

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

«Розробка гри в жанрі horror»

Виконав студент групи КН-20-2
Ринчківський Артем Володимирович

Керівник роботи:
Петросян Руслан Валерікович

Рецензент:
Коротун Ольга Володимирівна

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

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
Марина ГРАФ
«25» січня 2024 р.

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

Здобувач вищої освіти: **Ринчківський Артем Володимирович**
Керівник роботи: **Петросян Руслан Валерікович**
Тема роботи: «Розробка гри в жанрі horror»,
затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**
Термін здачі закінченої роботи: «25» червня 2024 р.
Вихідні дані роботи (зазначається предмет і об'єкт дослідження): Об'єктом дослідження є розробка 3D відеоігор із використанням існуючого ігрового двигуна. Предметом дослідження є створення гри жанру жахів на платформі Unity.

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

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд і аналіз аналогічних проєктів. Розробка технічного завдання. Вивчення літературних джерел.	2 квітня 2024 – 25 квітня 2024	Виконано
2	Розробка архітектури системи	26 квітня 2024 – 8 травня 2024	Виконано
3	Розробка програмного забезпечення	9 травня 2024 – 14 червня 2024	Виконано
4	Перевірка працездатності системи	15 червня 2024 – 22 червня 2024	Виконано
5	Оформлення пояснювальної записки	23 червня 2024 – 25 червня 2024	Виконано
6	Попередній захист	27 червня 2024	Виконано

Здобувач вищої освіти **Артем Ринчківський**
Керівник **Руслан Петросян**

РЕФЕРАТ

Основний зміст роботи викладено на 59 сторінках тексту, містить 4 таблиць, 21 ілюстрацій, 3 додатків, налічує 20 найменувань літературних джерел.

Мета кваліфікаційної роботи є вивчення специфіки проектування архітектури, розробки алгоритмів, впровадження та тестування гри в жанрі horror від першої особи, створеної на базі ігрового рушія Unity.

У роботі запропоновано основні завдання для розробки системи та проведено аналіз аналогічних систем. Представлено сценарії використання програмного забезпечення, загальну структуру системи, а також опис алгоритмів взаємодії між окремими модулями. Подано опис об'єктної структури системи у вигляді діаграми класів. Крім того, наведено результати тестування системи.

КЛЮЧОВІ СЛОВА: SURVIVAL HORROR, UNITY, ГЕЙМПЛЕЙ, ШТУЧНИЙ ІНТЕЛЕКТ, СЦЕНАРІЇ ГРИ, ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ

					КН.КР.Б – 122 – 24 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.	А.В. Ринчківський				Розробка гри в жанрі horror		Літ.	Арк.	Аркуші
Керівник	Р.В. Петросян							3	59
Рецензент	О.В. Коротун						Житомирська політехніка, група КН-20-2		
Н. контр.									
Зав. каф.	М.С. ГРАФ								
					Пояснювальна записка				

ABSTRACT

The main content of the work is set out on 59 pages of text, contains 4 tables, 21 illustrations, 3 appendices, and includes 20 references.

The purpose of the qualification work is to study the specifics of architecture design, algorithm development, implementation and testing of a first-person horror game created on the basis of the Unity game engine.

The paper proposes the main tasks for system development and analyzes similar systems. The paper presents scenarios of software use, the general structure of the system, as well as a description of the algorithms of interaction between individual modules. A description of the object structure of the system in the form of a class diagram is given. In addition, the system testing results are presented.

KEYWORDS: SURVIVAL HORROR, UNITY, GAMEPLAY, ARTIFICIAL INTELLIGENCE, GAME SCENES, PERFORMANCE OPTIMIZATION.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	4
ЗМІСТ	6
РОЗДІЛ 1. ПОШУК МЕТОДІВ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР У ЖАНРІ HORROR.....	8
1.1 Постановка задачі.....	8
1.2 Аналіз аналогів програмного продукту	14
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	16
1.4 Сучасні проблеми та перспективи подальшого розвитку інформаційних технологій для об'єкта дослідження	19
Висновки до розділу 1	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ГРИ В ЖАНРІ HORROR.....	22
2.1 Окреслення способів застосування системи.....	22
2.2 Розробка архітектури системи.....	26
2.3 Створення моделі даних	30
2.4 Розробка алгоритмів функціонування системи	33
2.5 Розробка програмного коду	37
2.6 Реалізація програмного продукту	41
Висновки до розділу 2	42
РОЗДІЛ 3. ОПИС СТВОРЕНОЇ ГРИ У ЖАНРІ HORROR	43
3.1 Опис вимог до системи, умов експлуатації та апаратного забезпечення	43
3.2 Опис та обґрунтування вибору засобів реалізації	44
3.3 Структура інтерфейсу та порядок роботи з програмним продуктом	45
3.4 Тестування роботи програмного продукту.....	50
Висновки до розділу 3	54
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- IDE – (англ. Integrated development environment) – комплексне програмне рішення для розробки програмного забезпечення
- PC – (англ. Personality Computer) - персональний комп'ютер
- AI – (англ. Artificial Intelligence) - штучний інтелект
- UI – (англ. User Interface) - інтерфейс користувача
- LOD – (англ. Level of Detail) - рівень деталізації
- GPU – (англ. Graphics Processing Unit) - графічний процесор

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ПОШУК МЕТОДІВ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР У ЖАНРІ HORROR

1.1 Постановка задачі

Жанр horror є одним із найстаріших та найбільш популярних у світі комп'ютерних ігор. Він включає у себе керування персонажем, який часто відчувається беззахисним та вразливим, здебільшого без належного способу захисту, проти зловісних істот у межах напруженого ігрового світу.

Жанр horror в ігровій індустрії поділяється на декілька піджанрів, серед яких основні:

- Survival Horror
- Psychological Horror
- Action Horror
- Supernatural Horror

Survival Horror є одним із найвизначніших піджанрів в ігровій індустрії, що спеціалізується на створенні напруження, страху та виживання для гравців. Цей жанр виник у 90-х роках минулого століття, і його популярність значно зростає завдяки унікальній співвідносній комбінації елементів жахів та елементів гри з виживання.

Головними особливостями Survival Horror є:

- Атмосфера і настрої

Часто гра базується на затемнених, мрачних місцях, таких як запуснені будівлі, смертельні лабіринти або віддалені місцевості. Звуки, освітлення, музика та звукові ефекти відіграють важливу роль у створенні напруження і підсиленні емоційного враження гравця.

- Обмежені ресурси

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

Це означає обмежений запас амуніції, медичних засобів, їжі або інших необхідних предметів. Гравець повинен стратегічно використовувати ці ресурси, щоб подолати ворожі загрози та вижити.

- Елементи гри з виживання

Наприклад, прийняття рішень про те, чи використовувати обмежену амуніцію для вбивства ворога чи втікання від нього, може визначати успіх або поразку.

- Вороги і їх поведінка

Поведінка цих ворогів може бути непередбачуваною, що створює додатковий страх і викликає потребу в обережності та стратегії у гравця.

- Елементи жаху

Це може включати в себе зустрічі з ворогами, несподівані події, стрибки-переляки, а також інші таємничі або шокуючі моменти, які підсилюють емоційну напругу гравця.

- Фокус на нарахунок

Це може включати в себе такі аспекти, як встановлення мінімум механіки, стратегій управління ресурсами, а також відповідальності за вибір, які визначають характер виживання гравця в тій або іншій ситуації.

Psychological Horror акцентується на психологічних аспектах жахів, створюючи напруження та неприємне відчуття у гравців через використання психологічних трюків, атмосфери та сюжетних елементів.

Головними особливостями Psychological Horror є:

- Психологічні ефекти

Це можуть бути моторошні галюцинації, психологічні травми героїв, зміни реальності або навіть маніпуляція зі свідомістю персонажів.

- Атмосфера та музика

Часто використовуються демпфоване освітлення, похмурі місцевості, непередбачувані звуки і музика, які допомагають

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

поглибити враження від гри і збільшити психологічний тиск на гравця.

- Глибокий сюжет і персонажі

Це може включати в себе розкриття теми ізолювання, безпорадності, втрати розуму чи відчуття втрати контролю.

- Маніпуляція гравцем

У Psychological Horror часто використовується маніпуляція гравцем через зміну сценарію, невідомість або примушує їх підтримувати ситуації з ризиком, не враховуючи того, що їх ви можете

Action Horror поєднує в собі елементи жахів і швидкого, динамічного геймплею.

Головними особливостями Action Horror є:

- Швидкість і динаміка геймплею

Гравець зазвичай стикається зі злом або монстрами, які рухаються або атакують швидко, вимагаючи від нього швидких реакцій та стратегій виживання.

- Елементи бойового мистецтва і вогнепальна зброя

Гравцеві часто доводиться майстерно володіти зброєю, керувати бойовими прийомами або використовувати спеціальні навички для ефективного подолання ворогів.

- Екшн-орієнтовані головоломки

Присутні елементи головоломок, але вони зазвичай орієнтовані на екшн і вимагають від гравця швидкі реакції та фізичного втручання для їх вирішення.

- Атмосфера і стиль

Досягається за допомогою моторошних ворогів, зловісної музики та освітлення, які додатково підсилюють атмосферу жаху.

- Елементи виживання

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Присутні елементи виживання, такі як управління ресурсами (наприклад, амуніція, медичні засоби), необхідність стратегічного планування і прийняття рішень щодо того, як краще використовувати доступні ресурси для виживання.

- Емоційна інтенсивність

Action Horror спрямований на створення інтенсивного емоційного досвіду для гравців, комбінуючи елементи екшну і жаху для досягнення максимальної напруженості і відчуття вовлечення.

Supernatural Horror акцентується на надприродних явищах, сутностях або подіях, що викликають страх у гравців.

Головними особливостями Supernatural Horror є:

- Надприродні явища та сутності

Це можуть бути вампіри, зомбі, привиди, демони, об'єкти з анімацією, культові монстри або інші надприродні істоти. Їхні здатності, поведінка та вплив на гравця часто перевищують закони природи, що підсилює відчуття страху та нервового напруження.

- Атмосфера та настрої

Вона часто використовує містику, темряву, містичні символи, духовність і аномалії для створення напруженого середовища. Атмосфера у Supernatural Horror може бути містичною, жахливою та загадковою, що поглинає гравця в ігровий світ і робить його більш вразливим до страху.

- Гра на психологічних страхах

Гра може використовувати елементи несподіванок, невизначеності, ізоляваності, втрати контролю або невизначених загроз для того, щоб підірвати емоційну стійкість гравця. Це може включати в себе зміну оточення, неочікувані звуки або мерехтливі зображення, що підсилюють відчуття страху та непевності.

- Емоційна інтенсивність

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

Спрямований на створення емоційно інтенсивного досвіду, де гравець постійно відчуває напруження, страх і ажіотаж. Це досягається за допомогою несподіваних подій, загроз і атмосфери, які роблять гравця чутливим до навколишнього середовища і збільшують ступінь емоційної вовлеченості.

- Напруженість і стратегія виживання

Гравцеві потрібно продумано використовувати обмежені ресурси, уникати надприродних загроз і вирішувати головоломки для того, щоб вижити. Це створює додатковий рівень глибини та іммерсії, коли гравець вступає в боротьбу з силою, яка перевищує його звичайні здібності.

Темою даного дипломного проекту був обраний жанр Survival Horror від першої особи. Центральним об'єктом ігрової механіки є персонаж-священик, який повинен вижити у небезпечному лісі, зібрати всі необхідні предмети для проведення обряду та захистити себе від ворогів. Головна камера гри надає вид від першої особи, що дозволяє гравцю зануритися в атмосферу жаху і безпосередньо переживати всі події очима персонажа.

Ігровий персонаж має кілька ключових здібностей:

- Заморожування ворогів за допомогою правої кнопки миші.
- Вбивство заморожених ворогів за допомогою лівої кнопки миші.
- Підбір предметів, необхідних для проведення обряду, за допомогою кнопки "E".
- Вразливість до атак ворогів, що може призвести до смерті персонажа.

Окрім основного завдання виживання, гравець повинен досліджувати ліс у пошуках предметів для проведення обряду. Ці предмети розкидані по різних локаціях, і їх збирання вимагає від гравця дослідження та взаємодії з оточуючим середовищем. Ігровий персонаж повинен переміщуватися по лісу, уникати ворогів або боротися з ними за допомогою своєї унікальної механіки атаки.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Вороги в грі однотипні та контролюються штучним інтелектом. Вони патрулюють по точках, заданих на карті за допомогою NavMeshAgent, і реагують на присутність гравця в певному радіусі, починаючи переслідування. Вороги здатні завдати шкоди або вбити персонажа, якщо він потрапить у їхній радіус дії.

Сюжет гри розпочинається з того, що до церкви надійшов запит від чоловіка, чия донька почала бачити дух жінки у білому, якого ніхто інший не бачить. Церква направляє священника для вигнання духу. Коли священник прибуває до дому чоловіка, він знаходить записку на двері, в якій говориться, що сім'я поїхала до лікарні через погіршення стану доньки. У записці також згадується ротонда, відвідування якої спричинило ці проблеми. Священник вирушає до ротонди, де розуміє, що вона слугувала оберігом місцевості. В середині ротонди він знаходить алтар, на якому проводяться священні обряди, і карту з позначенням предметів для обряду. Гравець має зібрати ці предмети та провести обряд, щоб очистити місцину від злих духів.

Події гри відбуваються в темній ночі в лісі, що підсилює атмосферу страху та небезпеки. Додані шум вітра, музика під час появи духу як скрімер та звуки переслідування ворогом допомагають гравцеві зрозуміти, що ворог десь поблизу.

Графічний інтерфейс гри включає підказки для гравця щодо клавіш управління: заморожування ворогів (права кнопка миші), підбір предметів та взаємодія (кнопка "E") і тд. Крім того, гравець може після того як підібрав карту, активувати її на весь екран для зручності навігації по лісу та пошуку необхідних предметів.

Розробка гри включала використання сучасних технологій та підходів для створення реалістичних моделей ігрових об'єктів та навколишнього середовища, а також розробку ефективного штучного інтелекту для ворогів. Завдяки цьому вдалося досягти високого рівня занурення гравця в ігровий процес та забезпечити захоплюючий ігровий досвід.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2 Аналіз аналогів програмного продукту

На даний момент ігри в жанрі Survival Horror займають значне місце на ринку відеоігор. Вони користуються популярністю серед нових гравців, які звикли до інтерактивного та захоплюючого геймплею, а також серед завзятих шанувальників цього жанру, які цінують атмосферу напруги та страху. Для аналізу аналогів програмного продукту було обрано кілька найпопулярніших представників цього жанру. Серед них для порівняння вибрано чотири гри: Resident Evil 7, Outlast, Amnesia: The Dark Descent та The Forest.

Resident Evil 7 є представником відомої серії ігор Resident Evil. У цій частині основний акцент було зроблено на виживанні та боротьбі з монстрами у замкненому просторі старого будинку. Гра пропонує вид від першої особи, що дозволяє максимально зануритися в ігровий світ. Гравець має обмежену кількість ресурсів, що змушує його ретельно планувати свої дії та економити боєприпаси і ліки. Крім того, гра має складну механіку бою, що вимагає від гравця точності та обережності. Бойові дії відбуваються в різних локаціях, включаючи будинки, підвали та відкриті простори.

Outlast – ще один яскравий представник жанру Survival Horror. Гра пропонує гравцеві роль журналіста, який потрапляє до психіатричної лікарні, наповненої небезпечними пацієнтами. Основною особливістю гри є відсутність можливості захищатися – гравець може лише ховатися та втікати від ворогів. Це створює відчуття безпорадності та постійної загрози, що є ключовим елементом атмосферного хорору. Вороги патрулюють територію лікарні, і гравець повинен уникати їх будь-якою ціною, використовуючи обмежені ресурси, такі як батарейки для камери.

Amnesia: The Dark Descent відома своїм акцентом на психологічному хорорі. Гра пропонує гравцеві досліджувати таємничий замок, зіштовхуючись з різноманітними жахами. Гравець повинен вирішувати головоломки, щоб просуватися далі, при цьому уникаючи монстрів, які можуть його вбити. Важливим аспектом гри є механіка страху – перебуваючи в темряві або

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

зіткнувшись з монстром, персонаж починає втрачати здоровий глузд, що впливає на його сприйняття світу. Гравець має обмежені ресурси, і кожне зіткнення з монстром може бути смертельним.

