

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 комп'ютерної гри у жанрі horror у
віртуальній реальності»**

Виконав студент групи КНк-22-1
КУЛЄШОВ Дмитро Андрійович

Керівник роботи:
ГРАФ Марина Сергіївна

Рецензент:
ПЕТРОСЯН Арсен Русланович

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: **КУЛЄШОВ Дмитро Андрійович**
Керівник роботи: **ГРАФ Марина Сергіївна**
Тема роботи: **«Розробка комп'ютерної гри у жанрі horror у віртуальній реальності»**,
затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**
Термін здачі закінченої роботи: **«05» червня 2024 р.**
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):
об'єктом дослідження є процес розробки гри у віртуальній реальності. Предметом дослідження є методи та інструменти розробки virtual reality ігор у жанрі horror.
Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Граф М.С	10.04.2024	05.06.2024
2	Граф М.С	01.05.2024	05.06.2024
3	Граф М.С	20.05.2024	05.06.2024

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2024 – 5 березня 2024	Виконано
2	Проектування структури системи	6 березня 2024 – 25 березня 2024	Виконано
3	Написання програмного коду	26 березня 2024 – 26 квітня 2024	Виконано
4	Тестування системи	27 квітня 2024 – 14 травня 2024	Виконано
5	Оформлення пояснювальної записки	15 травня 2024 – 1 червня 2024	Виконано
6	Попередній захист	10 червня 2024	Виконано

Здобувач вищої освіти **Дмитро КУЛЄШОВ**
Керівник **Марина ГРАФ**

РЕФЕРАТ

Ця кваліфікаційна робота присвячена розробці комп'ютерної гри у жанрі horror для віртуальної реальності. Метою роботи є створення гри, яка забезпечить гравцям повне занурення у страхітливий світ психіатричної лікарні за допомогою сучасних технологій VR.

Основний зміст роботи викладено на 61 сторінках тексту, містить 6 таблиць, 28 ілюстрацій, 1 формулу, налічує 28 найменувань літературних джерел, додатки містять 11 сторінок. Загальний обсяг роботи – 75 сторінок.

У роботі детально описано процес розробки гри "Fear Within", починаючи з ідеї та закінчуючи тестуванням кінцевого продукту. Було використано Unity 2021.3.19f1 як основний рушій гри, SteamVR для інтеграції VR, та гарнітуру Pico 4 для тестування та налагодження віртуальної реальності.

Розглянуто всі етапи розробки: створення сценарію, проектування ігрових локацій та персонажів, написання коду для основних механік гри, реалізація VR інтеграції та аудіовізуальних ефектів для створення атмосфери жаху. Проведено тестування гри, результати якого наведені у звіті, та описані процеси усунення виявлених дефектів.

Ключові слова: Unity, horror, virtual reality, комп'ютерна гра, програмне забезпечення, SteamVR, Pico 4.

					КН.КР.Б – 122 – 24 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Д.А. КУЛІСНОВ			Розробка комп'ютерної гри у жанрі horror у віртуальній реальності Пояснювальна записка	Літ.		Арк.	Аркушів
Керівник		М.С. ГРАФ						3	64
Рецензент		А.Р. ПЕТРОСЯН				Житомирська політехніка, група КНк-22-1			
Н. контр.									
Зав. каф.		М.С. ГРАФ							

ABSTRACT

This qualifying work is devoted to the development of a computer game in the genre of horror for virtual reality. The goal of the work is to create a game that will provide players with complete immersion in the terrifying world of a psychiatric hospital using modern VR technologies.

The main content of the work is laid out on 61 pages of text, contains 6 tables, 28 illustrations, 1 formula, includes 28 titles of literary sources, appendices contain 11 pages. The total volume of work is 75 pages.

The work describes in detail the process of developing the game "Fear Within", starting with the idea and ending with testing the final product. Unity 2021.3.19f1 was used as the main game engine, SteamVR for VR integration, and a Pico 4 headset for VR testing and debugging.

All stages of development are considered: creating a script, designing game locations and characters, writing code for the main mechanics of the game, implementing VR integration and audiovisual effects to create an atmosphere of horror. The game was tested, the results of which are given in the report, and the processes for eliminating the detected defects were described.

Keywords: Unity, horror, virtual reality, computer game, software, SteamVR, Pico 4.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	10
1.1 Постановка задачі	10
1.2 Аналіз існуючого програмного забезпечення.....	12
1.3 Концепт-документ (Concept Document)	17
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	24
2.1 Дизайн-документ (Game Design).....	24
2.2 Розробка архітектури ігрового додатку.....	33
2.3 Опис роботи з анімаціями.....	37
2.4 Основні етапи реалізації гри.....	39
РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ	49
3.1 Опис роботи з програмним додатком.....	49
3.2 Тестування роботи програмного забезпечення	55
ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ.....	5

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Blender – програмне забезпечення для створення 3D моделей та анімацій.

C# – мова програмування, яка використовується для написання коду гри в Unity.

Canvas – елемент графічного інтерфейсу користувача, що використовується для відображення тексту, зображень та інших елементів у грі.

ГГ (Головний Герой) – центральний персонаж гри, яким керує гравець.

IDE (Integrated Development Environment) – інтегроване середовище розробки, програмне забезпечення для написання, налагодження та тестування коду.

NavMesh – навігаційна сітка, яка використовується для визначення шляхів пересування персонажів в ігровому середовищі.

Pico 4 – гарнітура для віртуальної реальності, що використовується для створення ігрового досвіду у VR.

SDK (Software Development Kit) – набір інструментів для розробки програмного забезпечення.

Trigger – механізм, що активує певні події в ігровому процесі, коли гравець або інший об'єкт потрапляє в певну область.

UI (User Interface) – інтерфейс користувача, засоби взаємодії користувача з програмним забезпеченням.

Unity – ігровий рушій, що використовується для створення 2D та 3D ігор, а також інтеграції з VR.

VR (Virtual Reality) – віртуальна реальність, технологія, яка створює ілюзію присутності в тривимірному комп'ютерному середовищі.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми

У сучасному світі відеоігри є важливою складовою індустрії розваг. Вони не тільки забезпечують користувачам можливість відпочинку та розваг, але й сприяють розвитку когнітивних навичок, реакції, стратегічного мислення та навіть соціальних взаємодій. Зокрема, жанр horror займає особливе місце серед відеоігор завдяки своїй здатності викликати сильні емоції, адреналіновий поспіх та захоплення. Ігри цього жанру дозволяють гравцям переживати почуття страху та невизначеності в контрольованому середовищі, що створює унікальний ігровий досвід.

