об'єктно-орієнтована структура системи. Для цього було враховано технічні вимоги, викладені у технічному завданні, а також сюжетну складову гри, яка визначила основні аспекти історії та геймплею. Проведено розробку логіки різних типів діалогових систем.

Визначено ключові компоненти та принципи які забезпечують фундаментальну основу для успішної розробки відеогри, включаючи варіанти використання, об'єктно-орієнтовану структуру, діалогові системи, алгоритми та реалізацію ігрового додатку.

Наостанок, було проведено інтенсивне тестування розробленої гри з метою забезпечення оптимального функціонування. Тестування охоплювало широкий спектр функціональних аспектів гри, забезпечуючи відповідність специфікаціям та вимогам проекту, а також поліпшення загального користувацького досвіду. Розроблена гра виявилася

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				81
Змн.	Арк.	№ докум.	Підпис	Дата		

функціональною, стабільною та захоплюючою, надаючи користувачам незабутній досвід та задоволення від гри.

Наразі ігровий застосунок доступний для загалу в інтернет-магазині «Steam», де за бажання користувачі можуть завантажити його та отримати власні неповторювані враження від геймплею.

		<i>Н.О.Козакевич</i>			ІПЗ.КР.Б – 121 – 23 – ПЗ	<i>Арк.</i>
		<i>А.В. Морозов</i>				82
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Events & Delegates in Unity [Електронний ресурс] // gamedevbeginner. – 2021. – Режим доступу до ресурсу: <https://gamedevbeginner.com/events-and-delegates-in-unity/>.
2. ScriptableObject Game Events [Електронний ресурс] // Medium. – 2021. – Режим доступу до ресурсу: <https://blog.devgenius.io/scriptableobject-game-events-1f3401bbde72>.
3. Scriptable Objects in Unity (how and when to use them) [Електронний ресурс] // gamedevbeginner. – 2022. – Режим доступу до ресурсу: <https://gamedevbeginner.com/scriptable-objects-in-unity/>.
4. ScriptableObject Tutorial: Getting Started [Електронний ресурс] // Kodeco. – 2019. – Режим доступу до ресурсу: <https://www.kodeco.com/2826197-scriptableObject-tutorial-getting-started>.
5. How to instantiate the sriptable objects [Електронний ресурс] // Toonkorla. – 2023. – Режим доступу до ресурсу: <https://toonkorla.click/scriptableObject-instantiate/>.
6. ТЕОРІЯ ГРАФІВ, для ВЧИТЕЛІВ, для ШКОЛЯРІВ... І НЕ ТІЛЬКИ!, 2017. – 234 с. – (978-5-9710-4069-9).
7. Ferrone H. Learning C# by Developing Games with Unity 2019 / Harrison Ferrone. – Mumbai: Packt, 2019. – 532 с. – (4). – (978-1-78953-205-0).
8. Halpern J. Developing 2D Games with Unity / Jared Halpern. – New York: Apress, 2019. – 398 с. – (978-1-4842-3772-4).
9. Introduction to C# in Unity [Електронний ресурс] // Pluralsight. – 2023. – Режим доступу до ресурсу: https://www.pluralsight.com/courses/introduction-c-unity-1689?utm_source=google&utm_medium=paid-search&utm_campaign=upskilling-and-reskilling&utm_term=ssi-emea-xyz-dynamic&utm_content=free-trial&gclid=CjwKCAjw4ZWkBhA4EiwAVJXwqY-

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				83
Змн.	Арк.	№ докум.	Підпис	Дата		

[VAgddEd98yNHAD6luX6FebKVtVfavLkvCR7isRhpE2lc3OtcIFxoC2x8QAvD_BwE.](https://www.coursera.org/specializations/game-design-and-development)

10. Game Design and Development with Unity 2020 [Электронный ресурс] // coursera. – 2020. – Режим доступа до ресурсу: <https://www.coursera.org/specializations/game-design-and-development>.
11. Junior Programmer [Электронный ресурс] // Unity Learn. – 2023. – Режим доступа до ресурсу: <https://learn.unity.com/pathway/junior-programmer>.
12. В.В. Зиборов. Visual C# 2012 на прикладах / В.В. Зиборов., 2012. – 518 с. – (978-5-9775-0888-9).
13. Unity 2022 Game Development [Электронный ресурс] // Chris' Tutorials. – 2022. – Режим доступа до ресурсу: https://www.youtube.com/watch?v=BpkqwF9P9b0&list=PLyH-qXFkNSxkdYQQeFBNNlmnVj6nQOMOY&ab_channel=Chris%27Tutorials.
14. Практикум по розробці ігор на Unity [Электронный ресурс] // ItStep Academy. – 2023. – Режим доступа до ресурсу: https://itstep.dp.ua/ru/blog_19/knigi-po-unity-3d.
15. Unity User Manual 2021.3 (LTS) [Электронный ресурс] // Unity Documentation. – 2021. – Режим доступа до ресурсу: https://docs.unity3d.com/Manual/index.html?_ga=2.233213726.1082022322.1686527966-1620880330.1686527966.
16. TeamCherry [Электронный ресурс] // TeamCherry. – 2023. – Режим доступа до ресурсу: <https://www.teamcherry.com.au/>.
17. DevCats [Электронный ресурс] // DevCats. – 2023. – Режим доступа до ресурсу: <https://devcats.team/>.
18. Database of everything on Steam. [Электронный ресурс] // SteamDB. – 2023. – Режим доступа до ресурсу: <https://steamdb.info/>.