The Forest – гра, яка поєднує елементи Survival Horror та виживання. Гравець опиняється на острові після авіакатастрофи і повинен вижити, будуючи укриття, шукаючи їжу та обороняючись від мутантів. Гра пропонує відкритий світ, де гравець може вільно досліджувати острів, шукаючи ресурси та розгадуючи таємниці. Важливою частиною ігрового процесу є будівництво та крафтинг, що додає глибини та варіативності. Мутанти патрулюють територію острова, і гравець повинен використовувати всі доступні ресурси, щоб вижити.

Таблиця 1.1

Перелік аналогів програмного продукту

Назва	Розробник	Платформа	Рушій	Дата випуску
Resident Evil 7	Capcom	PC, PS4, Xbox 360	RE Engine	2017
Outlas	Red Barrels	PC,PS4,Xbox360	Unreal Engine 3	2013
Amnesia: The Dark Descent	Frictional Games	PC,PS4,Xbox360	HPL Engine 2	2010
The Forest	Endnight Games	PC,PS4	Unity	2014

Таблиця 1.2

Функціональні можливості

Програма	Заморожування ворогів	Вбивство ворогів	Підбір предметів	Крафтинг	Головоломки
Resident Evil 7	Ні	Так	Так	Ні	Так
Outlas	Ні	Ні	Так	Ні	Ні
Amnesia: The Dark Descent	Ні	Ні	Так	Ні	Так
The Forest	Ні	Так	Так	Так	Ні

За результатами аналізу аналогів програмного продукту видно, що майбутній програмний продукт, що буде створений на основі поставленої задачі, цілком відповідає жанру Survival Horror від першої особи. Майбутня гра матиме

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

унікальну механіку заморожування та вбивства ворогів, що переважає аналоги Outlast та Amnesia: The Dark Descent. Також, на відміну від більшості аналогів, гра включає систему підбору предметів для проведення обряду, що додає глибини ігровому процесу. Атмосфера гри, створена завдяки звуковим та візуальним ефектам, забезпечує високу зануреність гравця у світ жаху.

Збереження прогресу гри здійснюється автоматично після кожного скримера за допомогою серіалізації даних у файл save.gamesave. Під час завантаження гри дані десеріалізуються та підтягуються у гру, що забезпечує безперервність ігрового процесу.

Таким чином, розробка даного програмного продукту значно підвищить рівень ігрового досвіду та запропонує гравцям нові, унікальні можливості для занурення у світ Survival Horror. Майбутня гра буде виділятися серед аналогів своєю унікальною механікою та атмосферою, забезпечуючи захоплюючий ігровий процес та високий рівень інтерактивності.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки гри в жанрі Survival Horror було обрано ігровий рушій Unity. Unity є одним з найбільш популярних ігрових рушіїв завдяки своїй гнучкості, широкому спектру можливостей та підтримці різних платформ. Unity надає інтуїтивно зрозумілий інтерфейс для розробників та підтримує різні мови програмування, зокрема C#.

Основними причинами вибору Unity були наступні:

1. Гнучкість та простота використання: Unity забезпечує простий і зрозумілий інтерфейс для створення та редагування ігрових об'єктів, що значно спрощує процес розробки. Це дозволяє швидко вносити зміни та тестувати нові ідеї.
2. Підтримка різних платформ: Unity підтримує розробку для різних платформ, включаючи Windows, macOS, Linux, iOS, Android, та багато інших. Це дозволяє легко переносити гру на різні пристрої.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Широкий спектр інструментів: Unity надає великий набір інструментів для створення 2D та 3D ігор, а також підтримує роботу з фізикою, анімацією, аудіо та ін. Unity Asset Store пропонує безліч готових ресурсів, які можна використовувати у грі.
4. Активна спільнота та підтримка: Unity має велику та активну спільноту розробників, що дозволяє швидко знайти відповіді на запитання та отримати допомогу. Офіційна документація та численні навчальні матеріали роблять процес навчання та розробки більш ефективним.

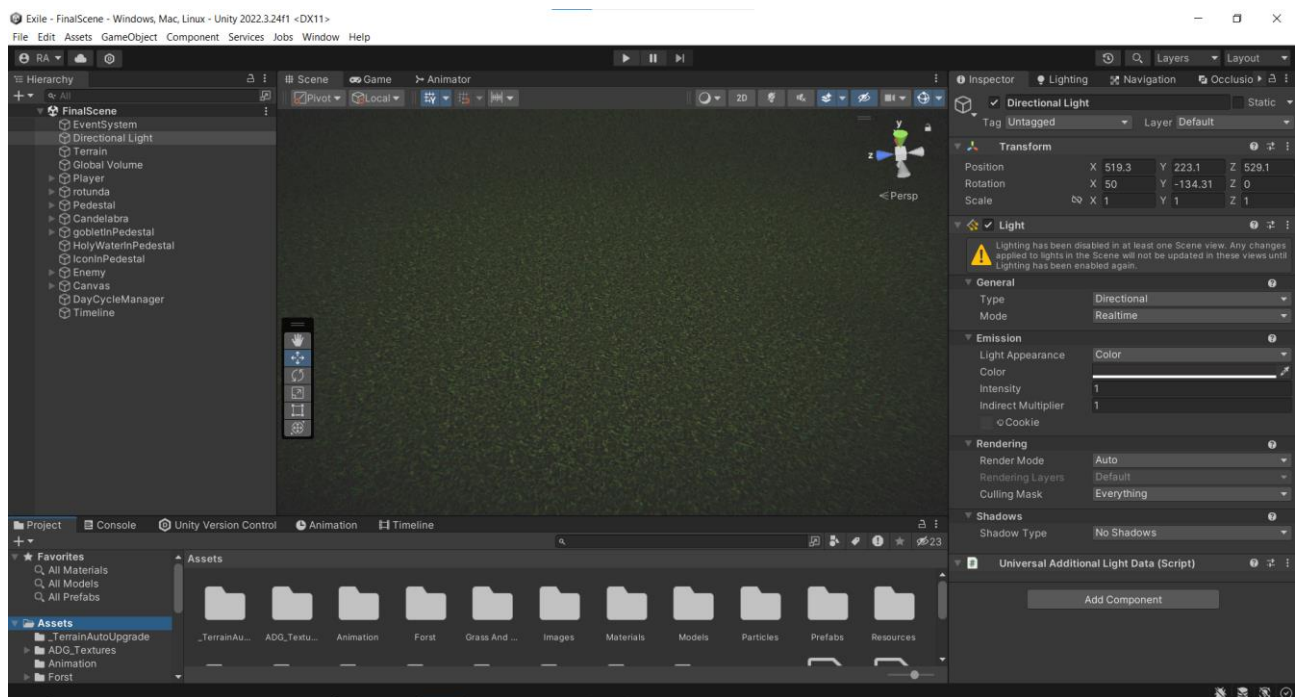


Рисунок 1.1 – Користувацький інтерфейс редактору Unity

Для розробки гри використовувалося інтегроване середовище розробки (IDE) Visual Studio 2022. Visual Studio 2022 забезпечує повну інтеграцію з Unity, що дозволяє зручно редагувати, компілювати та налагоджувати код. IDE надає потужні інструменти для роботи з C#, включаючи автозаповнення коду, інструменти аналізу та відладки, що значно підвищує продуктивність розробки.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

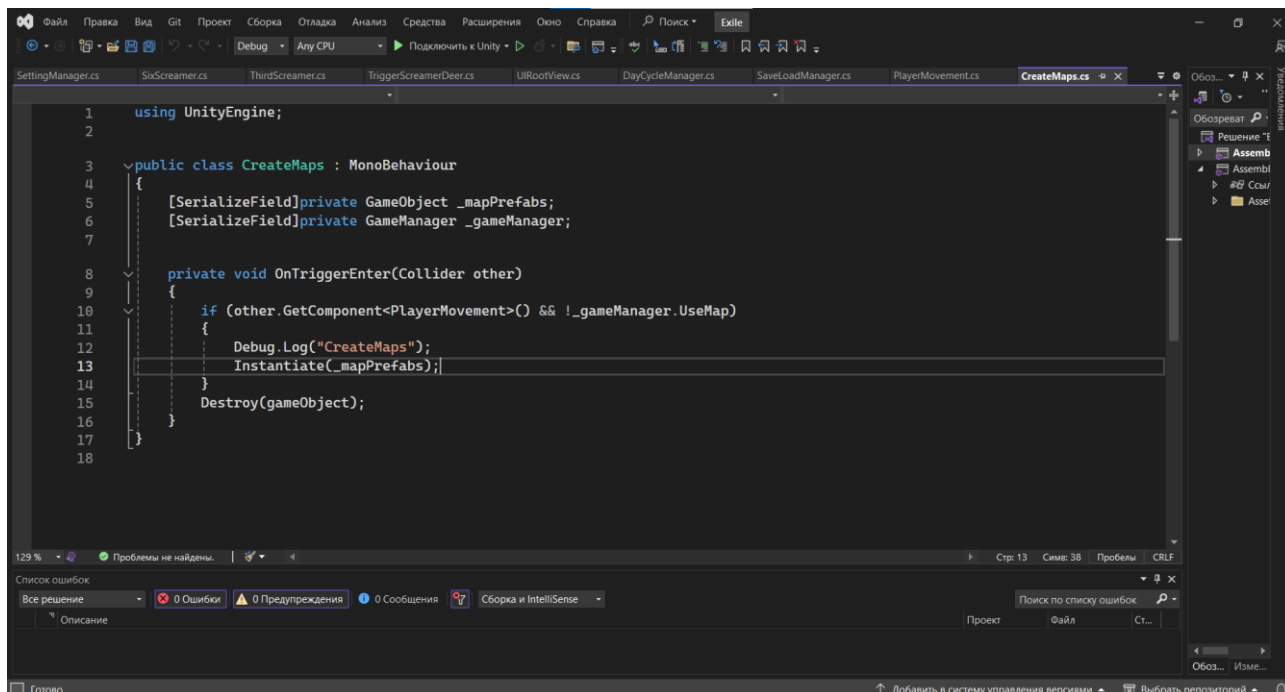


Рисунок 1.2 – Користувацький інтерфейс IDE VisualStudio 2022

В якості системи контролю версій було обрано Git. Git забезпечує надійне управління версіями коду, дозволяючи зберігати історію змін, відстежувати зміни та працювати у команді. Для хостингу репозиторію використовується GitLab, який надає безкоштовні приватні репозиторії та інтеграцію з іншими інструментами розробки.

Для створення 3D моделей та анімацій використовувався Blender. Blender є потужним безкоштовним інструментом для моделювання, анімації та рендерингу 3D графіки. Він підтримує різні формати файлів та інтегрується з Unity, що дозволяє легко імпортувати моделі у гру.

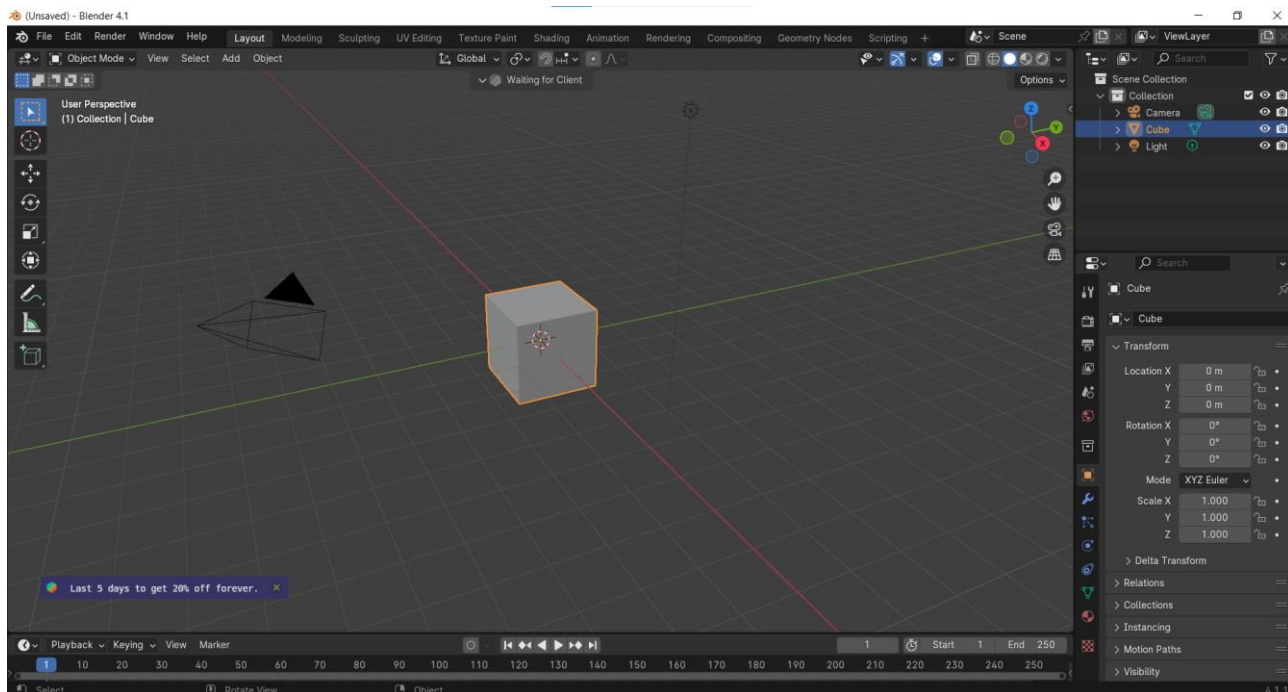


Рисунок 1.3 – Користувачський інтерфейс редактору Blender 4.1

Вимоги до апаратного забезпечення були визначені на основі вимог схожих програмних продуктів, реалізованих на ігровому рушії Unity:

Операційна система: Windows 10 x64

Процесор: Intel Core i5 або AMD Ryzen 5

Оперативна пам'ять: 8 ГБ

Відеокарта: NVIDIA GeForce GTX 1060 або AMD Radeon RX 580

DirectX: Версія 11

Місце на диску: 15 ГБ

Вибір зазначених інструментальних засобів та апаратного забезпечення дозволяє забезпечити ефективну розробку, тестування та запуск гри, відповідно до вимог проекту.

1.4 Сучасні проблеми та перспективи подальшого розвитку інформаційних технологій для об'єкта дослідження

Однією з головних проблем є оптимізація продуктивності. Хоррор-ігри часто містять великі, деталізовані світи, складну механіку взаємодії багатопланове звукове оформлення, що створює високе навантаження на апаратні

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

ресурси. Оптимізація графіки, використання більш ефективних алгоритмів обробки даних та покращене управління ресурсами важливі для забезпечення плавного ігрового процесу навіть на системах з нижчими технічними характеристиками. Використання таких технологій, як LOD (рівень деталізації) для зменшення навантаження на графічний процесор та завантаження асинхронних ресурсів, може допомогти в цьому напрямку.

Ще одним важливим завданням є створення реалістичного штучного інтелекту для противника. У грі horror ворог повинен переслідувати гравця, а також демонструвати складну поведінку, таку як засідка, обхід перешкод та реагування на звуки та інші сигнали. Машинне навчання та нейронні мережі допоможуть вам створити більш адаптивний та реалістичний штучний інтелект. Однак інтеграція таких технологій вимагає значних обчислювальних ресурсів і складних алгоритмів налаштування.

Якість графіки та звуку також є важливою сферою розвитку. У жанрі horror якість візуальних і акустичних ефектів відіграє важливу роль у створенні атмосфери. Сучасні технології рендеринга, такі як трасування променів, можуть значно поліпшити візуальне сприйняття гри.

Розвиток технологій віртуальної реальності також відкриває нові перспективи для ігор в жанрі horror. Віртуальна реальність дозволяє гравцям повністю зануритися в ігровий світ, значно посилюючи емоційний вплив гри. Однак розробка VR-ігор вимагає певних знань і навичок, а також значних інвестицій в спеціальне обладнання та програмне забезпечення.

Інноваційні способи взаємодії з гравцями також можуть покращити ігровий процес. Використання сенсорних технологій, таких як розпізнавання жестів і рухів гравця, може вивести взаємодію на новий рівень. Крім того, інтеграція технології доповненої реальності (AR) може створити цікаву ігрову механіку, в якій реальний і віртуальний світи зливаються в єдине.

Висновки до розділу 1

У цьому розділі було розглянуто особливості жанру Survival Horror від першої особи та сформульовано завдання для розробки гри в цьому жанрі. Після

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

цього було проведено аналіз існуючих аналогів та створено порівняльні таблиці для оцінки їхніх переваг і недоліків. Крім того, було обґрунтовано вибір інструментів для реалізації програмного продукту, включаючи ігровий рушій, середовище розробки, систему контролю версій та інструменти для створення графічного контенту. Нарешті, було визначено вимоги до апаратного забезпечення для ефективної розробки та запуску гри.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ГРИ В ЖАНРІ HORROR

2.1 Окреслення способів застосування системи

У цій грі визначено ряд функціональних вимог, що можна поділити на декілька логічних частин, кожна з яких має свій перелік варіантів використання.