З розвитком технологій віртуальної реальності (VR) жанр horror отримав новий поштовх для розвитку. VR-технології забезпечують гравцям повне занурення в ігровий світ, створюючи відчуття присутності, яке неможливо досягти за допомогою традиційних плоских екранів. Використання VR гарнітур дозволяє гравцям безпосередньо взаємодіяти з об'єктами віртуального світу, що підвищує рівень реалістичності та інтенсивності геймплею. Віртуальна реальність дає можливість створювати більш насичені, детальні та емоційно напружені сцени, які глибше впливають на гравців.

Психіатрична лікарня як місце дії для horror гри є особливо вдалим вибором, оскільки вона сама по собі має атмосферу страху і невизначеності. Замкнені простори, темні коридори, загадкові звуки та непередбачувані події – все це створює ідеальні умови для розвитку хоррор сюжету. Гравець, перебуваючи у віртуальній психіатричній лікарні, відчуває себе частиною цього страхітливого світу, що значно посилює емоційний вплив гри.

Інтеграція VR у розробку гри також відкриває нові можливості для дослідження поведінки гравців та їхньої реакції на різні стимули. Завдяки VR можна більш точно моделювати реакції на страх, вивчати психологічні аспекти взаємодії людини з віртуальним середовищем, а також розробляти нові методики для створення ще більш захоплюючих і реалістичних ігрових досвідів.

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

З економічної точки зору, ринок VR-ігор продовжує активно розвиватися, залучаючи все більше інвестицій та інтересу з боку як розробників, так і користувачів. Створення якісних ігор у жанрі horror для VR може стати успішним комерційним проектом, що сприятиме подальшому розвитку індустрії VR. Більше того, такі проекти можуть стати цінним внеском у розвиток VR технологій, сприяючи їх популяризації та впровадженню в різні сфери, включаючи освіту, медицину та інші галузі.

Таким чином, розробка гри у жанрі horror для віртуальної реальності є актуальним та перспективним напрямом досліджень і розробок. Вона об'єднує новітні технології, психологічні аспекти геймплею та економічний потенціал, що робить цей проект важливим як з наукової, так і з практичної точки зору.

Мета і завдання дослідження. Метою даної роботи є розробка комп'ютерної гри у жанрі horror для віртуальної реальності, яка забезпечить гравцям повне занурення у страхітливий світ психіатричної лікарні за допомогою сучасних технологій VR. Для досягнення цієї мети було поставлено наступні завдання:

- Розробка сюжету гри та сценаріїв.
- Створення ігрових локацій та персонажів.
- Інтеграція VR технологій з використанням SteamVR та гарнітури Pico 4.
- Розробка аудіовізуальних ефектів для посилення атмосфери жаху.
- Тестування та налагодження гри.

Об'єкт дослідження. Об'єктом дослідження є процес розробки гри у віртуальній реальності.

Предмет дослідження. Предметом дослідження є методи та інструменти розробки VR ігор у жанрі horror.

Методи дослідження. В роботі використані методи системного аналізу, об'єктно-орієнтованого програмування, VR інтеграції, а також методи розробки сценаріїв та графіки.

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

Практичне значення отриманих результатів. Розроблена гра може бути використана як комерційний продукт, а також як навчальний матеріал для студентів, які вивчають розробку відеоігор та VR технологій. Результати роботи можуть стати основою для подальших досліджень у сфері VR та геймдизайну.

Апробація результатів. Матеріали даної роботи були представлені та обговорені на низці наукових конференцій. Результати досліджень викликали зацікавленість серед науковців та студентів, що підтверджує актуальність та важливість теми.

Практичні результати розробки комп'ютерної гри "Fear Within" були продемонстровані під час відкритих занять та майстер-класів для студентів спеціальності "Комп'ютерні науки", що дозволило отримати цінні відгуки та пропозиції щодо вдосконалення проєкту.

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

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Постановка задачі

Розробка комп'ютерної гри у жанрі horror для віртуальної реальності є складним завданням, яке вимагає глибоких знань у галузі програмування, дизайну, психології та технологій віртуальної реальності. Метою цього розділу є детальний аналіз проблематики створення такого роду ігрових проектів, а також обґрунтування вибору методів та засобів для їх вирішення.

Визначення основних компонентів гри:

- **Сюжет та сценарії.** Створення цікавого та захоплюючого сюжету є одним з ключових елементів успішної гри. Важливо розробити сценарії, які будуть підтримувати атмосферу жаху та утримувати гравця в напрузі протягом всієї гри.
- **Ігрові локації та персонажі.** Для досягнення високого рівня занурення в гру необхідно створити детальні та реалістичні ігрові локації та персонажів. У нашому випадку, дія гри відбувається в психіатричній лікарні, що додає до атмосфери гри.
- **Механіка гри.** Необхідно розробити механіки, які будуть відповідати жанру horror та підтримувати інтерактивність гри у VR. Це включає рухи персонажа, взаємодію з об'єктами, розгадування загадок та уникнення ворогів.
- **Аудіовізуальні ефекти.** Звукові та візуальні ефекти грають важливу роль у створенні атмосфери жаху. Вони повинні бути налаштовані таким чином, щоб максимально підсилювати відчуття страху та невизначеності.
- **Технології віртуальної реальності.** Використання VR технологій, таких як SteamVR та гарнітура Pico 4, дозволяє створити повне занурення в ігровий процес.

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

Аналіз існуючих проблем:

- **Технічні обмеження.** Віртуальна реальність вимагає значних ресурсів від комп'ютерної техніки, що може стати обмеженням для розробки складних ігрових світів з високою деталізацією.
- **Проблеми з оптимізацією.** Забезпечення плавного та безперебійного геймплею у VR є викликом, оскільки будь-які затримки або проблеми з продуктивністю можуть негативно вплинути на занурення гравця.
- **Складність взаємодії.** Розробка інтуїтивно зрозумілих і зручних механік взаємодії у VR може бути складною, оскільки гравець повинен відчувати себе комфортно при взаємодії з віртуальними об'єктами.
- **Психологічні аспекти.** Жанр horror має викликати страх та напругу, але при цьому не перевантажувати гравця негативними емоціями. Важливо знайти баланс, щоб гра була захоплюючою, але не надто лякаючою.