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				84
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				85
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток А. Відгуки на кваліфікаційну роботу



Вулиця Довженка, 64, Житомир, Житомирська область, 10000.

Тел. +380 067 411 5422.

email: info@inp-software.com

Шановна, Наталія Козакевич.

Після детального огляду вашої роботи, ми хотіли б висловити свої враження та думки про вашу професійну діяльність. Ваші знання та навички в розробці Unity демонструють високий рівень володіння цим інструментарієм.

Ваш дипломний проект показує, що ви маєте глибоке розуміння механік квестів та діалогів. Реалізація цих елементів в продукті вражає своєю якістю та професіоналізмом. Ваша здатність створювати захоплюючі та цікаві ігрові взаємодії підтверджує ваш талант у галузі геймдевелопменту.

Крім того, я високо оцінюю якість коду та функціоналу, які ви реалізували в проекті. Ваш підхід до програмування відображає вашу вправність у вирішенні завдань та вміння створювати оптимальний та стабільний код. Ваша робота вище середнього рівня Strong Junior Unity Developer і свідчить про ваш потенціал для подальшого росту у цій області.

Узагалі, ваша робота підтверджує ваші здібності та професійну підготовку. Ви проявилися як талановитий розробник з ігровою спрямованістю, здатний досягти високих результатів у відповідних посадах. На основі представленого дипломного проекту, я можу охарактеризувати ваш рівень як Middle Game Director.

Ви володієте потенціалом стати цінним активом у галузі розробки ігор, і ми бажаємо вам успіху в подальшій кар'єрі. Ваш внесок у цей проект був високо оцінений, і ми можемо рекомендувати вас як обдарованого фахівця у сфері геймдевелопменту.

Рекомендую оцінити роботу на високому рівні.

З повагою,

Представник компанії INP-SOFTWARE

Рудницький С. Г.



		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				86
Змн.	Арк.	№ докум.	Підпис	Дата		



КОМП'ЮТЕРНА АКАДЕМІЯ IT STEP

вул. Перемоги, буд. 29, м. Житомир, Україна, 10001
тел.: +38 (067) 303-05-33; +38(093) 087-30-33
e-mail: HYPERLINK "mailto:itstep.zhitomir@gmail.com" itstep.zhitomir@gmail.com

Шановна Наталіє Козакевич,

Після детального огляду вашої роботи, ми хотіли б висловити наші враження та думки про вашу професійну діяльність. Ваші навички в галузі розробки Unity продемонстрували високий рівень володіння цим інструментарієм.

Наша команда вражена вашими знаннями та навичками в галузі розробки Unity. Ваші досягнення у створенні лаунчеру, механік квестів та діалогів про вашу здатність створювати захоплюючі та функціональні ігрові елементи. Ваша робота вище середнього рівня Middle Unity Developer і свідчить про ваш потенціал для подальшого росту у цій області.

Ваша здатність досягати результатів у співпраці з іншими розробниками підтверджує ваш потенціал. Ви проявляєте вміння організовувати робочі процеси та працювати в команді для досягнення спільних цілей.

Узагалі, ваша робота підтверджує ваші здібності та професійну підготовку. Ви проявилися як талановитий розробник з глибоким розумінням геймдевелопменту. На основі ваших досягнень, ми можемо охарактеризувати ваш рівень як Middle Unity Developer та Middle Game Director.

Ви володієте потенціалом стати цінним активом у галузі розробки ігор, і ми бажаємо вам успіху в подальшій кар'єрі. Ваш внесок у цей проект був високо оцінений, і ми раді рекомендувати вас як обдарованого фахівця у сфері геймдевелопменту.

Рекомендую оцінити роботу на високому рівні, у літерованій міжнародній системі, як 90A+.

З повагою,
Нікіта Юхимчук
Представник компанії ITSTEP Academy



		Н.О.Козакевич			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		87