Центральною частиною функціоналу є персонаж гравця. Його основні можливості включають переміщення, яке може здійснюватися у двох режимах: ходьба та біг. Це виділяє два основні прецеденти: Walk і Sprint. Крім того, персонаж має можливість атакувати ворогів шляхом їх заморожування та вбивства, що виділяє прецеденти Freeze enemy та Kill enemy. Персонаж також може підбирати предмети, необхідні для проведення обряду, та взаємодіяти з об'єктами на ігровому рівні, що виділяє прецеденти Pick up item, Place item on altar та Light candles.

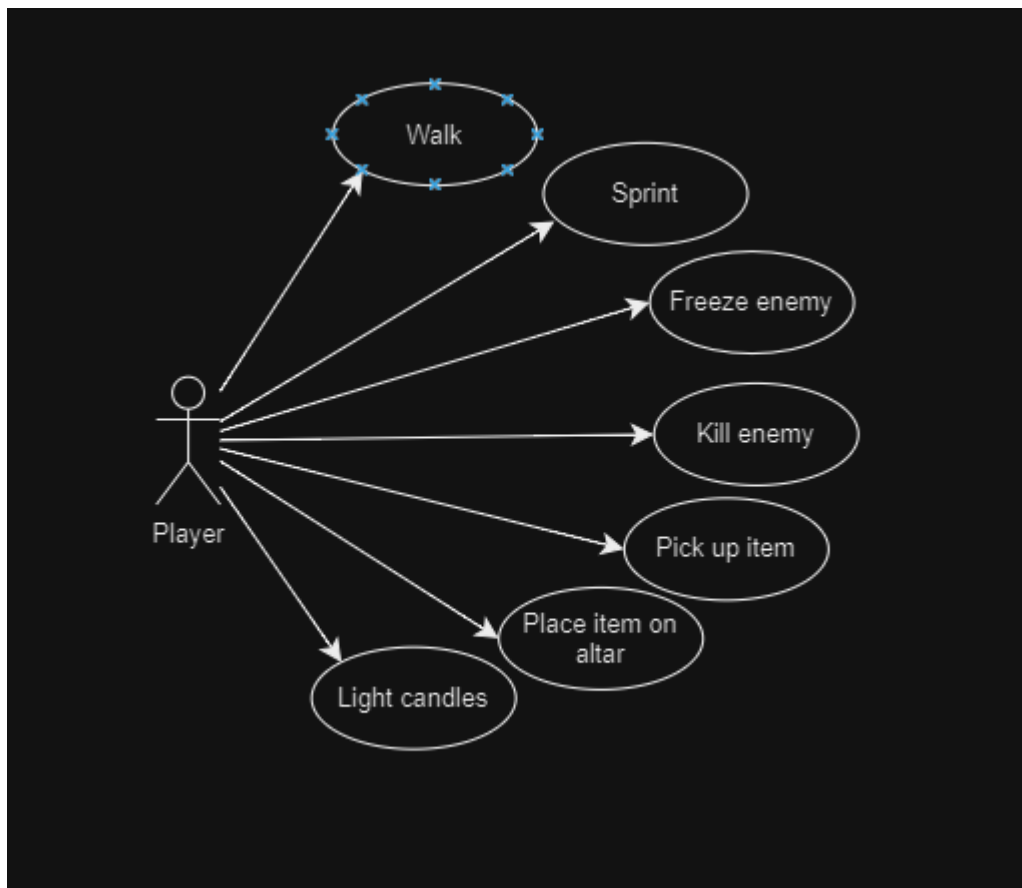


Рисунок 2.1 – Діаграма прецедентів персонажа

Важливою частиною гри є вороги. Вони переслідують персонажа гравця і, догнавши його, повідомляють про його смерть. За це відповідає прецедент Chase player. Вороги можуть лише переслідувати та вбивати гравця, що виділяє прецедент Kill player.

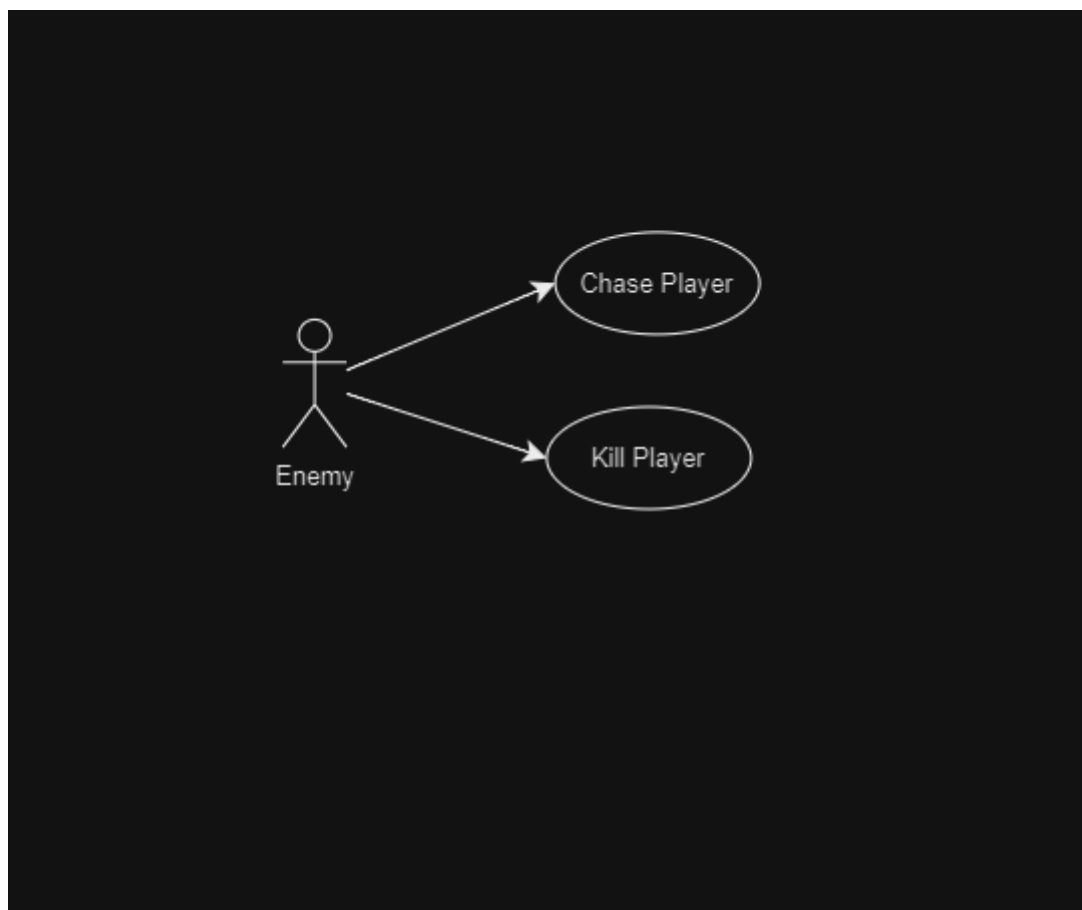


Рисунок 2.2 – Діаграма прецедентів ворогів

Інвентар є невід'ємною частиною гри, де зберігаються підібрані предмети, необхідні для обряду. Гравець може використовувати ці предмети, наприклад, поставити їх на алтар або запалювати свічки. За ці дії відповідають прецеденти Place item on altar та Light candles.

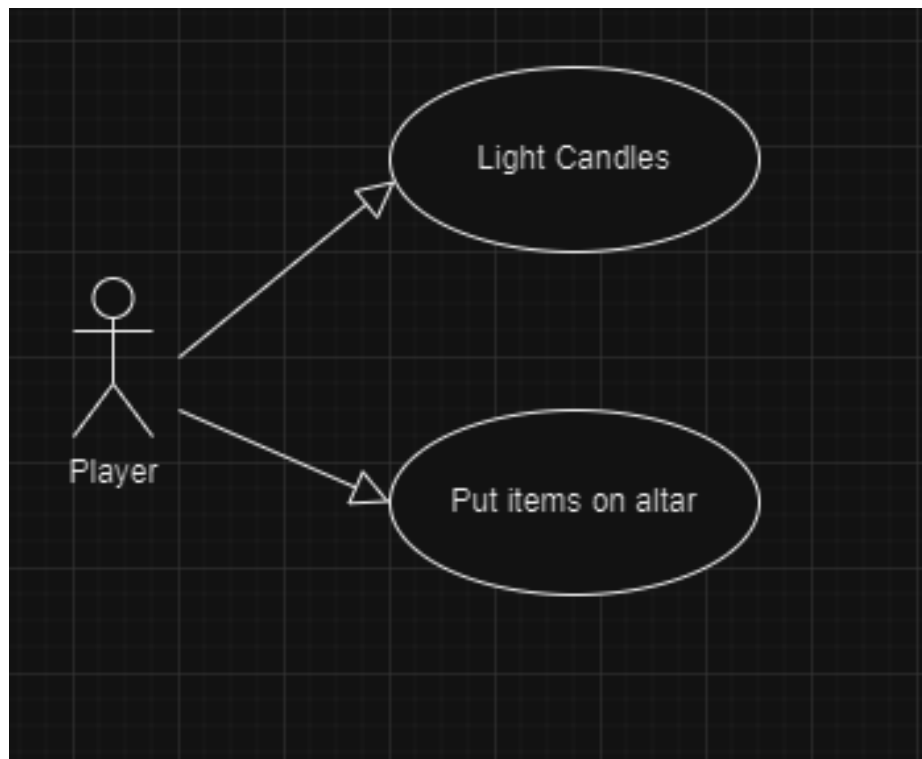


Рисунок 2.3 – Діаграма прецедентів інвентарю

Гравець також може взаємодіяти з об'єктами на ігровому рівні. Наприклад, при першому наближенні до алтаря з'являється скример. Гравець може ставити предмети на алтар та запалювати свічки, якщо у нього є сірники. За ці взаємодії відповідають прецеденти Trigger screamer та Use altar.

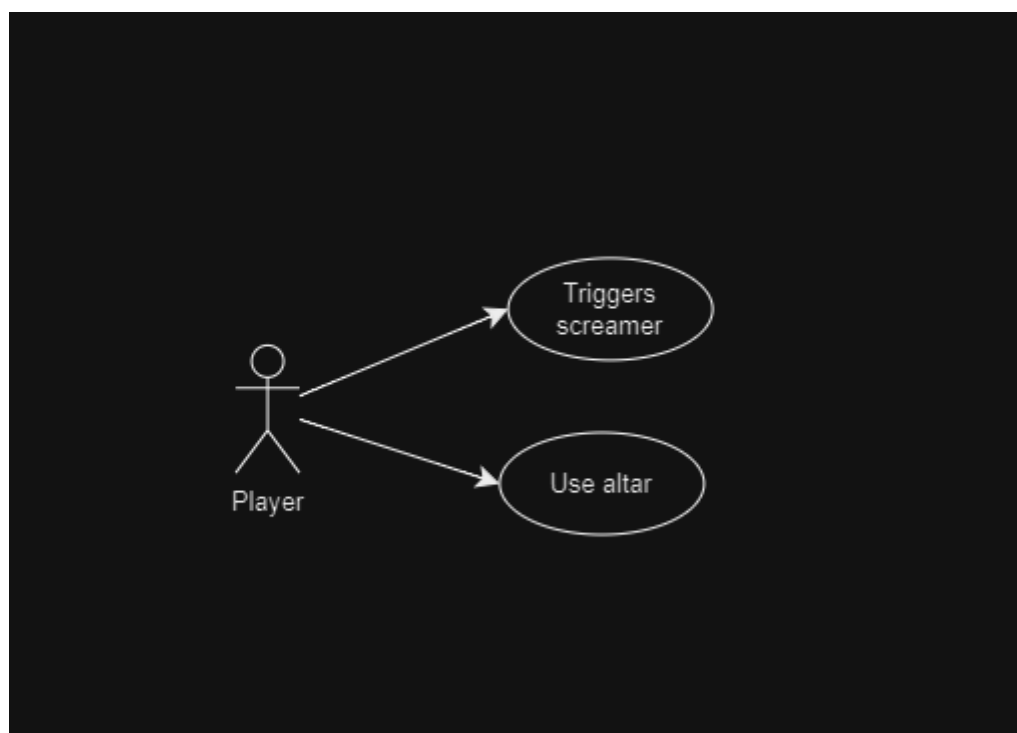


Рисунок 2.4 – Діаграма прецедентів ігрового рівня

Інтерфейс користувача дозволяє доступ до різних меню гри. Через головне меню гравець може запустити гру, відкрити налаштування або вийти з гри. Через меню паузи у грі можна відкрити налаштування, продовжити гру або вийти у головне меню. Меню налаштувань дозволяє змінювати чутливість миші та гучність. За ці функції відповідають прецеденти Open main menu, Open pause menu та Open settings menu.

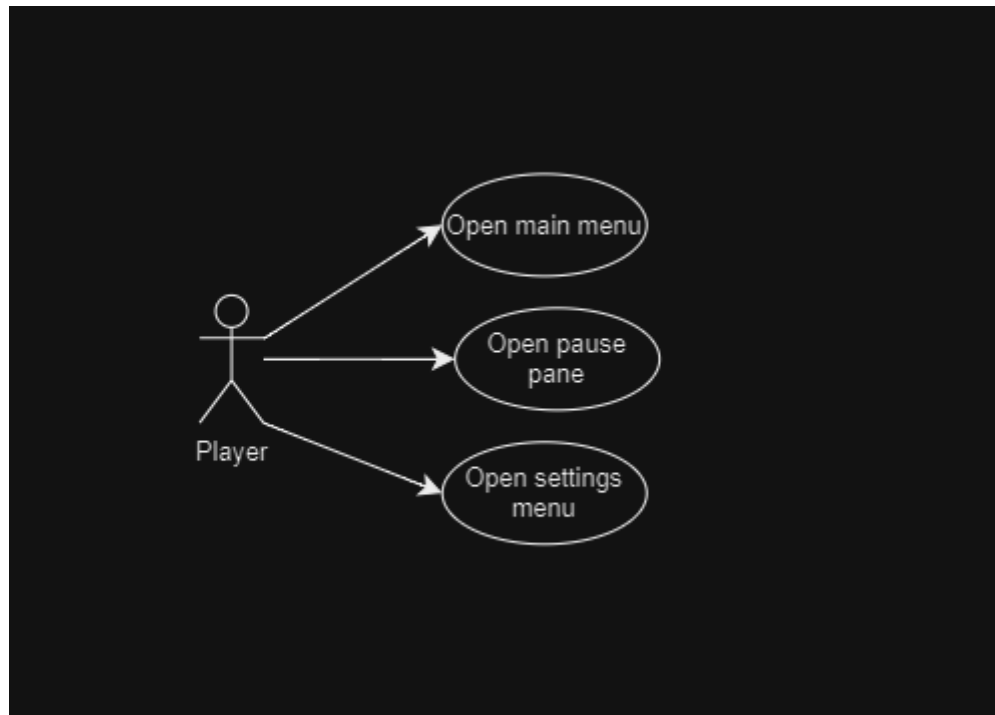


Рисунок 2.5 – Діаграма прецедентів меню гри

Усі визначені варіанти використання системи були об'єднані в загальну діаграму прецедентів для визначення загальної складності продукту, спрощення перегляду доступних варіантів використання та уникнення дублювання варіантів використання в різних контекстах системи. Основна увага при складанні діаграми приділялася її читабельності.