Методи вирішення задачі:

- **Вибір інструментів та технологій.** Для розробки гри було обрано Unity 2021.3.19f1 як основний рушій, SteamVR для інтеграції VR, та гарнітуру Pico 4 для тестування та налагодження гри. Ці інструменти дозволяють створити високоякісну графіку та забезпечити стабільний геймплей у віртуальній реальності.
- **Розробка та тестування прототипів.** Створення прототипів на ранніх стадіях розробки дозволяє виявити потенційні проблеми та оцінити ігрові механіки, перш ніж перейти до детального опрацювання.
- **Покрокове тестування та налагодження.** Регулярне тестування гри на різних етапах розробки дозволяє вчасно виявляти та усувати дефекти, що забезпечує високу якість кінцевого продукту.

Таким чином, розробка гри у жанрі horror для віртуальної реальності вимагає комплексного підходу та детального аналізу кожного етапу процесу. У наступних підрозділах буде розглянуто існуючі програмні засоби та технології, які можуть бути використані для вирішення поставлених завдань.

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

1.2 Аналіз існуючого програмного забезпечення

При розробці комп'ютерної гри у жанрі horror для віртуальної реальності важливо провести детальний аналіз існуючих ігор, що вже зайняли свою нішу на ринку. Це дозволяє врахувати найкращі практики та уникнути поширених помилок. У цьому підрозділі буде розглянуто кілька відомих horror VR ігор, їхні геймплейні особливості, системні вимоги та графіку.

Resident Evil 7: Biohazard

Resident Evil 7: Biohazard є однією з найвідоміших ігор у жанрі horror, яка використовує технології віртуальної реальності. Гра пропонує гравцям занурення у світ жаху з реалістичною графікою та інтерактивністю. Дія гри відбувається в похмурому особняку, де головний герой, Ітан Вінтерс, шукає свою зниклу дружину Мію. Атмосфера гри створюється за рахунок тьмяного освітлення, зловісних звуків та несподіваних зустрічей з ворогами.

Особливості Resident Evil 7: Biohazard включають:

- Високоякісна графіка завдяки використанню рушія RE Engine. Деталізація середовища та персонажів є однією з найсильніших сторін гри.
- Інтерактивний геймплей, що включає пошук предметів, вирішення головоломок та боротьбу з ворогами. Важливим елементом є система керування ресурсами, що змушує гравців уважно обирати, коли і як використовувати знайдені предмети.
- Реалістичні звукові ефекти, які підсилюють відчуття страху та небезпеки. Звуки кроків, скрип дверей та моторошні голоси створюють атмосферу постійної напруги.
- Підтримка VR забезпечує глибоке занурення в ігровий світ, дозволяючи гравцям відчувати себе частиною подій.

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

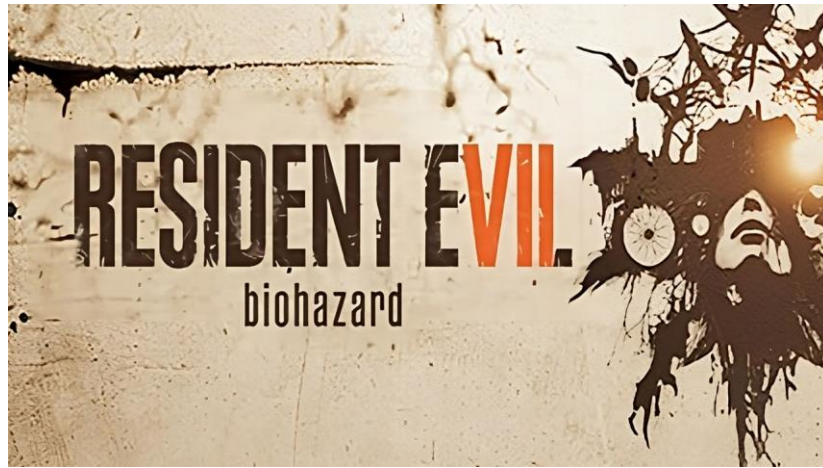


Рис. 1.1 – Resident Evil 7: Biohazard

The Exorcist: Legion VR

The Exorcist: Legion VR є інтерактивною грою, яка зосереджується на екзорцизмі та боротьбі з демонами. Гра розроблена з використанням рушія Unity і доступна на таких платформах, як Oculus Rift, HTC Vive та PlayStation VR. Гравці виступають у ролі детектива, який розслідує паранормальні явища і стикається з демонами.

Основні особливості The Exorcist: Legion VR включають:

- Атмосферний геймплей, що включає розслідування та взаємодію з паранормальними об'єктами. Гравці можуть використовувати різні інструменти для виявлення і боротьби з демонами, такі як хрести, святу воду та інші артефакти.
- Стилїзована графіка, яка, хоча і не така фотореалістична, як у Resident Evil 7, все ж створює потужний ефект занурення завдяки правильному використанню освітлення та візуальних ефектів.
- Моторошні аудіо ефекти, що включають шепоти, крики та інші звуки, які підсилюють відчуття жаху та тривоги.
- Підтримка VR дозволяє гравцям безпосередньо взаємодіяти з об'єктами гри, що робить процес екзорцизму ще більш реалістичним та страшним.

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

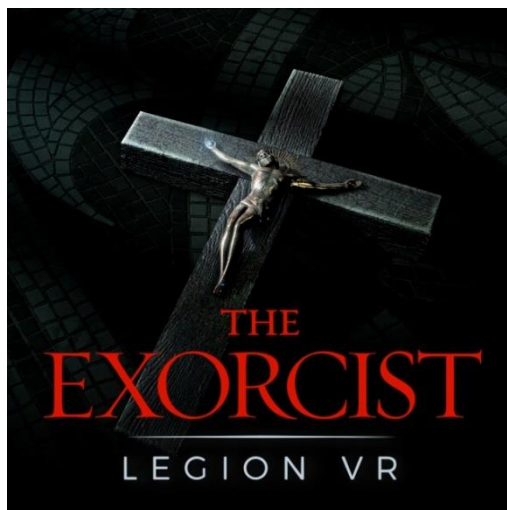


Рис. 1.2 – The Exorcist: Legion VR

Layers of Fear VR

Layers of Fear VR є психологічним horror, де гравці досліджують особняк, розкриваючи секрети минулого та стикаючись з моторошними явищами. Гра розроблена з використанням рушія Unity і доступна на таких платформах, як Oculus Rift, HTC Vive та PlayStation VR. У центрі сюжету – художник, який намагається завершити свій шедевр, борючись із власними демонами та психічними розладами.