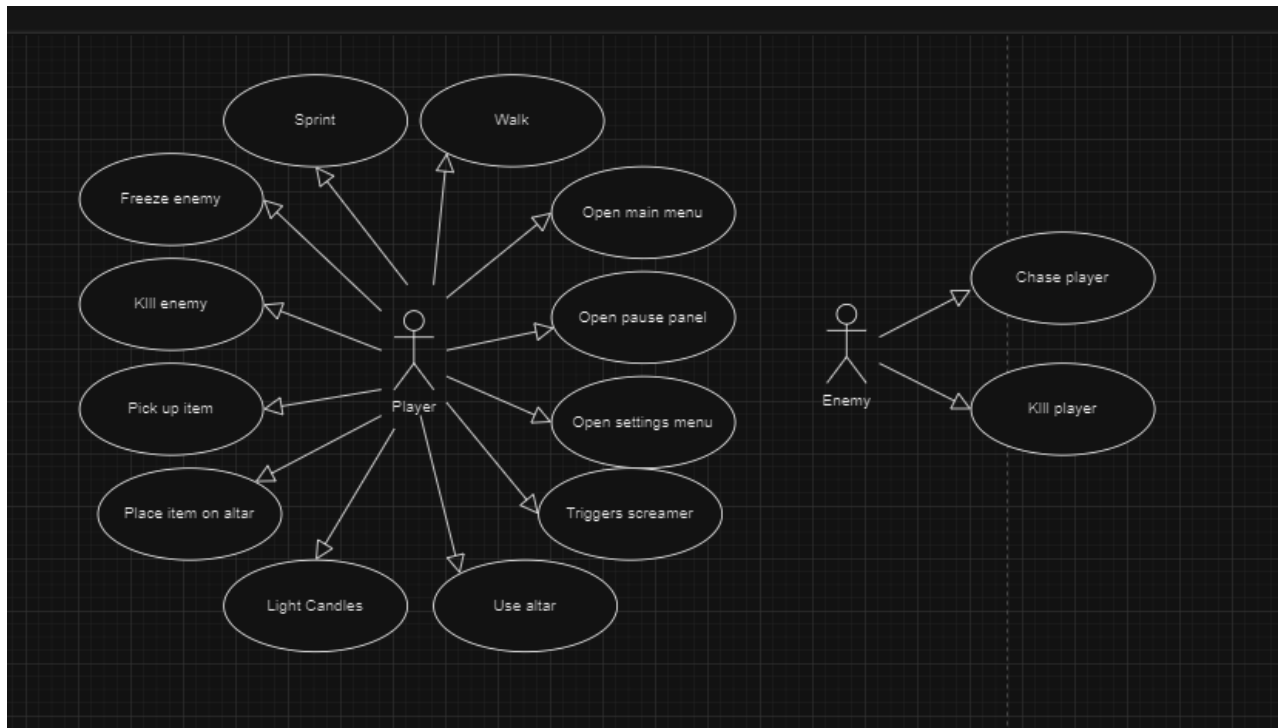


Рисунок 2.6 – Загальна діаграма прецедентів

2.2 Розробка архітектури системи

Проектування структури системи для комп'ютерної гри в жанрі Survival Horror передбачає розробку архітектури, яка забезпечує гнучкість, масштабованість та можливість подальшого розширення функціональності. Це досягається за рахунок принципу модульності, що дозволяє легко додавати нові механіки та замінювати існуючі без значних змін у загальній структурі системи.

Для реалізації принципу модульності було вирішено розділити функціонал гри на окремі модулі, кожен з яких відповідає за певний аспект гри. Основними модулями є: модуль гравця, модуль ворогів, модуль загальних механік гри та модуль управління інтерфейсом і звуком. Такий підхід забезпечує зменшення залежностей між компонентами і спрощує їх інтеграцію та тестування.

Основними цілями при проектуванні архітектури гри були:

Гнучкість: Забезпечення можливості легкого розширення функціональності гри шляхом додавання нових модулів або модифікації існуючих.

Модульність: Розділення функціоналу на окремі, незалежні модулі, що дозволяє легше розробляти, тестувати та підтримувати систему.

Масштабованість: Створення архітектури, яка може підтримувати збільшення кількості та складності механік без значних змін у базовій структурі.

Модуль гравця:

PlayerMovement: Клас, що відповідає за переміщення гравця. Він обробляє введення від користувача для ходьби та бігу, змінюючи відповідно швидкість руху персонажа.

PlayerAttack: Клас, що реалізує механіку атаки. Гравець може заморожувати ворогів і вбивати їх, використовуючи відповідні дії. Цей клас забезпечує анімації та ефекти атаки.

PlayerInteraction: Клас для взаємодії з об'єктами в грі. Гравець може підбирати предмети, взаємодіяти з алтарем та запалювати свічки. Цей клас також відповідає за перевірку наявності предметів в інвентарі гравця та їх правильне використання.

PlayerDead: Клас, що обробляє смерть гравця. Він зменшує здоров'я гравця при отриманні шкоди та викликає анімацію смерті. Цей клас також керує подіями, що відбуваються після смерті гравця, такими як перезавантаження рівня або показ екрану смерті.

Модуль ворогів:

EnemyBehavior: Клас для управління поведінкою ворогів. Вороги патрулюють певні області, переслідують гравця, коли його бачать, та атакують його при наближенні. Цей клас також реалізує різні стратегії поведінки ворогів, такі як засади або групові атаки.

EnemyAI: Клас, що реалізує штучний інтелект ворогів, використовуючи навігаційні сітки для патрулювання та переслідування гравця. Цей клас забезпечує адаптивну поведінку ворогів, враховуючи дії гравця та змінюючи свої стратегії відповідно.

Модуль загальних механік гри:

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

GameController: Клас, що відповідає за загальні механіки гри, такі як збереження та завантаження прогресу, управління рівнями та перевірка умов завершення гри. Цей клас також координує взаємодію між різними модулями гри, забезпечуючи їхню узгоджену роботу. У GameController зберігається інвентар гравця, який зберігає всі підібрані предмети та забезпечує їхню доступність для використання під час гри.

UIController: Клас для управління інтерфейсом користувача. Він забезпечує роботу меню, налаштувань та панелі паузи. UIController відповідає за відображення важливої інформації для гравця, такої як здоров'я, інвентар та підказки.

AudioController: Клас, що відповідає за звукові ефекти та музичний супровід гри. Він керує відтворенням звуків під час подій гри та змінює гучність відповідно до налаштувань користувача. Цей клас також забезпечує динамічну зміну музики та звукових ефектів залежно від ситуації в грі.

Всі модулі гри взаємодіють між собою через чітко визначені інтерфейси. Це забезпечує мінімізацію залежностей між компонентами і спрощує їх заміну та тестування. Наприклад, модуль гравця взаємодіє з GameController для підбору предметів та з модулем ворогів для атаки.

Приклад взаємодії модулів:

Коли гравець наближається до предмета, клас **PlayerInteraction** викликає метод **AddItem** з класу **GameController**, щоб додати предмет до інвентарю.

Коли ворог помічає гравця, клас **EnemyBehavior** викликає метод **ChasePlayer** для початку переслідування.

Після того, як ворог досягне гравця, клас **EnemyBehavior** викликає метод **KillPlayer** з класу **PlayerDead**, щоб обробити смерть гравця.

Основні компоненти та їхні ролі:

Player: Головний об'єкт гравця, що агрегує всі компоненти, пов'язані з управлінням персонажем.

Enemy: Головний об'єкт ворога, що агрегує всі компоненти, пов'язані з управлінням ворогом.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

GameController: Центральний контролер гри, що координує взаємодію між всіма іншими модулями та зберігає інвентар гравця.

UIController: Контролер для управління інтерфейсом користувача.

AudioController: Контролер для управління звуковими ефектами та музикою.

Ця архітектура дозволяє легко адаптувати гру під нові вимоги, додаючи нові модулі або розширюючи функціональність існуючих без значних змін у кодовій базі. Це робить систему більш стабільною і зручною для подальшого розвитку.

Для кращого розуміння структури системи та взаємодії між її компонентами були розроблені діаграми класів і діаграми взаємодії. Ці діаграми демонструють, як окремі класи і модулі взаємодіють між собою, які методи викликаються і які дані передаються між компонентами.

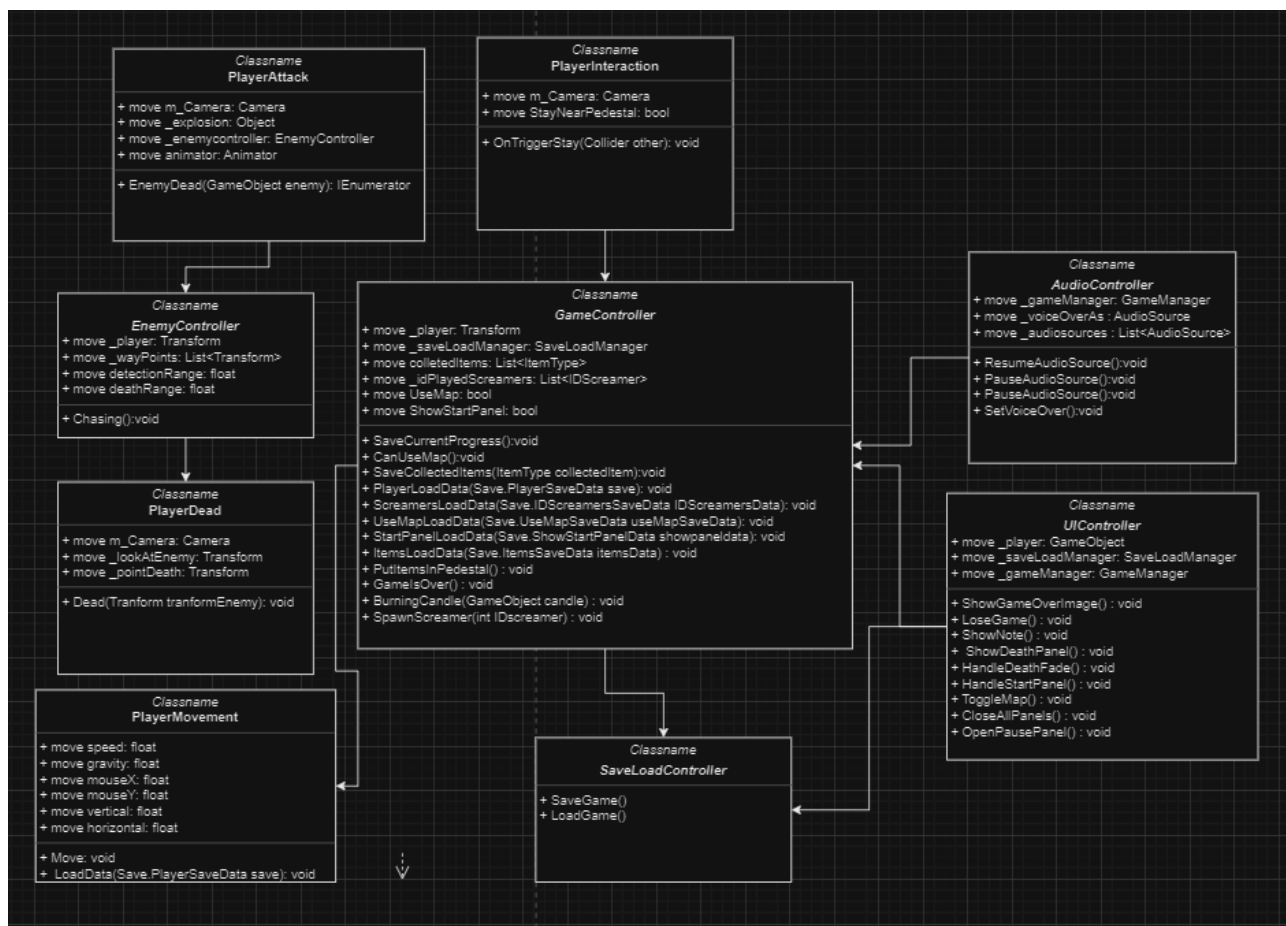


Рисунок 2.7 – Діаграма основних класів функціональності

Змн.	Арк.	№ докум.	Підпис	Дата

2.3 Створення моделі даних

У проєкті гри в жанрі Survival Horror використовується серіалізація для зберігання інформації про стан гравця, зокрема його позицію, напрямок та підібрані предмети. Дані серіалізуються у файл за допомогою BinaryFormatter, що дозволяє зберігати та відновлювати стан гри.

Оскільки стандартні типи даних Unity, такі як Vector3 і Quaternion, не можуть бути напряму серіалізовані, були створені власні структури для зберігання позиції та повороту гравця.

Основні компоненти моделі даних:

Структура Vec3

Опис: Структура для зберігання позиції гравця. Вона містить три компоненти: x, y, z, які відповідають координатам позиції у тривимірному просторі.

Структура Quat

Опис: Структура для зберігання повороту гравця. Вона містить чотири компоненти: x, y, z, w, які відповідають компонентам кватерніона для представлення повороту в тривимірному просторі.

Тип ItemType

Опис: Перелічуваний тип, що описує різні типи предметів, які можуть бути підібрані гравцем. До таких типів належать, наприклад, зілля здоров'я, зілля мани, ключі тощо.

Клас Save

Опис: Клас для зберігання всіх даних про стан гри, включаючи позицію, поворот гравця, підібрані предмети та інші важливі параметри.

Властивості:

PlayerData: PlayerSaveData: Зберігає позицію та поворот гравця.

ItemsData: ItemsSaveData: Зберігає список підібраних предметів.

IDScreamersData: IDScreamersSaveData: Зберігає ідентифікатори скримерів, що були показані.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

UseMapData: UseMapSaveData: Зберігає інформацію про використання карти.

ShowPanelData: ShowStartPanelData: Зберігає інформацію про показ панелі.

Дані про стан гравця серіалізуються та зберігаються у файл під час збереження гри, а також десеріалізуються при завантаженні гри для відновлення стану гравця.

Під час збереження гри створюється об'єкт Save, в який записуються всі необхідні дані про стан гри. Цей об'єкт серіалізується у файл за допомогою BinaryFormatter.

При завантаженні гри дані читаються з файлу і використовуються для відновлення позиції, повороту та списку підібраних предметів гравця. Це дозволяє гравцеві продовжити гру з того місця, де він зупинився.

Клас GameController відповідає за завантаження даних при старті гри і налаштування початкового стану гравця. Він отримує дані про позицію та поворот гравця, а також про підібрані предмети, і відновлює їх у грі.

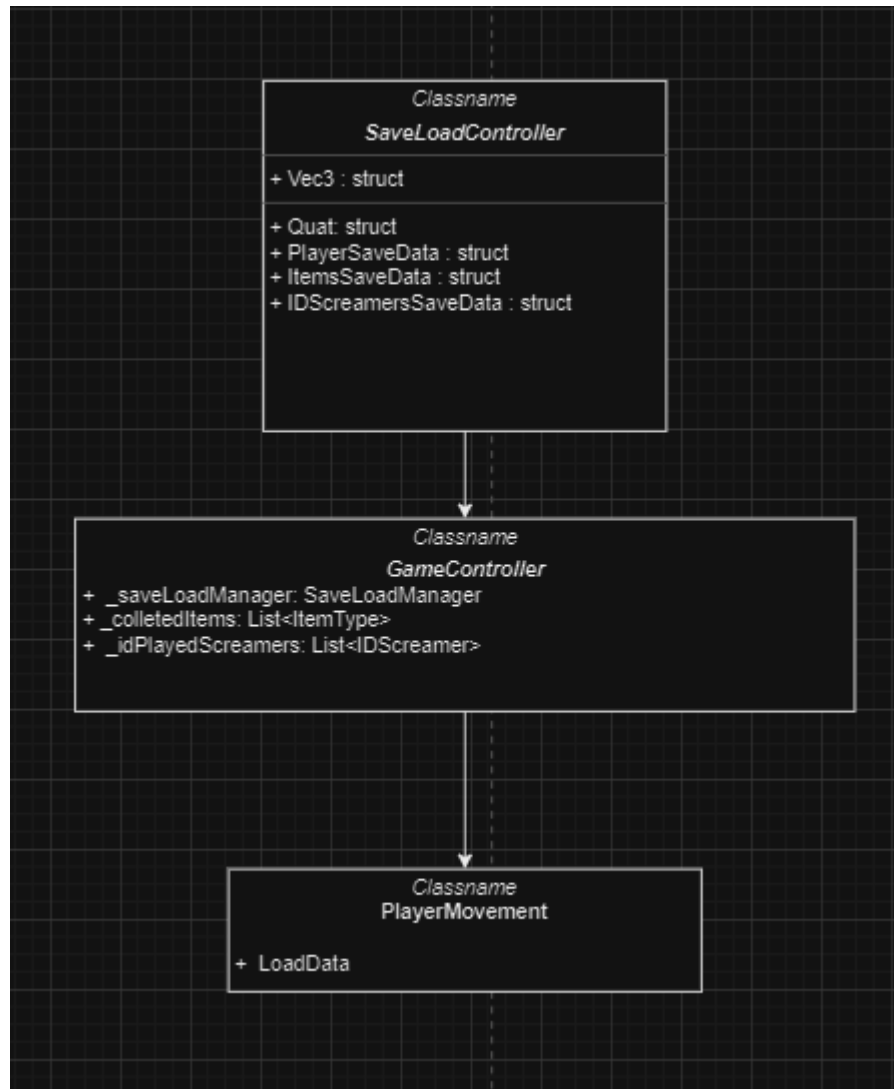


Рисунок 2.8 – Діаграма моделі даних системи

Процес завантаження даних:

1. Читання файлу збереження: GameController відкриває файл збереження і читає з нього дані.
2. Відновлення позиції та повороту: Позиція та поворот гравця встановлюються відповідно до збережених даних.
3. Відновлення підібраних предметів: Список підібраних предметів відновлюється з даних файлу і відповідно налаштовується у грі.