Основні особливості Layers of Fear VR включають:

- Глибокий і емоційно насичений сюжет, який розкривається через взаємодію з оточуючими предметами та спогадами головного героя.
- Художній стиль графіки, що підсилює атмосферу гри. Візуальні ефекти змінюють сприйняття реальності, створюючи відчуття невизначеності та тривоги.
- Психологічні аудіо ефекти, які включають музику, звуки та голоси, що підсилюють відчуття страху та напруги.
- Середній рівень інтерактивності, що дозволяє гравцям досліджувати оточення та вирішувати головоломки без активної боротьби з ворогами.

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



Рис. 1.3 – Layers of Fear VR

Порівняння мінімальних системних вимог аналогів

Для забезпечення оптимального геймплею у віртуальній реальності важливо враховувати системні вимоги ігор. У таблиці наведено порівняння мінімальних системних вимог для трьох розглянутих ігор.

Таблиця 1.1

Порівняння мінімальних системних вимог аналогів

Гра	Процесор	Оперативна пам'ять	Відеокарта	Місце на диску	Операційна система
Resident Evil 7: Biohazard	Intel Core i5-4460	8 ГБ	NVIDIA GeForce GTX 760	24 ГБ	Windows 7/8/10 (64-bit)
The Exorcist: Legion VR	Intel i5-4590	8 ГБ	NVIDIA GTX 970	5 ГБ	Windows 7 SP1/8.1/10 (64-bit)
Layers of Fear VR	Intel Core i5-3470	8 ГБ	NVIDIA GeForce GTX 750 Ti	5 ГБ	Windows 7 (64-bit)

Порівняння геймплейних особливостей аналогів

Геймплейні особливості є критично важливими для забезпечення захоплюючого досвіду у VR іграх. У таблиці наведено порівняння основних геймплейних елементів для трьох розглянутих ігор.

Таблиця 1.2

Порівняння геймплейних особливостей аналогів

Гра	Режим гри	Тип ворогів	Головолом ки	Інтерактивні сть	Систе ма бою
Resident Evil 7: Biohazard	Однокористувач ький	Монстри, мутанти	Так	Висока	Так
The Exorcist : Legion VR	Однокористувач ький	Демони	Так	Висока	Ні
Layers of Fear VR	Однокористувач ький	Паранормал ьні явища	Так	Середня	Ні

Порівняння графіки та аудіо аналогів

Графіка та аудіо є важливими елементами для створення атмосфери жаху у VR іграх. У таблиці наведено порівняння якості графіки та аудіо для трьох розглянутих ігор.

Таблиця 1.3

Порівняння графіки та аудіо аналогів

Гра	Якість графіки	Тип графіки	Аудіо ефекти	Музичний супровід
Resident Evil 7: Biohazard	Висока	Фотореалістична	Реалістичні, лякаючі	Атмосферна, напружена
The Exorcist:	Висока	Стилізована	Реалістичні, моторошні	Атмосферна, напружена

Legion VR				
Layers of Fear VR	Середня	Художня	Атмосферні, психологічні	Атмосферна, інтригує

1.3 Концепт-документ (Concept Document)

Жанр і цільова аудиторія

"Fear Within" – це гра у жанрі horror, спеціально розроблена для віртуальної реальності (VR). Цей жанр обирається через його здатність викликати сильні емоційні реакції та забезпечувати незабутній ігровий досвід. Основна цільова аудиторія включає любителів horror ігор, фанатів VR технологій, а також тих, хто шукає інтенсивних ігрових відчуттів. Вік аудиторії орієнтовно складає 18-35 років, що обумовлено як тематикою гри, так і необхідністю використання VR обладнання.

Основні особливості гри

"Fear Within" пропонує гравцям унікальний досвід, що базується на повному зануренні у світ жаху. Основні особливості гри включають:

- Висока реалістичність.** Завдяки використанню передових графічних технологій і VR, гравці зможуть відчувати себе всередині гри, взаємодіяти з оточуючими об'єктами та відчувати їхню присутність.
- Атмосферність.** Гра створює напружену і моторошну атмосферу через використання звукових і візуальних ефектів, що сприяють зануренню у світ гри.
- Інтерактивність.** Гравці можуть взаємодіяти з різноманітними об'єктами, вирішувати головоломки, знаходити підказки та ключі, що просувають сюжет.
- Психологічний аспект.** Гра включає елементи психологічного horror, які впливають на підсвідомість гравця і створюють глибше відчуття страху.

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

Геймплей

Геймплей у "Fear Within" зосереджений на дослідженні оточуючого світу, розгадуванні головоломок і уникненні небезпек. Гравець керує головним героєм від першої особи, використовуючи VR контролери для переміщення та взаємодії з об'єктами. Основні механіки гри включають:

- **Рух персонажа.** Гравець може пересуватися по різних локаціях лікарні, досліджуючи їх та знаходячи корисні предмети.

- **Взаємодія з об'єктами.** Гравець може підбирати предмети, використовувати їх для вирішення головоломок, відкривати двері та активувати різні механізми.

- **Пошук підказок та ключів.** Щоб просуватися по сюжету, гравець повинен знаходити підказки та ключі, що відкривають доступ до нових локацій.

- **Уникання ворогів.** Медсестра, яка є головним ворогом гри, переслідує гравця. Якщо вона доторкнеться до героя, гра перезавантажується з останньої контрольної точки.

Сюжет

Сюжет гри "Fear Within" розвивається у похмурій атмосфері покинутої психіатричної лікарні, де головний герой прокидається в операційній кімнаті, нічого не пам'ятаючи про те, як він туди потрапив. На столі поруч він знаходить записку з підказкою та ключ від дверей. Відкривши двері, герой бачить кровавий слід, що веде до класної кімнати.

У класній кімнаті ГГ знаходить труп з іншою запискою та ключем. Записка містить тривожні відомості про лікарню та підказує, що герой повинен продовжувати свій шлях до офісу на другому поверсі. По дорозі його лякає збожеволіла медсестра – головний ворог гри, яка переслідує гравця протягом усього сюжету.