Розробка моделі даних для гри в жанрі Survival Horror передбачає використання серіалізації для зберігання інформації про стан гравця. Це включає позицію, поворот та список підібраних предметів. Використання структур для зберігання цих даних забезпечує гнучкість та зручність управління контентом гри. Такий

підхід дозволяє легко зберігати і відновлювати стан гри, забезпечуючи безперервність ігрового процесу для гравця.

2.4 Розробка алгоритмів функціонування системи

У цьому розділі описано проектування та реалізацію основних алгоритмів, які забезпечують функціонування гри в жанрі Survival Horror. Для запобігання дублювання коду та спрощення структури алгоритмів більшість з них реалізовано в окремих допоміжних класах-менеджерах, що забезпечує модульність і знижує ризик помилок.

Алгоритм переслідування ворогами реалізовано за допомогою системи NavMeshAgent у Unity, яка дозволяє ворогам переміщуватися по заданому маршруту та реагувати на появу гравця в їхньому полі зору.

Патрулювання:

- Вороги патрулюють між визначеними точками на карті. Після досягнення кожної точки вони затримуються на деякий час перед рухом до наступної точки.
- Використовується NavMeshAgent для визначення маршруту патрулювання.

Виявлення гравця:

- Вороги мають певний радіус виявлення, в якому вони можуть помітити гравця.
- Якщо гравець потрапляє в цей радіус, вороги переходять у режим переслідування.

Переслідування:

- В режимі переслідування вороги використовують NavMeshAgent для навігації до гравця.
- Якщо ворог наближається до гравця на достатню відстань, він атакує гравця, реалізуючи відповідну анімацію та звук атаки.

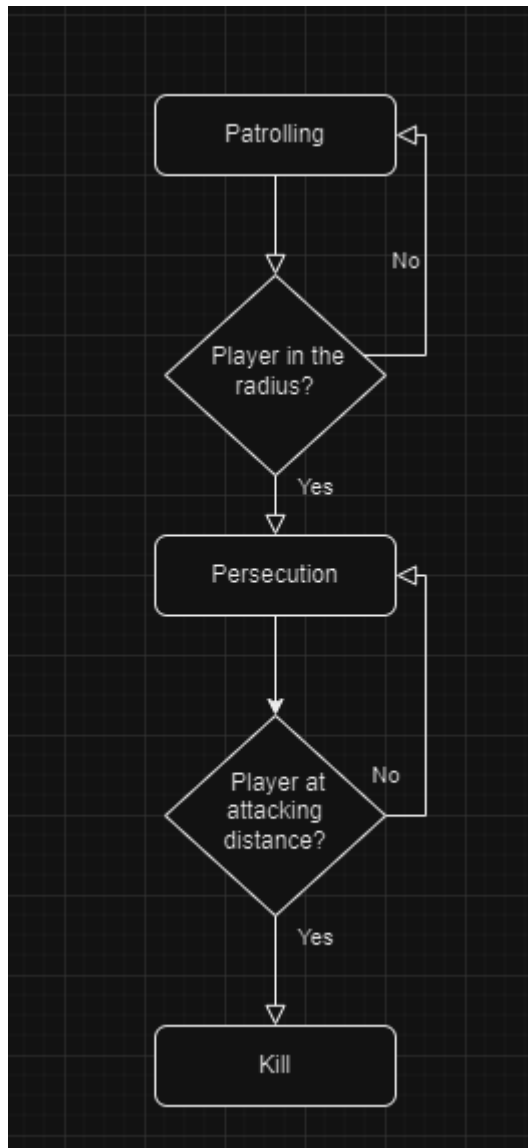


Рисунок 2.9 – Діаграма діяльності ігрового інтелекту ворога

Для збереження і завантаження даних гри використовується серіалізація об'єктів у файл за допомогою BinaryFormatter. Це дозволяє зберігати та відновлювати стан гри, включаючи позицію, поворот гравця та підібрані предмети.

Збереження даних:

- Створюється об'єкт Save, який містить всю необхідну інформацію про стан гри: позицію і поворот гравця, список підібраних предметів та інші параметри.
- Об'єкт Save серіалізується у файл за допомогою BinaryFormatter і зберігається у визначеному місці на диску.

Завантаження даних:

- При завантаженні гри файл збереження відкривається і дані десеріалізуються з нього.
- Дані використовуються для відновлення позиції, повороту та списку підібраних предметів гравця.

Алгоритм взаємодії гравця з об'єктами реалізований через перевірку тригерних подій і виклик відповідних методів. Гравець може підбирати предмети, взаємодіяти з ритуальними об'єктами та використовувати карту.

Гравець може підбирати предмети, наближаючись до них і натискаючи кнопку взаємодії (за замовчуванням клавіша "E"). При цьому відбувається перевірка натискання клавіші та виконання відповідної дії.

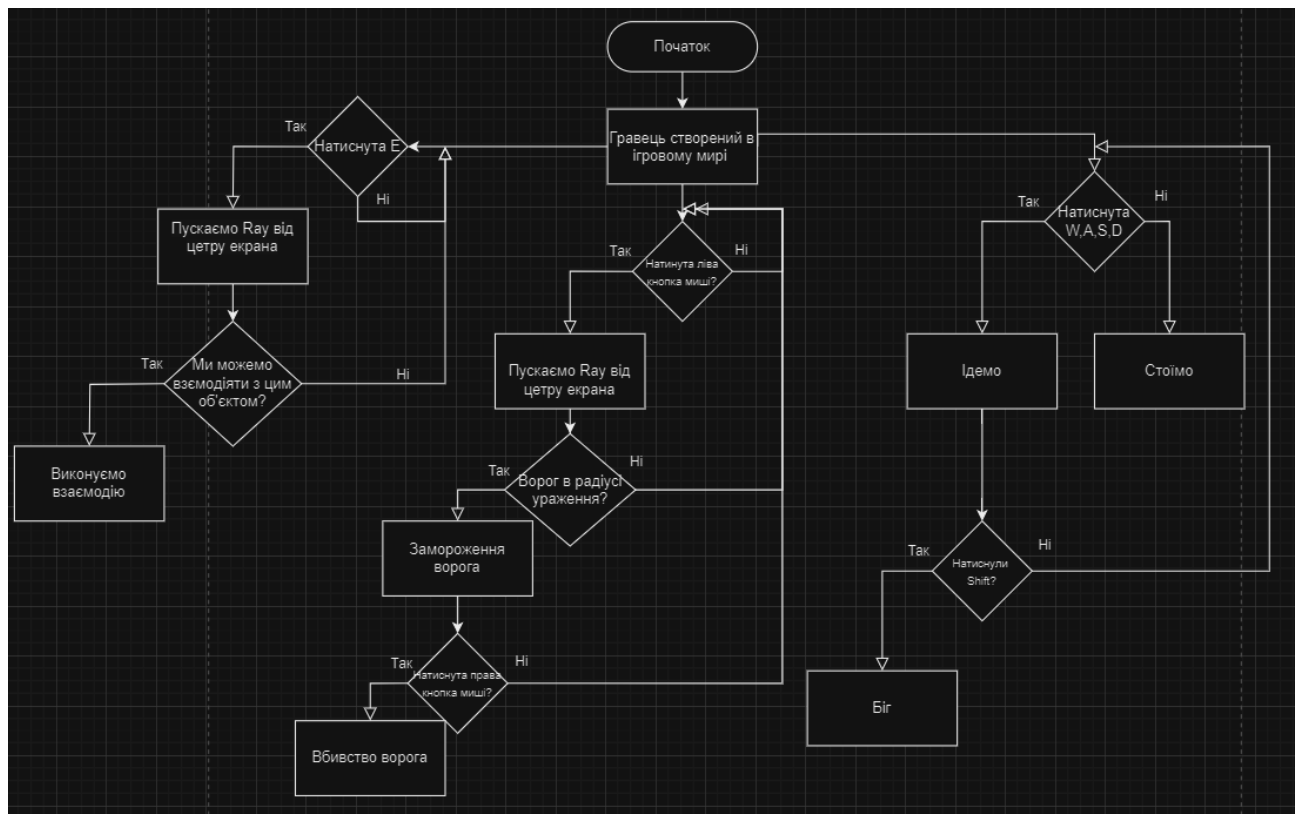


Рисунок 2.10 – Алгоритм блок-схема гравця

Коли гравець натискає клавішу "E", відправляється промінь (Ray) з камери гравця. Якщо промінь перетинає об'єкт з тегом "Item", об'єкт підбирається: він зникає зі сцени і додається до списку підібраних предметів.

Викликається метод `ItemsCollected.Invoke()`, який додає тип підбраного предмета до списку підібраних предметів гравця.

Викликається метод `spawnScreamer.Invoke()`, який може запускати скримери при підборі певних предметів.

Взаємодія з ритуальними об'єктами

Гравець може взаємодіяти з ритуальними об'єктами, такими як свічки та алтарі. Взаємодія здійснюється через перевірку натискання клавіші та виклик відповідного методу, який змінює стан об'єкта.

Коли гравець натискає клавішу "E", відправляється промінь з камери гравця.

Якщо промінь перетинає об'єкт з тегом "Candle", викликається метод `BurningCandle.Invoke()`, який змінює стан свічки (наприклад, запалює її).

Взаємодія з алтарем:

Якщо гравець знаходиться біля алтаря (визначається через перевірку тригера), натискання клавіші "E" викликає метод `PutItems.Invoke()`, який дозволяє гравцеві розмістити зібрані предмети на алтарі.

Використання карти

Гравець може активувати карту для навігації у грі. Карта відображається на весь екран і показує місцезнаходження гравця та важливі об'єкти на рівні.

Коли гравець наближається до об'єкта з тегом "Map" і натискає клавішу "E", викликається метод `CanUseMap.Invoke()`.

Карта активується та відображається на весь екран, після чого об'єкт карти зникає зі сцени.

Для розуміння логіки поведінки ворогів було створено діаграму діяльності алгоритму ігрового інтелекту. Вона відображає послідовність дій, які виконує ворог під час патрулювання та переслідування гравця.

Проектування алгоритмів роботи системи для гри в жанрі Survival Horror включає реалізацію алгоритмів переслідування ворогами, збереження та завантаження даних, а також взаємодії гравця з об'єктами. Використання окремих класів-менеджерів забезпечує модульність коду, знижує ризик помилок і спрощує підтримку та розширення функціоналу гри.

2.5 Розробка програмного коду

Розглянемо основні частини програмного коду, що реалізують ключові функціональні можливості гри. Зокрема, увага приділяється збереженню та завантаженню гри, управлінню грою та взаємодії гравця з об'єктами. Функція збереження та завантаження гри є однією з найважливіших, оскільки дозволяє гравцям зберігати свій прогрес і повертатися до гри з того місця, де вони зупинилися.

Збереження та завантаження гри (SaveLoadManager.cs)

Лістинг коду SaveLoadManager.cs:

```
public void SaveGame(GameObject player, List<ItemType> collectedItems,
List<IDScreamer> playedScreamers, bool usemap, bool showpanel)
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream fs = new FileStream(filePath, FileMode.Create);

    Save save = new Save();
    save.SavePlayer(player);
    if(collectedItems != null)
        save.SaveItems(collectedItems);
    if (playedScreamers != null)
        save.SaveIDScreamers(playedScreamers);
    save.SaveUseMap(usemap);
    save.SaveShowPanel(showpanel);
    bf.Serialize(fs, save);
    fs.Close();
    Debug.Log("save");
}

public void LoadGame()
{
    if (!File.Exists(filePath))
        return;

    GameManager _gameManager = GetComponent<GameManager>();
    BinaryFormatter bf = new BinaryFormatter();
```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

FileStream fs = new FileStream(filePath, FileMode.Open);

Save save = (Save)bf.Deserialize(fs);
fs.Close();

_gameManager.PlayerLoadData(save.PlayerData);
_gameManager.ItemsLoadData(save.ItemsData);
_gameManager.ScreamersLoadData(save.IDScreamersData);
_gameManager.UseMapLoadData(save.UseMapData);
_gameManager.StartPanelLoadData(save.ShowPanelData);
Debug.Log("Load");
}

```

У цьому фрагменті коду реалізовано методи збереження та завантаження гри. Метод SaveGame приймає параметри, що містять дані про гравця, зібрані предмети, активовані скримери, використання карти та відображення панелі. Дані серіалізуються та зберігаються у файл. Метод LoadGame завантажує ці дані з файлу та передає їх до відповідних компонентів гри для відновлення стану.

GameManager відповідає за координацію основних аспектів гри, таких як оновлення стану гравця та контроль за виконанням ігрових механік.

Лістинг коду GameManager.cs:

```

public void PlayerLoadData(Save.PlayerSaveData data)
{
    _player.transform.position = new Vector3(data.PlayerPosition.x, data.PlayerPosition.y,
data.PlayerPosition.z);
    _player.transform.rotation = new Quaternion(data.PlayerRotaion.x, data.PlayerRotaion.y,
data.PlayerRotaion.z, data.PlayerRotaion.w);
    _cameraTransform.rotation = new Quaternion(data.CameraRotaion.x, data.CameraRotaion.y,
data.CameraRotaion.z, data.CameraRotaion.w);
}

public void ItemsLoadData(Save.ItemsSaveData data)
{
    foreach (var item in data.Items)

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        // Логіка для відновлення зібраних предметів
    }
}

public void ScreammersLoadData(Save.IDScreamersSaveData data)
{
    foreach (var id in data.ID)
    {
        // Логіка для відновлення скримерів які вже були
    }
}

```

Ці методи відповідають за завантаження збережених даних і відновлення відповідних станів об'єктів у грі. PlayerLoadData встановлює позицію та обертання гравця та камери на основі збережених даних. ItemsLoadData та ScreammersLoadData відновлюють зібрані предмети та активовані скримери відповідно.

Скрипт PlayerInteraction.cs обробляє взаємодію гравця з різними об'єктами у грі, такими як підбір предметів, активація скримерів та використання карти.

Лістинг коду PlayerInteraction.cs:

```

void Update()
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        Ray ray = m_Camera.ScreenPointToRay(new Vector3(Screen.width / 2, Screen.height / 2, 0));
        RaycastHit[] hits = Physics.RaycastAll(ray, 4.5f);
        foreach (RaycastHit hit in hits)
        {
            switch (hit.collider.tag)
            {
                case "Item":
                    Destroy(hit.collider.gameObject);

                    ItemsCollected.Invoke(hit.collider.gameObject.GetComponent<CollectibleItem>().itemType);
            }
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

```

UIController.spawnScreamer.Invoke((int)hit.collider.gameObject.GetComponent<CollectibleItem>
().itemType);
    break;
case "Candle":
    BurningCandle.Invoke(hit.collider.gameObject);
    break;
case "Pedestal":
    if (StayNearPedestal)
    {
        PutItems.Invoke();
    }
    break;
case "Note":
    ShowNote.Invoke();
    break;
case "Map":
    CanUseMap.Invoke();
    Destroy(hit.collider.gameObject);
    break;
}
}
}

private void OnTriggerStay(Collider other)
{
    if (other.tag == "Pedestal_Floor")
    {
        StayNearPedestal = true;
        UIController.spawnScreamer.Invoke(6);
    }
    else
        StayNearPedestal = false;
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей фрагмент коду відповідає за обробку натискання клавіші “Е” для взаємодії з об'єктами, що знаходяться перед гравцем. За допомогою Raycast виявляються об'єкти, і в залежності від їх тегу виконується відповідна дія, така як підбір предметів, активація скрімерів або використання карти.

2.6 Реалізація програмного продукту

Процес розробки гри в жанрі Survival Horror розпочався з вивчення офіційної документації Unity. Було освоєно архітектуру рушія, принципи розробки, налаштування фізичних об'єктів, матеріалів, анімацій, звуків, побудову ігрових рівнів, графічний інтерфейс та штучний інтелект. Далі вдосконалювались знання мови C#. Було налаштовано IDE Visual Studio. Для проекту підготовлено графічні та звукові ресурси, анімації для скелетів персонажів, звуки взаємодій, скрімерів та атмосферні ефекти. Для реалізації проекту був підготовлений набір графічних і звукових ресурсів. Також було відібрано 3D моделі різних об'єктів та предметів, а для гри були підготовлені звуки взаємодії гравця з об'єктами, звуки скрімерів та атмосферні ефекти.

Реалізація почалась зі створення механіки руху та анімацій персонажа. Була реалізована система пересування, що включала в себе ходьбу та біг. Потім була реалізована механіка взаємодії з об'єктами, яка дозволяє гравцеві підбирати предмети, використовувати карту та взаємодіяти з ритуальними об'єктами.

Наступним кроком була розробка архітектури користувацького інтерфейсу. Це дозволило досить просто додавати нові механіки, створюючи графічний інтерфейс для них. В інтерфейсі було реалізовано відображення підказок для гравця щодо клавіш керування, відображення карти на весь екран та різні меню (головне меню, меню паузи, меню налаштувань).

Для асоціації персонажа з іншими предметами був розроблений інвентар. Першою створеною ігровою механікою, що використовувала такий спосіб асоціації, були підібрані предмети, які гравець міг розміщувати на алтарі або використовувати для інших дій, таких як запалювання свічок.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Окремим етапом реалізації гри стала розробка штучного інтелекту ворогів. Вороги патрулюють визначені області, виявляють гравця в радіусі видимості та переслідують його до моменту атаки. Для цього використовувалися навігаційні системи Unity, такі як NavMeshAgent.

Було реалізовано механізм збереження та завантаження стану гри. Для цього використовувалася серіалізація даних у файл за допомогою BinaryFormatter. Дані зберігалися в спеціально створеному форматі, включаючи позицію та поворот гравця, а також список підібраних предметів.

На останньому етапі було налаштовано навколишнє середовище, звукові ефекти та додано скрімери для створення атмосферного досвіду гри.

Висновки до розділу 2

В даному розділі було детально описано основні науково-технічні результати, отримані в ході розробки гри в жанрі Survival Horror. Визначено варіанти використання системи, спроектовано структуру системи з модульним підходом, розроблено модель даних для збереження стану гри та проектування алгоритмів, таких як переслідування ворогами і взаємодія гравця з об'єктами. Процес реалізації включав вивчення документації, створення механік, користувацького інтерфейсу, ігрових рівнів та підготовку графічних і звукових ресурсів, забезпечуючи комплексний підхід до створення якісного ігрового досвіду.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ОПИС СТВОРЕНОЇ ГРИ У ЖАНРІ HORROR

3.1 Опис вимог до системи, умов експлуатації та апаратного забезпечення

Для забезпечення стабільної та плавної роботи гри, яка розроблена на рушії Unity, необхідно дотримуватися певних вимог до апаратного забезпечення та операційної системи. Гра орієнтована на роботу на сучасних комп'ютерах з операційною системою Windows 10 x64 та підтримкою DirectX 11. Це гарантує оптимальну сумісність з програмним забезпеченням та забезпечує належну продуктивність.

Мінімальні системні вимоги включають процесор Intel Core i5 або еквівалентний від AMD з тактовою частотою не менше 2.5 ГГц. Це необхідно для забезпечення плавної роботи основних ігрових процесів, включаючи обробку анімацій, взаємодії з об'єктами та ворогами, а також для стабільного функціонування штучного інтелекту ворогів.

Оперативна пам'ять повинна бути не менше 8 ГБ. Це дозволяє гри завантажувати необхідні ресурси, такі як текстури, звукові файли та інші медіа-ресурси, без значних затримок та забезпечує стабільну частоту кадрів навіть у найінтенсивніших сценах.

Відеокарта повинна підтримувати DirectX 11 і мати не менше 2 ГБ відеопам'яті. Це необхідно для коректного відображення високоякісних графічних ефектів, таких як тіні, освітлення та текстури високої роздільної здатності, які використовуються у грі для створення атмосфери Survival Horror. Рекомендовані моделі відеокарт включають Nvidia GTX 1050 або AMD Radeon RX 560.

Для гри також необхідно мати не менше 5 ГБ вільного місця на жорсткому диску для встановлення гри та зберігання даних користувача, таких як збереження прогресу та налаштування. Рекомендується використовувати SSD для скорочення часу завантаження ігрових рівнів та забезпечення більш плавного ігрового процесу.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

Крім того, для комфортної гри необхідно мати сучасний дисплей з роздільною здатністю не менше 1920x1080 пікселів. Це дозволить гравцю насолодитися усіма візуальними ефектами та деталями гри. Звукова карта, яка підтримує 5.1 або 7.1 аудіо, також є бажаною для повного занурення в атмосферу гри, особливо для відтворення навколишніх звуків і скрімерів.

Інші системні вимоги включають наявність клавіатури та миші для управління персонажем та взаємодії з інтерфейсом гри. Інтернет-з'єднання не є обов'язковим, оскільки гра розрахована на однокористувацький режим, але може бути корисним для отримання оновлень та додаткового контенту.

Дотримання цих вимог забезпечить оптимальну продуктивність гри та комфортний ігровий досвід, дозволяючи гравцям повністю зануритися в атмосферу Survival Horror.

3.2 Опис та обґрунтування вибору засобів реалізації

Для реалізації проекту було обрано декілька ключових інструментів та технологій, кожен з яких відіграє важливу роль у створенні та підтримці гри. Вибір цих засобів базувався на їхніх функціональних можливостях, зручності використання та сумісності з іншими компонентами проекту.

Ігровий рушій Unity був обраний за його потужні інструменти для розробки 3D ігор. Unity надає широкий спектр можливостей для створення інтерактивних середовищ, управління фізикою, анімаціями та звуковими ефектами. Однією з ключових переваг Unity є велика спільнота розробників та наявність багатой документації, що значно полегшує вирішення технічних проблем та обмін досвідом. Unity також підтримує багатоплатформенність, що дозволяє розробляти гру одночасно для декількох платформ.

Мова програмування C# була обрана для написання скриптів у Unity. C# забезпечує високу продуктивність, простоту розробки та хорошу інтеграцію з Unity. Вона підтримує об'єктно-орієнтоване програмування, що дозволяє легко створювати та підтримувати складні структури даних та алгоритми. Завдяки

своїй синтаксичній зрозумілості та широким можливостям, C# стала ідеальним вибором для реалізації логіки гри.

Для створення та анімації 3D моделей було використано Blender. Blender є потужним і безкоштовним інструментом для моделювання, текстурювання, рігінгу та анімації. Він підтримує широкий спектр форматів файлів та надає можливості для створення високоякісних графічних ресурсів. Blender також має активну спільноту та багатий набір навчальних матеріалів, що полегшує освоєння цього інструменту.

Visual Studio було вибрано як основне середовище розробки завдяки своїм можливостям інтеграції з Unity та зручному інтерфейсу для написання коду. Visual Studio пропонує зручні інструменти для налагодження, тестування та профілювання коду, що значно спрощує процес розробки та підвищує ефективність роботи. Інтеграція з Unity дозволяє швидко переходити від написання коду до тестування та налагодження без необхідності використання окремих інструментів.

Комбінація цих інструментів забезпечила високу якість розробки та ефективність роботи над проектом. Кожен з них відігравав важливу роль у різних аспектах створення гри, від моделювання та анімації до написання коду та редагування звукових ефектів. Це дозволило створити цілісний та добре оптимізований продукт, що відповідає сучасним стандартам якості та забезпечує захоплюючий ігровий досвід.

3.3 Структура інтерфейсу та порядок роботи з програмним продуктом

Для встановлення гри користувачу необхідно завантажити на комп'ютер архів з грою, розархівувати його та запустити файл «Exile.exe». При цьому слід зазначити, що комп'ютер має відповідати всім вимогам до апаратного забезпечення, зазначеного в розділі 1.3, на ньому також має бути встановлена ОС Windows 10 x64 та DirectX 11.

Після запуску гри, користувач потрапляє в головне меню, де він може

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

відразу перейти до основного геймплею, натиснувши кнопку «Грати». Також на головному меню можна перейти до налаштувань та налаштувати чутливість миші та гучність.

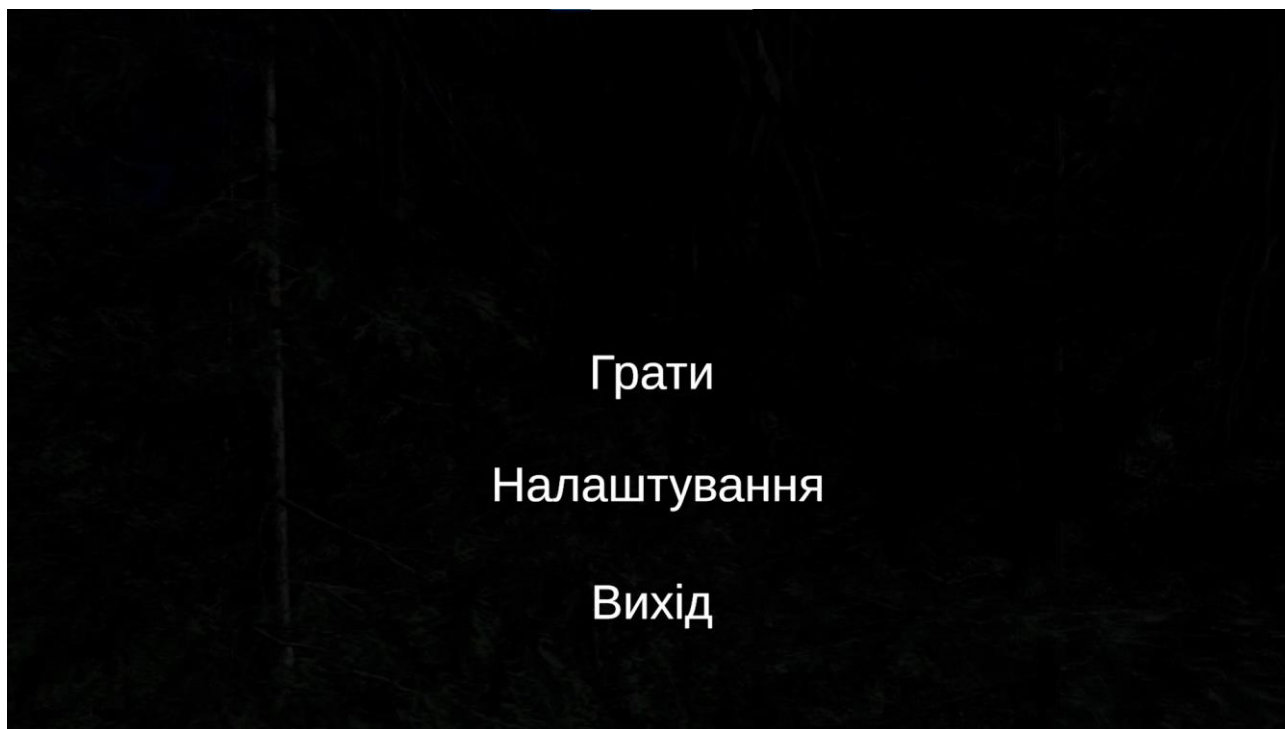


Рисунок 3.1 – Головне меню

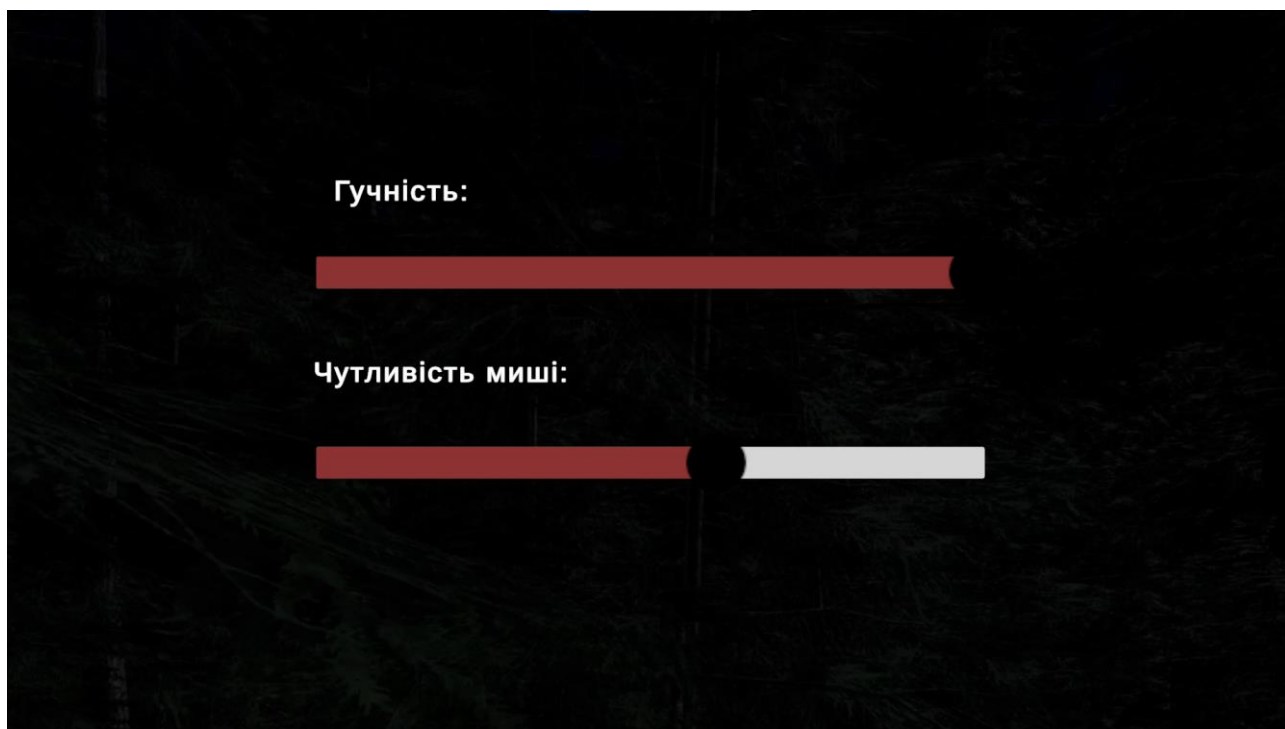


Рисунок 3.2 – Панель налаштування

Після натискання кнопки «Грати» відбувається завантаження ігрового рівня. Гравець починає гру на стартовій позиції, де йому надаються інформація про те за кого граю гравець.

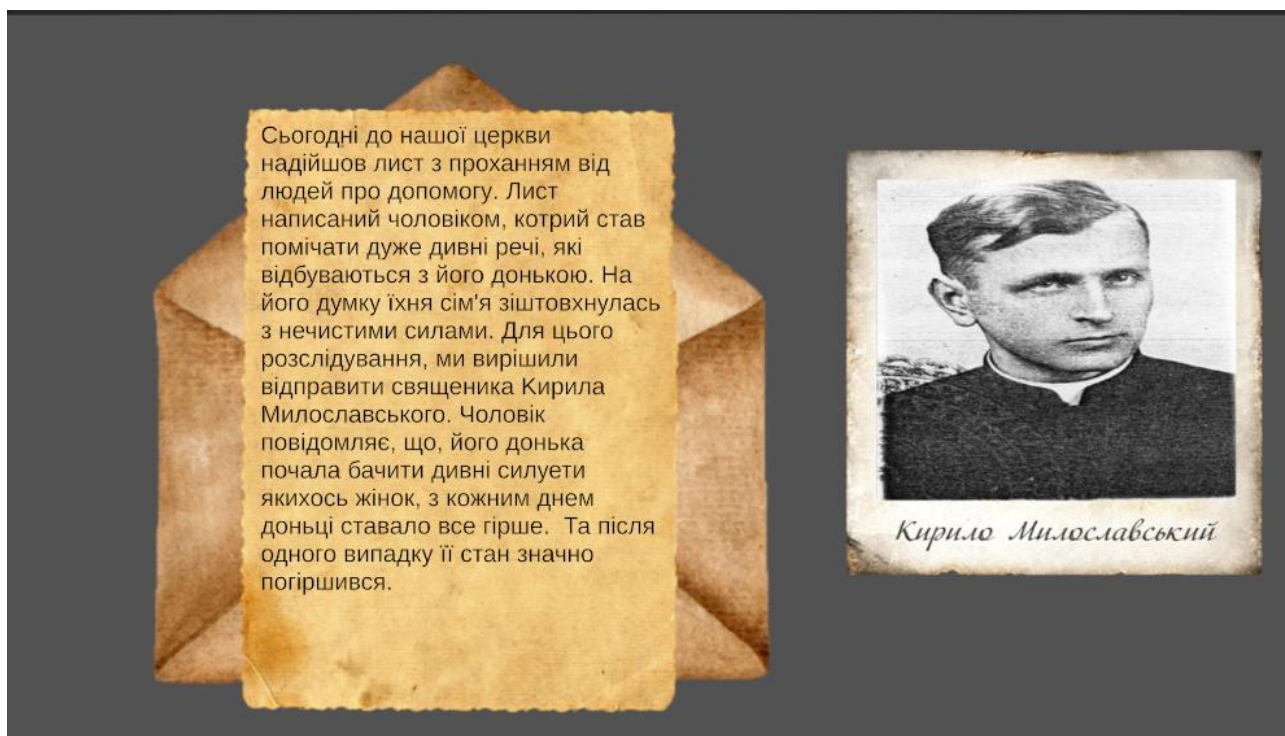


Рисунок 3.3 – Стартова інформація

Завданням гравця є вижити в лісі, зібрати необхідні предмети та провести обряд для вигнання духу.

Під час гри гравець може взаємодіяти з різними об'єктами, збирати предмети, заморожувати та вбивати ворогів. У разі смерті персонажа або виконання завдання гравець отримує відповідне повідомлення з можливістю повернутися до головного меню або почати гру з останньої контрольної точки.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47



Рисунок 3.4 – Повідомлення про смерть гравця

Гравець може поставити гру на паузу в будь-який момент, натиснувши клавішу «Escape». При цьому на екрані з'явиться меню паузи, де є можливість продовжити гру, змінити налаштування або вийти до головного меню.

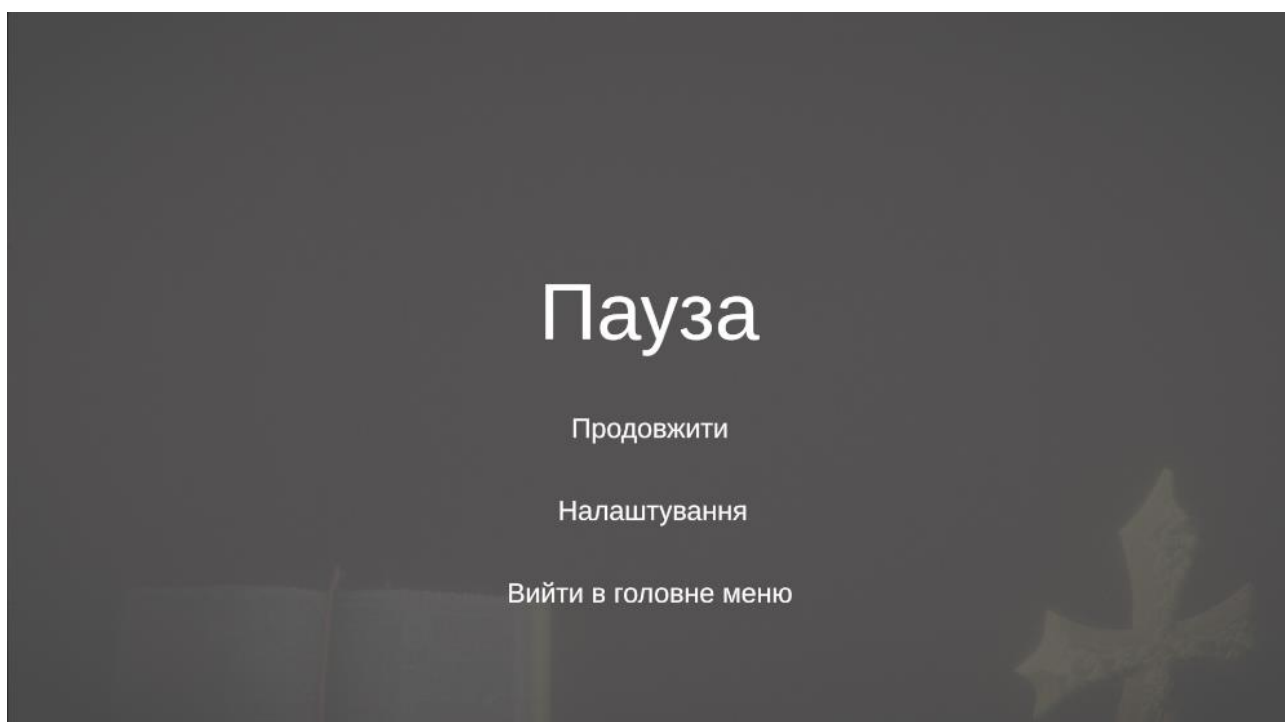


Рисунок 3.5 – Меню паузи

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

У грі також передбачені підказки щодо управління, які відображаються на екрані під час гри. Наприклад, підказка про використання лівої кнопки миші для заморожування ворогів та клавіші «Е» для підбору предметів. Це допомагає гравцеві швидко орієнтуватися в ігровому процесі та ефективно використовувати доступні можливості.



Рисунок 3.6 – Підказки управління

Таблиця 3.1

Керування	