Потрапивши в офіс, ГГ знаходить старий телевізор і включає його. На екрані він бачить запис з камер спостереження, на якому видно, як медсестра

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

ховає ключ від головних дверей у кімнаті охоронця. У цій кімнаті також знаходиться телефон, за допомогою якого герой викликає поліцію.

ГГ спускається на перший поверх, щоб перевірити кімнату охоронця, але вона зачинена. З'являється підказка, що скло в цю кімнату можна розбити чимось важким. Досліджуючи локацію, гравець заходить у актову залу, де бачить проектор, повішений труп з молотком у голові та записку. В записці зазначено, що потрібно знайти касету у підвалі (в морзі), з якою можна буде відтворити відео на проекторі.

ГГ спускається у морг, знаходить касету у трупа, але медсестра починає переслідувати його. Втікши від неї, герой повертається до актової зали, вставляє касету у проектор і бачить страшне відео. Після завершення відео труп з молотком у голові падає, що дозволяє ГГ забрати молоток.

Розбивши вікно в кімнаті охоронця, герой знаходить фінальний ключ та виходить з лікарні. На вулиці його зустрічає поліція, яка допомагає йому вибратися з цього моторошного місця. Гра закінчується, ГГ бачить фінальний екран де розповідається що з ним сталося далі, та безпечно перекидається до головного меню.

Сюжет "Fear Within" побудований таким чином, щоб поступово розкривати історію лікарні та головного героя, підсилюючи атмосферу страху та невизначеності. Кожна нова локація приносить нові виклики та загадки, які гравець повинен розгадати, щоб просуватися по сюжету.

Платформа

Гра "Fear Within" розробляється з використанням рушія Unity 2021.3.19f1, який підтримує кросплатформенну розробку та інтеграцію VR технологій. Для інтеграції VR використовується SteamVR, що забезпечує підтримку гарнітури Pico 4. Unity надає широкий спектр інструментів для створення високоякісної графіки та складних ігрових механік, що робить його ідеальним вибором для розробки horror VR гри.

Розробка для платформи VR забезпечує гравцям повне занурення у гру, дозволяючи взаємодіяти з оточенням і відчувати реалістичність подій.

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

Підтримка SteamVR гарантує, що гра буде сумісна з різними VR гарнітурами, забезпечуючи широку аудиторію користувачів.

Таким чином, концепт-документ гри "Fear Within" визначає основні аспекти розробки, включаючи жанр, цільову аудиторію, основні особливості, геймплей, сюжет та платформу, що забезпечує чітке розуміння проекту у процесі створення гри.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Розробка гри у жанрі horror для віртуальної реальності вимагає ретельного підбору інструментальних засобів та врахування вимог до апаратного забезпечення. Вибір інструментів визначається не тільки їхніми можливостями, але й здатністю забезпечити оптимальну продуктивність та якість гри.

Unity 2021.3.19f1

Для розробки гри "Fear Within" було обрано рушій Unity 2021.3.19f1 з кількох причин:

1. **Кросплатформеність.** Unity підтримує розробку ігор для різних платформ, включаючи Windows, macOS, Android, iOS, а також VR гарнітури, такі як Oculus Rift, HTC Vive та Pico 4. Це забезпечує гнучкість у виборі цільової платформи для випуску гри.
2. **Потужний інструментарій.** Unity надає великий набір інструментів для створення 2D та 3D графіки, фізики, анімацій та звукових ефектів. Це дозволяє реалізувати всі необхідні аспекти гри на високому рівні.
3. **Інтеграція з VR.** Unity має вбудовану підтримку SteamVR та інших VR SDK, що забезпечує просту інтеграцію VR функціональності у гру. Це особливо важливо для створення занурювального ігрового досвіду.
4. **Велика спільнота та підтримка.** Unity має велику спільноту розробників та обширну документацію, що значно спрощує процес навчання та вирішення проблем, які можуть виникнути під час розробки.

SteamVR

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

SteamVR було обрано для інтеграції VR через його переваги:

1. **Широка підтримка VR гарнітур.** SteamVR підтримує більшість сучасних VR гарнітур, включаючи HTC Vive, Oculus Rift та Pico 4, що забезпечує сумісність гри з різними пристроями.

2. **Потужний SDK.** SteamVR SDK надає всі необхідні інструменти для розробки VR контенту, включаючи відстеження рухів, налаштування контролерів та інтеграцію з ігровими рушіями. Це дозволяє легко додавати VR функціональність у гру та налаштовувати її під конкретні потреби.

3. **Гнучкість налаштувань.** SteamVR дозволяє розробникам налаштовувати параметри гри під особливості VR обладнання, що забезпечує оптимальну продуктивність та якість зображення.

Pico 4

Гарнітура Pico 4 була обрана для тестування та налагодження гри через наступні характеристики:

1. **Висока роздільна здатність.** Гарнітура має дисплей з високою роздільною здатністю, що забезпечує чітке та реалістичне зображення, важливе для занурення у світ horror.

2. **Зручність використання.** Ергономічний дизайн та легка вага гарнітури роблять її комфортною для тривалого використання, що є критично важливим для гри у VR.

3. **Відстеження рухів.** Pico 4 оснащена системою відстеження рухів, що дозволяє точно передавати рухи голови та рук у віртуальному середовищі, підвищуючи реалістичність взаємодії з об'єктами гри.

Мова програмування

Для розробки ігрової логіки та основних механік гри було обрано мову програмування C#. Вибір C# обумовлений наступними факторами:

1. **Сумісність з Unity.** Unity підтримує C# як основну мову програмування для створення скриптів, що дозволяє легко інтегрувати ігрову логіку у проект.

2. **Висока продуктивність.** C# забезпечує високу продуктивність і дозволяє ефективно працювати з ресурсами гри.

3. **Зручність для розробників.** C# має простий та зрозумілий синтаксис, що робить його доступним як для новачків, так і для досвідчених розробників.

Visual Studio

Для написання коду гри було обрано інтегроване середовище розробки (IDE) Visual Studio. Основні переваги Visual Studio:

1. **Потужні інструменти для розробки.** Visual Studio надає розширені можливості для написання, налагодження та тестування коду.

2. **Інтеграція з Unity.** Visual Studio підтримує інтеграцію з Unity, що дозволяє зручно редагувати та налагоджувати скрипти безпосередньо в IDE.

3. **Розширюваність.** Visual Studio має велику кількість плагінів та розширень, які можуть підвищити продуктивність розробки та полегшити роботу з кодом.

Blender

Для створення 3D моделей та анімацій було обрано програму Blender. Переваги Blender:

1. **Безкоштовність та відкритий код.** Blender є безкоштовним інструментом з відкритим кодом, що робить його доступним для всіх розробників.

2. **Широкий функціонал.** Blender надає потужні інструменти для створення 3D моделей, текстур, анімацій та візуальних ефектів.

3. **Сумісність з Unity.** Blender підтримує експорт моделей у формати, що сумісні з Unity, що дозволяє легко імпортувати створені об'єкти у гру.

Вимоги до апаратного забезпечення

Розробка гри для віртуальної реальності вимагає врахування високих вимог до апаратного забезпечення, щоб забезпечити плавний та безперебійний геймплей. Нижче наведено мінімальні та рекомендовані вимоги до апаратного забезпечення для гри "Fear Within".

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Системні вимоги

Параметр	Мінімальні вимоги	Рекомендовані вимоги
Процесор	Intel Core i5-4590	Intel Core i7-9700K
Оперативна пам'ять	8 ГБ	16 ГБ
Відеокарта	NVIDIA GTX 970	NVIDIA GTX 1080 Ti
Місце на диску	12 ГБ	12 ГБ
Операційна система	Windows 10 (64-bit)	Windows 10 (64-bit)

Вимоги до апаратного забезпечення відображають необхідність потужного обладнання для забезпечення високої якості графіки та плавного геймплею у VR середовищі. Рекомендовані вимоги забезпечують оптимальну продуктивність та найкращий досвід для гравців.

Висновки до першого розділу

У першому розділі було проведено аналіз проблематики розробки гри у жанрі horror для віртуальної реальності, зокрема досліджено такі ігри, як Resident Evil 7: Biohazard, The Exorcist: Legion VR та Layers of Fear VR. Це дозволило визначити найкращі практики та особливості, що можуть бути використані при розробці "Fear Within".

Обґрунтовано вибір інструментів: Unity 2021.3.19f1 як основний рушій гри, SteamVR для інтеграції VR, гарнітуру Pico 4 для тестування, C# для написання коду, Visual Studio для розробки та Blender для створення 3D моделей.

Було вирішено використовувати механіку плавного переміщення замість телепорту, інтерактивність у взаємодії з об'єктами та атмосферні аудіовізуальні ефекти для підвищення рівня занурення у гру.

Ці висновки забезпечили чітке уявлення про основні аспекти розробки гри "Fear Within" та її успішну реалізацію у наступних етапах проекту.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Дизайн-документ (Game Design)

Сюжет

"Fear Within" розгортається у похмурій атмосфері покинутої психіатричної лікарні. Головний герой прокидається в операційній кімнаті, не пам'ятаючи, як він туди потрапив. На столі поруч він знаходить записку з підказкою та ключ від дверей. Головний герой починає своє дослідження, натрапляючи на різні загадки, записки, і ключі, що допомагають розкрити історію лікарні та його власного минулого. Гравець стикається з різними випробуваннями, включаючи зустрічі зі збожеволілою медсестрою, яка переслідує його протягом всієї гри. У кінці, гравець знаходить спосіб вибратися з лікарні за допомогою поліції.

Призначення гри

Гра "Fear Within" створюється для занурення гравця у захоплюючий світ жаху за допомогою технологій віртуальної реальності. Гра спрямована на любителів horror-жанру та шанувальників VR, надаючи їм унікальний ігровий досвід з високим рівнем занурення та інтенсивними емоціями.

Мета створення гри

Метою створення "Fear Within" є поєднання сучасних технологій VR з класичними елементами horror-жанру для створення захоплюючого та емоційно насиченого ігрового досвіду. Гра покликана дослідити психологічні аспекти страху, забезпечуючи гравців інноваційними ігровими механіками та захоплюючим сюжетом. Крім того, гра має комерційний потенціал, враховуючи зростаючий інтерес до VR-ігор.

Основні особливості гри

- Висока реалістичність:** Використання передових графічних технологій та VR для створення реалістичного ігрового світу.
- Атмосферність:** Напружена та моторошна атмосфера, створена за допомогою звукових та візуальних ефектів.

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

3. **Інтерактивність:** Взаємодія з об'єктами, вирішення головоломок, пошук підказок та ключів.

4. **Психологічний аспект:** Елементи психологічного horror, що впливають на підсвідомість гравця.

Геймплей

Гравець керує головним героєм від першої особи, використовуючи VR контролери для переміщення та взаємодії з об'єктами. Основні механіки включають рух персонажа, підбирання та використання предметів, вирішення головоломок, уникнення ворогів та інші дії.

Платформа

Гра розробляється з використанням рушія Unity 2021.3.19f1, який підтримує кросплатформенну розробку та інтеграцію VR технологій. Для VR інтеграції використовується SteamVR, що забезпечує підтримку гарнітури Pico 4. Unity надає всі необхідні інструменти для створення високоякісної графіки та складних ігрових механік.

Управління головним героєм

Одним із ключових аспектів гри "Fear Within" є управління головним героєм. Використання VR забезпечує повне занурення у гру, що вимагає особливої уваги до деталей, таких як спосіб пересування персонажа. У стандартних налаштуваннях SteamVR для пересування у VR іграх часто використовується механіка телепорту, яка дозволяє гравцям швидко переміщуватися з однієї точки в іншу. Однак, для створення більш реалістичного та занурювального геймплею у "Fear Within" було вирішено використати механіку плавного переміщення (скользіння) за допомогою стіка контролера.

Для реалізації цієї механіки був написаний спеціальний код, який забезпечує плавне переміщення головного героя у віртуальному середовищі. Це створює більш природне відчуття пересування, подібне до руху у реальному світі. Щоб забезпечити коректну роботу механіки плавного переміщення, було налаштовано SteamVR Input, що включало налаштування відповідних дій для стіка контролера та їх інтеграцію з системою управління персонажем у Unity.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

У певний момент геймплею гравцеві буде надано можливість використовувати механіку телепорту для того, щоб залізти у розбите вікно в кімнаті охоронця. Це забезпечує гнучкість у пересуванні та додає додатковий рівень інтерактивності в грі.

Гравець має можливість активно взаємодіяти з оточуючими об'єктами, що додає реалістичності та інтерактивності ігровому процесу. Основні види взаємодії включають підбирання предметів, кидання об'єктів та відкривання дверей, шаф і інших об'єктів. Переваги використання плавного переміщення та взаємодії з об'єктами полягають у підвищеній реалістичності, покращеному зануренні, більшій свободі дій та різноманітності взаємодії.