Дія	Клавіша
Рухатись вперед	W
Рухатись назад	S
Рухатись вліво	A
Рухатись вправо	D
Бігти	Left Shift
Відкрити меню паузи	ESC
Взаємодіяти	E
Відкрити мапу	M
Заморозити	Left Mouse Button
Вигнати	Right Mouse Button

3.4 Тестування роботи програмного продукту

Тестування гри в жанрі Survival Horror проводилося для забезпечення її коректної роботи, продуктивності та стабільності. Тестування включало кілька ключових етапів і різні типи тестів, спрямованих на виявлення можливих проблем та їх виправлення.

Функціональне тестування було одним з основних етапів. Воно включало перевірку всіх основних механік гри, таких як рух персонажа, взаємодія з об'єктами, атака ворогів, заморожування ворогів та підбір предметів. Кожна функціональна частина гри тестувалася на предмет правильного виконання запланованих дій.

Таблиця 3.2

Функціональні тести		
Дія	Очікувана реакція	Отримана реакція
Гравець натискає клавіші WASD для переміщення персонажа вперед, назад, вліво та вправо.	Персонаж починає рухатися відповідно до натисканих клавіш: вперед (W), назад (S), вліво (A), вправо (D).	Персонаж правильно рухається у всіх зазначених напрямках, відповідно до натискань клавіш WASD.
Гравець натискає клавішу E при наведенні прицілу на об'єкт з тегом "Item".	Об'єкт зникає з ігрової сцени, а тип предмета додається до списку підібраних предметів.	Об'єкт правильно зникає з сцени, а тип предмета успішно додається до списку підібраних предметів.
Гравець натискає праву кнопку миші для заморожування ворога, а потім ліву кнопку миші для його вбивства.	Ворог зупиняється і не рухається при натисканні правої кнопки миші, а після натискання лівої кнопки миші ворог зникає з ігрової сцени.	Ворог правильно зупиняється при заморожуванні та зникає після атаки, як очікувалося.
Гравець підходить до алтаря та натискає клавішу E для взаємодії.	Гравець може розмістити на алтарі підібрані предмети, якщо вони є в інвентарі.	Гравець успішно розміщує предмети на алтарі, якщо вони були підібрані раніше.
Гравець натискає клавішу E при наведенні прицілу на об'єкт з тегом "Map".	Карта зникає з ігрової сцени, і гравець отримує можливість використовувати карту через інтерфейс..	Карта правильно зникає з сцени, і гравець отримує доступ до карти через інтерфейс, як очікувалося.
Гравець натискає клавішу E при наведенні прицілу на об'єкт з тегом "Candle".	Свічка запалюється, якщо у гравця є підібрані сірники.	Свічка успішно запалюється, якщо у гравця є сірники, як очікувалося.
Гравець підходить до певного тригера на карті.	З'являється скример, який тимчасово лякає гравця і супроводжується звуковими та візуальними ефектами.	Скример правильно з'являється у визначеному місці, створюючи очікуваний ефект лякування.

Гравець натискає клавішу Escape під час гри.	Гра зупиняється, і на екрані з'являється меню паузи, де гравець може обрати продовження гри, змінити налаштування або повернутися до головного меню.	Меню паузи з'являється як очікувалося, дозволяючи гравцю обрати відповідні опції.
--	--	---

Для тестування штучного інтелекту ворогів були проведені численні тести, щоб переконатися, що вороги правильно патрулюють, виявляють та переслідують гравця. Перевірялося, чи правильно працюють алгоритми переслідування, чи не застрягають вороги в перешкодах, чи адекватно реагують на дії гравця. Було перевірено різні сценарії поведінки ворогів, щоб забезпечити їх природну та передбачувану поведінку.

Тестування на продуктивність включало оцінку швидкості роботи гри. Оцінювалися часи завантаження, частота кадрів, плавність анімацій та загальна продуктивність. Було виявлено та оптимізовано кілька вузьких місць, що дозволило покращити продуктивність гри навіть на менш потужних системах.

Тестування проводилось в системі з наступними характеристиками:

- ОС: Windows 10 x64
- Процесор: Intel Core i5-9300H
- Оперативна пам'ять: 8Гб
- Відеокарта: Geforce GTX 1050
- DirectX: 12

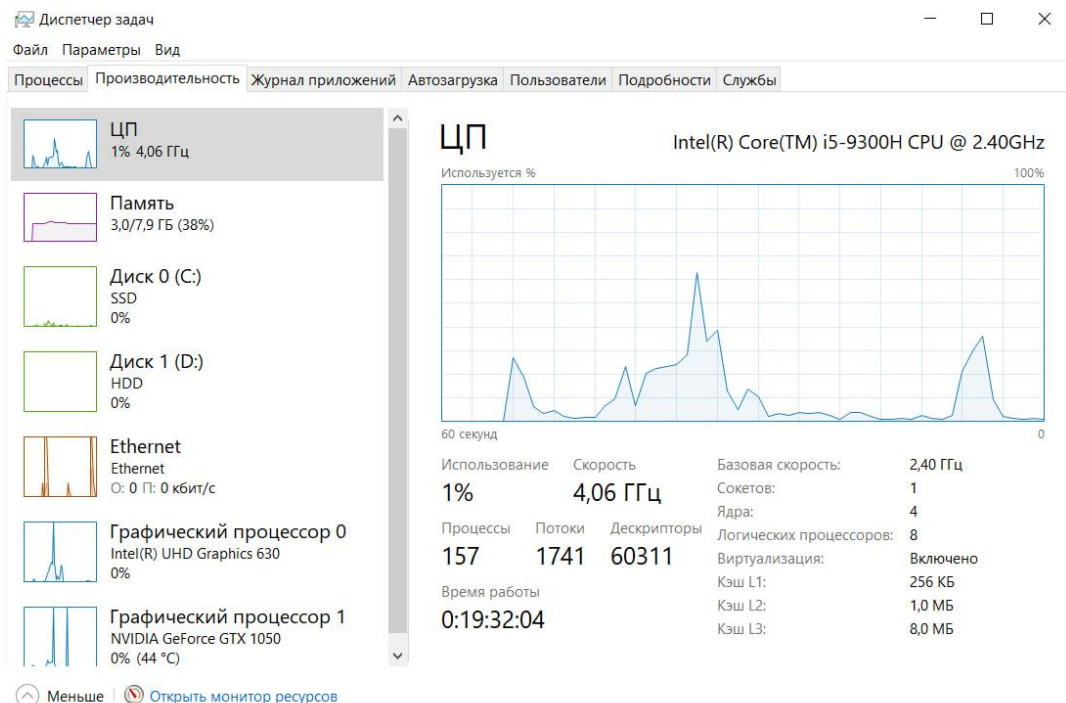


Рисунок 3.7 – Графік споживання ресурсів системи до запуску гри

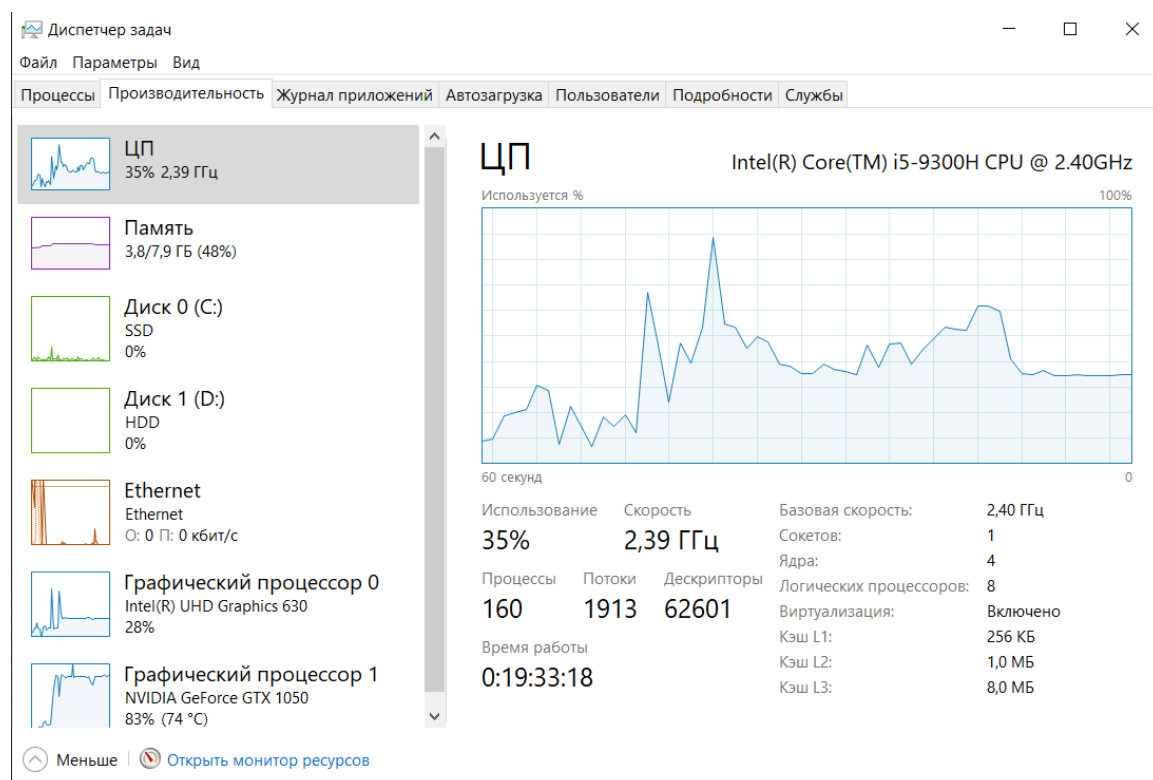


Рисунок 3.8 – Графік споживання ресурсів системи під запуску гри

Результати навантажувального тестування показали задовільні показники. Частота кадрів була стабільною на рівні 60 кадрів на секунду завдяки встановленому обмеженню. Завантаження процесора коливалося в межах 30-35% з незначними піками. Споживання оперативної пам'яті було на рівні 800 МБ. Ці результати свідчать про те, що гра буде працювати стабільно на комп'ютері, який відповідає мінімальним вимогам, забезпечуючи частоту кадрів не менше 60 кадрів на секунду.

Для забезпечення стабільності гри було проведено тривале тестування, під час якого перевірялися різні сценарії гри, збереження та завантаження прогресу. Перевірялося, чи коректно працює система збереження, чи правильно відновлюється стан гри після завантаження. Тестувалися різні ситуації, які могли б призвести до збоїв, такі як несподіване вимкнення гри або перезавантаження системи під час гри. Було виявлено та виправлено кілька помилок, що підвищило загальну стабільність гри.

Під час тестування особлива увага приділялася графічному інтерфейсу. Перевірялося, чи всі елементи інтерфейсу правильно відображаються на екрані, чи легко вони читаються, чи коректно працюють підказки та індикатори. Інтерфейс тестувався на різних роздільних здатностях екрана, щоб переконатися, що він адаптивний та зручний для користувача на будь-якому пристрої.

Також проводилося тестування звукових ефектів та музики. Перевірялося, чи правильно відтворюються звукові ефекти під час взаємодії гравця з об'єктами, чи коректно працюють звукові індикатори наближення ворогів, чи підходить музика до загальної атмосфери гри. Було враховано різні звукові налаштування та перевірено їхній вплив на ігровий досвід.

Результати тестування показали, що гра працює стабільно та продуктивно на більшості сучасних комп'ютерів, відповідає всім функціональним вимогам та надає гравцям захоплюючий ігровий досвід. Було виявлено та виправлено кілька помилок, що значно підвищило якість гри. Завдяки ретельному тестуванню вдалося забезпечити високу надійність та задоволення гравців.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

Висновки до розділу 3

У третьому розділі було проаналізовано та описано вимоги до системи і апаратного забезпечення, обґрунтовано вибір засобів реалізації, розроблено основні проектні рішення, описано структуру інтерфейсу та порядок роботи з грою, а також проведено тестування системи. Результати тестування підтвердили стабільну роботу гри на відповідному обладнанні та забезпечили плавний ігровий процес, підтвердивши відповідність усім заявленим вимогам.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У межах дипломної роботи бакалавра було проведено повний цикл розробки відеоігри в жанрі survival horror. Розробка базувалася на ігровому рушії Unity з використанням мови програмування C#.

У першому розділі було досліджено специфіку жанру survival horror, зокрема елементи виживання, напружену атмосферу та взаємодію з ворогами. Сформульовано і поставлено задачу створення програмного продукту, визначено основні критерії оцінки ігор цього жанру. Проведено порівняння аналогічних програмних продуктів для виявлення найкращих практик та уникнення типових помилок. Перший розділ завершився вибором інструментарію для розробки гри, таких як Unity, Blender для 3D-моделей та Visual Studio для написання коду. Також визначено мінімальні системні вимоги для стабільної роботи гри.

Другий розділ було присвячено проектуванню гри. Виходячи із поставлених завдань, визначено основні варіанти використання програми, такі як взаємодія з об'єктами, атака ворогів, збір предметів тощо. Архітектура програмного забезпечення була спроектована та описана у вигляді діаграм, що містять модульну структуру з чітким розподілом обов'язків між класами. Для зберігання стану гри була розроблена модель даних, що дозволяє реалізувати функцію збереження і завантаження прогресу. Також наведено опис алгоритму штучного інтелекту ворогів, які патрулюють територію і переслідують гравця у разі виявлення.

У третьому розділі детально описано процес реалізації гри. Перед початком розробки було детально вивчено офіційну документацію Unity і освоєно основні принципи роботи з рушієм, фізичними об'єктами, матеріалами, анімаціями та налаштуваннями звуку. Крім того, було покращено знання мови програмування C# з особливою увагою до об'єктно-орієнтованого підходу. Були підготовлені графічні і звукові ресурси, створені 3D-моделі персонажів, ворогів і об'єктів навколишнього середовища, налаштовані анімації, підготовлені звукові

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

ефекти і розроблений користувацький інтерфейс, що включає головне меню, меню паузи та інші елементи, необхідні для зручної взаємодії з грою. Тестування гри показало стабільну роботу при правильній конфігурації обладнання, а функціональні тести підтвердили точність реалізації основних механік.

Результатом цієї роботи стала гра в жанрі survival horror, що відповідає всім поставленим вимогам і цілям. Гравці повинні не лише виживати, але й вирішувати різні завдання, взаємодіяти з об'єктами та уникати ворогів. Проведені тести підтвердили стабільність і ефективність розробленого рішення, забезпечуючи високий рівень задоволення користувачів.

Таким чином, розроблена гра повністю відповідає поставленим цілям і стандартам, надаючи гравцям захоплюючий досвід у жанрі survival horror. Вона не тільки розважає, але й надає конкретні завдання і навички виживання, роблячи значний внесок у жанр і обіцяючи подальший розвиток.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація до ігрового рушія Unity – Unity Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unity.com>
2. C# Programming Guide - Microsoft Corporation [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/csharp/>
3. Blender Documentation - Blender Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.blender.org>
4. WHAT IS UML – Dr. Jon Siegel [Електронний ресурс] – Режим доступу до ресурсу: <https://www.uml.org/what-is-uml.htm>.
5. Blender Guru – Andrew Price [Електронний ресурс] – Режим доступу до ресурсу: <https://www.blenderguru.com>.
6. GameDev.net – GameDev.net [Електронний ресурс] – Режим доступу до ресурсу: <https://www.gamedev.net>
7. Stack Overflow – Stack Exchange Inc [Електронний ресурс] – Режим доступу до ресурсу: <https://stackoverflow.com>
8. Gamasutra – Informa PLC [Електронний ресурс] – Режим доступу до ресурсу: <https://www.gamedeveloper.com>
9. Ray Wenderlich – Razeware LLC [Електронний ресурс] – Режим доступу до ресурсу: <https://www.kodeco.com>
10. Game Development Stack Exchange – Stack Exchange Inc. [Електронний ресурс] – Режим доступу до ресурсу: <https://gamedev.stackexchange.com>
11. Unity Forum – Unity Technologies [Електронний ресурс] – Режим доступу до ресурсу: <https://forum.unity.com>
12. Шелла Д. The Art of Game Design: A Book of Lenses / Джессі Шелл., 2008.
13. Найстром Р. Game Programming Patterns / Роберт Найстром., 2014
14. Вагнер Б. Effective C#: 50 Specific Ways to Improve Your C# / Білл Вагнер., 2004
15. Рабін С. Introduction to Game Development / Стів Рабін., 2005
16. Бакленд М. Programming Game AI by Example / Мет Бакленд., 2004

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

17. Хокінг Д. Unity in Action: Multiplatform Game Development in C# / Джозеф Хокінг., 2015
- 18.Гріг Ш. Blender 3D By Example / Ш. Гріг, О. Башлер., 2019.
19. Бірн Д. Digital Lighting and Rendering / Джеремі Бірн., 2000.
20. Грегор Д. Game Engine Architecture / Джейсон Грегор., 2009

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу SaveLoadManager.cs

```

using System.Collections.Generic;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using UnityEngine;

public class SaveLoadManager : MonoBehaviour
{
    string filePath;

    private void Awake()
    {
        filePath = Application.persistentDataPath + "/save.gamesave";
    }

    public void SaveGame(GameObject player, List<ItemType> collectedItems,
List<IDScreamer> playedScreamers, bool usemap, bool showpanel)
    {
        BinaryFormatter bf = new BinaryFormatter();
        FileStream fs = new FileStream(filePath, FileMode.Create);

        Save save = new Save();
        save.SavePlayer(player);
        if(collectedItems != null)
            save.SaveItems(collectedItems);
        if (playedScreamers != null)
            save.SaveIDScreamers(playedScreamers);
        save.SaveUseMap(usemap);
        save.SaveShowPanel(showpanel);
        bf.Serialize(fs, save);
        fs.Close();
        Debug.Log("save");

    }

    public void LoadGame()
    {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!File.Exists(filePath))
            return;
        GameManager _gameManager = GetComponent<GameManager>();
        BinaryFormatter bf = new BinaryFormatter();
        FileStream fs = new FileStream(filePath, FileMode.Open);

        Save save = (Save)bf.Deserialize(fs);
        fs.Close();

        _gameManager.PlayerLoadData(save.PlayerData);
        _gameManager.ItemsLoadData(save.ItemsData);
        _gameManager.ScreamersLoadData(save.IDScreamersData);
        _gameManager.UseMapLoadData(save.UseMapData);
        _gameManager.StartPanelLoadData(save.ShowPanelData);
        Debug.Log("Load");
    }
}

```

```

[System.Serializable]
public class Save
{
    [System.Serializable]
    public struct Vec3
    {
        public float x, y, z;

        public Vec3(float x, float y, float z)
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }
    }

    [System.Serializable]
    public struct Quat
    {
        public float x, y, z, w;

        public Quat(float x, float y, float z, float w)
        {
            this.x = x;
            this.y = y;

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        this.z = z;
        this.w = w;
    }
}

[System.Serializable]
public struct PlayerSaveData
{
    public Vec3 PlayerPosition;
    public Quat PlayerRotaion;
    public Quat CameraRotaion;

    public PlayerSaveData(Vec3 playerPos, Quat playerRot, Quat cameraRot)
    {
        PlayerPosition = playerPos;
        PlayerRotaion = playerRot;
        CameraRotaion = cameraRot;
    }
}

public PlayerSaveData PlayerData = new PlayerSaveData();
public void SavePlayer(GameObject player)
{
    PlayerMovement _player = player.GetComponent<PlayerMovement>();
    Vec3 playerPos = new Vec3(player.transform.position.x,
player.transform.position.y, player.transform.position.z);
    Quat playerRot = new Quat(player.transform.rotation.x,
player.transform.rotation.y, player.transform.rotation.z,
player.transform.rotation.w);
    Quat cameraRot = new Quat(_player.cameraTransform.rotation.x,
_player.cameraTransform.rotation.y, _player.cameraTransform.rotation.z,
_player.cameraTransform.rotation.w);
    PlayerData = new PlayerSaveData(playerPos, playerRot, cameraRot);
}

[System.Serializable]
public struct ItemsSaveData
{
    public List<ItemType> Items;
    public ItemsSaveData(List<ItemType> items)
    {
        Items = items;
    }
}

```

```

}

public ItemsSaveData ItemsData = new ItemsSaveData();

public void SaveItems(List<ItemType> CollectedItems)
{
    List<ItemType> items = CollectedItems;
    ItemsData = new ItemsSaveData(items);
}

[System.Serializable]
public struct IDScreamersSaveData
{
    public List<IDScreamer> ID;
    public IDScreamersSaveData(List<IDScreamer> id)
    {
        ID = id;
    }
}

public IDScreamersSaveData IDScreamersData = new IDScreamersSaveData();

public void SaveIDScreamers(List<IDScreamer> playedScreamers)
{
    List<IDScreamer> id = playedScreamers;
    IDScreamersData = new IDScreamersSaveData(id);
}

[System.Serializable]
public struct UseMapSaveData
{
    public bool Usemap;
    public UseMapSaveData(bool usemap)
    {
        Usemap = usemap;
    }
}

public UseMapSaveData UseMapData = new UseMapSaveData();

public void SaveUseMap(bool usemap)
{
    bool UseMap = usemap;
    UseMapData = new UseMapSaveData(UseMap);
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

    }

[System.Serializable]
public struct ShowStartPanelData
{
    public bool ShowPanel;
    public ShowStartPanelData(bool showpanel)
    {
        ShowPanel = showpanel;
    }
}

public ShowStartPanelData ShowPanelData = new ShowStartPanelData();

public void SaveShowPanel(bool showpanel)
{
    bool ShowPanel = showpanel;
    ShowPanelData = new ShowStartPanelData(ShowPanel);
}

}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу GameManager.cs

```

using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using UnityEngine.Events;
public class GameManager : MonoBehaviour
{
    [SerializeField] private GameObject _player;
    [SerializeField] private AudioSource _audioSource;
    [SerializeField] private AudioClip _soundBurningCandle;
    [SerializeField] private AudioClip _soundItemCollected;
    [SerializeField] private Object _fireExplosion;
    [SerializeField] private List<GameObject> _collectibleItems;
    [SerializeField] private List<GameObject> _itemsInPedestal;
    [SerializeField] private List<GameObject> _candleBurning;
    [SerializeField] private List<GameObject> _screamerPrefabs;

    public bool UseMap = false;
    public bool Showstartpanel = false;

    private SaveLoadManager _saveLoadManager;

    private List<ItemType> _collectedItems = new List<ItemType>();
    private List<IDScreamer> _idPlayedScreamers = new List<IDScreamer>();

    public static UnityEvent<bool> ShowFinalBlackImage = new UnityEvent<bool>();

    private void Awake()
    {
        PlayerInteraction.BurningCandle.AddListener(BurningCandle);
        PlayerInteraction.ItemsCollected.AddListener(SaveCollectedItems);
        PlayerInteraction.PutItems.AddListener(PutItemsInPedestal);
        PlayerInteraction.CanUseMap.AddListener(CanUseMap);
        UIController.spawnScreamer.AddListener(SpawnScreamer);
        _saveLoadManager = GetComponent<SaveLoadManager>();
        _saveLoadManager.LoadGame();
    }
    private void CanUseMap()
    {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _audioSource.PlayOneShot(_soundItemCollected);
        UseMap = true;
        AudioManager.ActionPlayVoiceOver.Invoke("4");
    }

    private void SaveCollectedItems(ItemType collectedItem)
    {
        _audioSource.PlayOneShot(_soundItemCollected);
        _collectedItems.Add(collectedItem);
        if(collectedItem != ItemType.MatchesBox)
            IsLastItem();
    }

    public void SaveCurrentProgress()
    {
        if (_saveLoadManager != null) {
            _saveLoadManager.SaveGame(_player, _collectedItems,
            _idPlayedScreamers, UseMap, Showstartpanel);
        }
    }

    public void PlayerLoadData(Save.PlayerSaveData save)
    {
        _player.GetComponent<PlayerMovement>().LoadData(save);
    }

    public void ScreamersLoadData(Save.IDScreamersSaveData IDScreamersData)
    {
        _idPlayedScreamers = IDScreamersData.ID;
    }

    public void UseMapLoadData(Save.UseMapSaveData useMapSaveData)
    {
        UseMap = useMapSaveData.Usemap;
    }

    public void StartPanelLoadData(Save.ShowStartPanelData showpaneldata)
    {
        Showstartpanel = showpaneldata.ShowPanel;
    }

    public void ItemsLoadData(Save.ItemsSaveData itemsData)
    {
        foreach (var collectibleItem in _collectibleItems)
        {
            if (itemsData.Items != null)

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            _collectedItems = itemsData.Items;
            if (_collectedItems.Where(i => i ==
collectibleItem.GetComponent<CollectibleItem>().itemType).Count() == 0)
            {
                Instantiate(collectibleItem);

            }
        }
        else
        {
            Instantiate(collectibleItem);
        }
    }

}

private void PutItemsInPedestal()
{
    List<GameObject> itemsToRemove = new List<GameObject>();

    foreach (var item in _itemsInPedestal)
    {
        ItemType currentItemType =
item.GetComponent<CollectibleItem>().itemType;

        if (_collectedItems.Contains(currentItemType))
        {
            _audioSource.PlayOneShot(_soundItemCollected);
            Instantiate(item);

            itemsToRemove.Add(item);
        }
    }
    foreach (var item in itemsToRemove)
    {
        _itemsInPedestal.Remove(item);
    }
    SaveCurrentProgress();
    GameIsOver();
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    private void GameIsOver()
    {
        if (_candleBurning.Count == 6 && _itemsInPedestal.Count == 0)
        {
            ShowFinalBlackImage.Invoke(true);
        }
    }

    private void BurningCandle(GameObject candle)
    {
        if (_collectedItems.Contains(ItemType.MatchesBox))
        {
            _audioSource.PlayOneShot(_soundBurningCandle);
            GameObject explosionRef = (GameObject)Instantiate(_fireExplosion);
            explosionRef.transform.position = new
            Vector3(candle.transform.position.x, candle.transform.position.y,
            candle.transform.position.z);
            candle.tag = "Candle_Burning";
            _candleBurning.Add(candle);
            GameIsOver();
        }
    }

    public void SpawnScreamer(int IDscreamer)
    {
        foreach (GameObject screamer in _screamerPrefabs)
        {
            IDScreamer idScreamer =
            screamer.GetComponent<IDScreamers>().Idscreamer;
            if (!_idPlayedScreamers.Contains(idScreamer) && (int)idScreamer ==
            IDscreamer)
            {
                Instantiate(screamer);
                _idPlayedScreamers.Add(idScreamer);
                SaveCurrentProgress();
            }
        }
    }

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public bool WasActivatedSixScreamer()
    {
        return _idPlayedScreamers.Contains(IDScreamer.Screamer6);
    }

    public void IsLastItem()
    {
        if(_collectedItems.Contains(ItemType.Icon) &&
        _collectedItems.Contains(ItemType.Goblet) &&
        _collectedItems.Contains(ItemType.HolyWater))
        {
            AudioManager.ActionPlayVoiceOver.Invoke("5");
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу PlayerInteraction.cs

```

using UnityEngine;
using UnityEngine.Events;

public class PlayerInteraction : MonoBehaviour
{
    [SerializeField] private Camera m_Camera;

    public static UnityEvent<GameObject> BurningCandle = new
    UnityEvent<GameObject>();
    public static UnityEvent<ItemType> ItemsCollected = new
    UnityEvent<ItemType>();
    public static UnityEvent PutItems = new UnityEvent();
    public static UnityEvent ShowNote = new UnityEvent();
    public static UnityEvent CanUseMap = new UnityEvent();
    private bool StayNearPedestal;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            Ray ray = m_Camera.ScreenPointToRay(new Vector3(Screen.width / 2,
            Screen.height / 2, 0));
            RaycastHit[] hits = Physics.RaycastAll(ray, 4.5f);
            foreach (RaycastHit hit in hits)
            {
                switch (hit.collider.tag)
                {
                    case "Item":
                        Destroy(hit.collider.gameObject);

                        ItemsCollected.Invoke(hit.collider.gameObject.GetComponent<CollectibleItem>().item
                        Type);

                        UIController.spawnScreamer.Invoke((int)hit.collider.gameObject.GetComponent<Collec
                        tibleItem>().itemType);
                        break;
                    case "Candle":
                        BurningCandle.Invoke(hit.collider.gameObject);
                        break;
                    case "Pedestal":
                        if (StayNearPedestal)
                        {
                            PutItems.Invoke();
                        }
                        break;
                    case "Note":
                        ShowNote.Invoke();
                        break;
                    case "Map":
                        CanUseMap.Invoke();

                        Destroy(hit.collider.gameObject);
                        break;
                }
            }
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        }
    }
}
private void OnTriggerStay(Collider other)
{
    if (other.tag == "Pedestal_Floor")
    {
        StayNearPedestal = true;
        UIController.spawnScreamer.Invoke(6);
    }
    else
        StayNearPedestal = false;
}
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		