Ігрові механіки

Ігрові механіки гри "Fear Within" визначають, як гравець взаємодіє з ігровим світом, проходить сюжет і які випробування стоять перед ним. Окрім механіки переміщення, яка була описана раніше, важливою частиною гри є взаємодія з оточуючими об'єктами, вирішення головоломок та уникання ворогів. Гравець може підбирати різні предмети, такі як ключі, записки, інструменти, та використовувати їх для просування по сюжету, вирішення головоломок або взаємодії з іншими об'єктами. Підняті предмети можна кидати для взаємодії з іншими об'єктами. Гравець може відчиняти двері, шафи та інші об'єкти, що відкривають нові локації або приховані предмети.

Однією з основних загроз у грі є збожеволіла медсестра, яка переслідує гравця протягом усього сюжету. Гравець повинен уникати зустрічей з медсестрою, оскільки її доторк може призвести до загибелі персонажа та перезапуску з останньої контрольної точки. Медсестра може з'являтися несподівано, створюючи елементи страху та напруги, що додає динамічності та непередбачуваності ігровому процесу, підсилюючи атмосферу жаху.

Головоломки та підказки є важливими елементами гри, які допомагають гравцеві просуватися по сюжету. Гравець повинен розгадувати різні головоломки, які можуть включати логічні завдання, пошук предметів, активацію механізмів тощо. Головоломки інтегровані у сюжет гри та сприяють

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

розвитку історії, відкриттю нових локацій та отриманню необхідних предметів. У різних місцях гри гравець знаходить підказки, які допомагають розгадувати головоломки та просуватися по сюжету. Підказки можуть бути представлені у вигляді записок, листів, аудіозаписів та інших елементів, що містять інформацію про подальші дії гравця.

Звукові та візуальні ефекти грають важливу роль у створенні атмосфери жаху та занурення у гру. Реалістичні звуки кроків, скрип дверей, шепоти, крики та інші звуки підсилюють атмосферу страху та тривоги. Звукові ефекти використовуються для створення напруженої атмосфери, підказок про присутність ворогів та взаємодії з об'єктами. Тьмяне освітлення, тіні, кроваві сліди та інші візуальні ефекти створюють моторошну атмосферу, підсилюючи емоційний вплив гри та створюючи несподіванки.

Таким чином, ігрові механіки гри "Fear Within" спрямовані на створення захоплюючого та занурювального досвіду у віртуальній реальності. Використання плавного переміщення, можливість телепортації, взаємодія з об'єктами, уникання ворогів, головоломки та підказки, а також звукові та візуальні ефекти роблять гру багатогранною та цікавою для гравців. Ці механіки забезпечують високу інтерактивність та реалістичність, що є ключовими елементами для успішної реалізації проекту "Fear Within".

Персонажі та створені моделі

Розробка персонажів та моделей є важливою частиною створення гри "Fear Within". У цьому підпункті буде детально описано основні персонажі гри, а також розглянуто процес створення моделей для ігрового середовища.

Головний герой гри "Fear Within" створений з використанням стандартних інструментів SteamVR. Персонаж управляється гравцем від першої особи, що забезпечує повне занурення у віртуальний світ. Стандартний персонаж SteamVR включає VR Rig для позиціонування та рухів голови і рук персонажа відповідно до рухів гравця, контролери для взаємодії з оточуючими об'єктами, підбирання предметів, відкривання дверей тощо, а також колізійні об'єкти, які забезпечують

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

фізичну взаємодію з оточенням, запобігаючи проходженню крізь стіни та інші об'єкти.

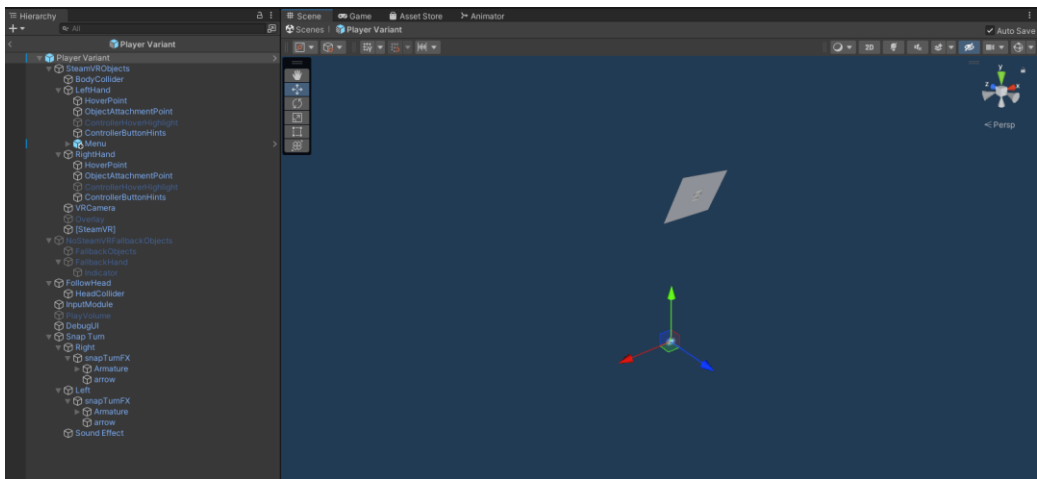


Рис 2.1 – Модель головного героя

Головним ворогом у грі є збожеволіла медсестра. Візуально вона має моторошний вигляд і служить основною загрозою для гравця протягом всієї гри. Модель медсестри включає скелетну анімацію для створення плавних рухів та реалістичної поведінки, детальні текстури для підкреслення її жахливого вигляду, а також звуковий супровід, що складається з криків, шепотів та інших моторошних звуків, що підсилюють атмосферу гри.



Рис 2.2 – Модель медсестри

Ігрове середовище у "Fear Within" розгортається у психіатричній лікарні, яка включає безліч різноманітних кімнат та коридорів. Кожна кімната має свої унікальні елементи, що створюють атмосферу жаху та напруги. Основні локації, які використовуються в сюжеті гри, включають операційну кімнату, де гравець прокидається, класну кімнату, де він знаходить першу підказку і ключ, офіс, де головний герой вмикає телевізор і бачить запис з камер спостереження, підвал (морг), де гравець знаходить касету, та актову залу, де він використовує касету для перегляду відео на проекторі. Ці кімнати були створені з використанням Unity та Blender.

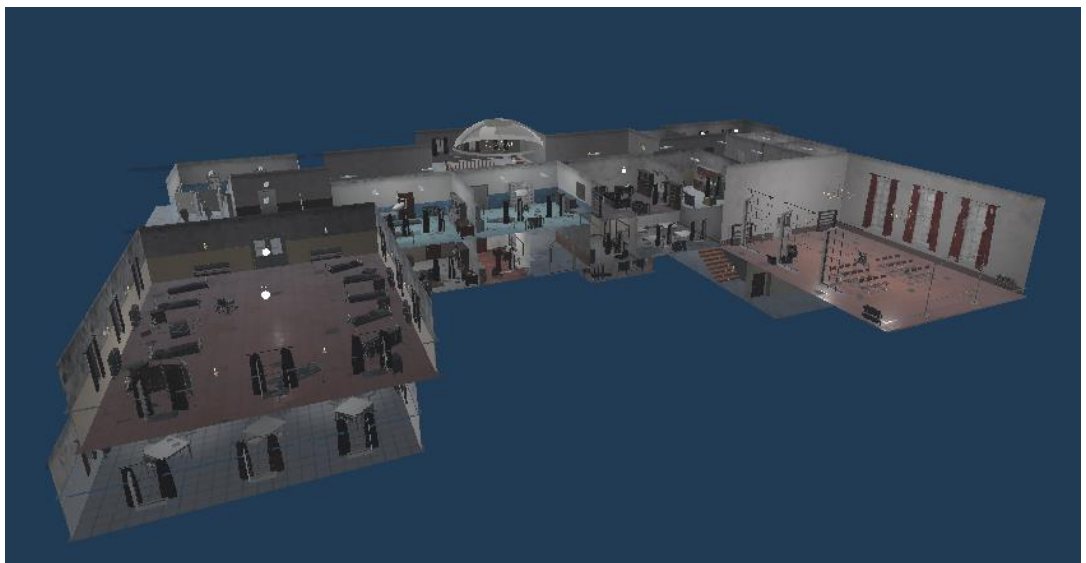


Рис 2.3 – Модель психіатричної лікарні

Записки виконують роль важливих елементів, що надають підказки та розкривають сюжет гри. Моделі записок мають вигляд старих пожовклих аркушів, які гравець може знаходити у різних місцях психіатричної лікарні. Кожна записка має свій унікальний текст, який гравець може прочитати, щоб отримати підказки щодо подальших дій. Приклад такої записки наведено на рисунку 2.4.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

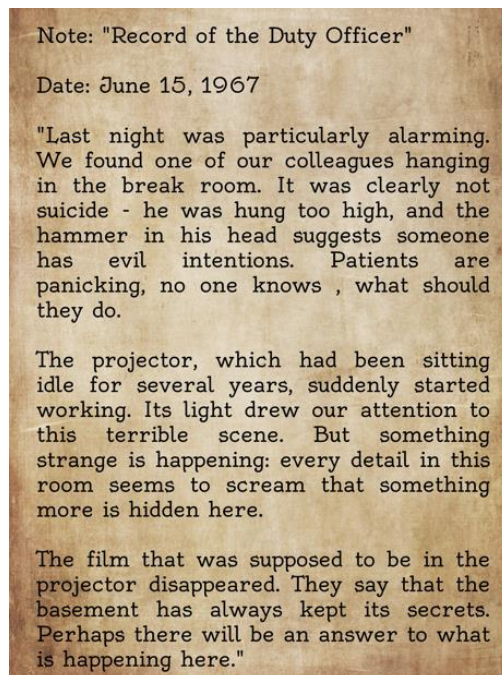


Рис 2.4 – Записка з підказкою

Модель мертвого лікаря використовується у грі для створення моторошної атмосфери та надання додаткових підказок. Лікар лежить на підлозі у калюжі крові, тримаючи у руці записку. Така сцена підсилює відчуття страху та непередбачуваності, адже гравець розуміє, що лікарня є небезпечним місцем, де трапляються жахливі події.



Рис 2.5 – Мертвий лікар з запискою

Ключ є важливим предметом у грі, оскільки він дозволяє гравцеві відчиняти двері та отримувати доступ до нових локацій. Модель ключа має класичний вигляд зі старовинним дизайном, що підходить до загальної

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

атмосфери гри. Гравець може знаходити ключі у різних місцях лікарні та використовувати їх для просування по сюжету.



Рис 2.6 – Ключ

Плівка для проектора є ще одним ключовим предметом у грі. Гравець повинен знайти цю плівку у підвалі (моргу) та використати її для перегляду відео на проекторі в актовій залі. Модель плівки виконана у вигляді старовинної кіноплівки, що підкреслює атмосферу гри та додає їй реалістичності.



Рис 2.7 – Плівка проектора

Однією з наймоторошніших моделей у грі є модель повішеного тіла з молотком у голові. Цей персонаж висить у актовій залі, створюючи страхітливую атмосферу. Молоток, що стирчить з голови тіла, додає жаху до сцени і є ключовим елементом для просування по сюжету, оскільки гравець має взаємодіяти з ним для розв'язання головоломок.

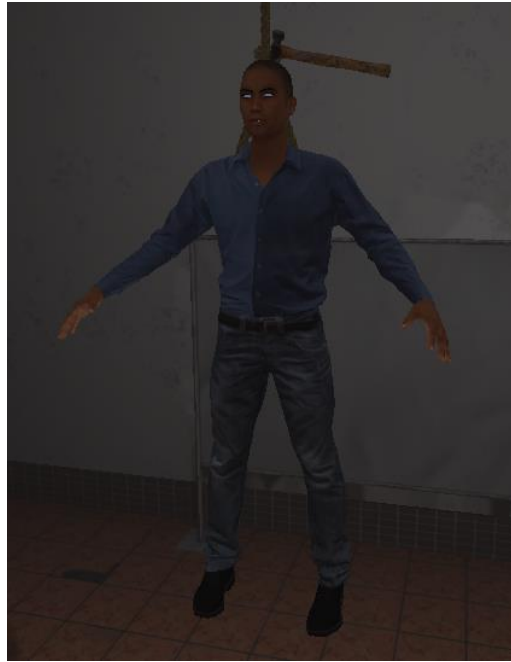


Рис 2.8 – Повішене тіло

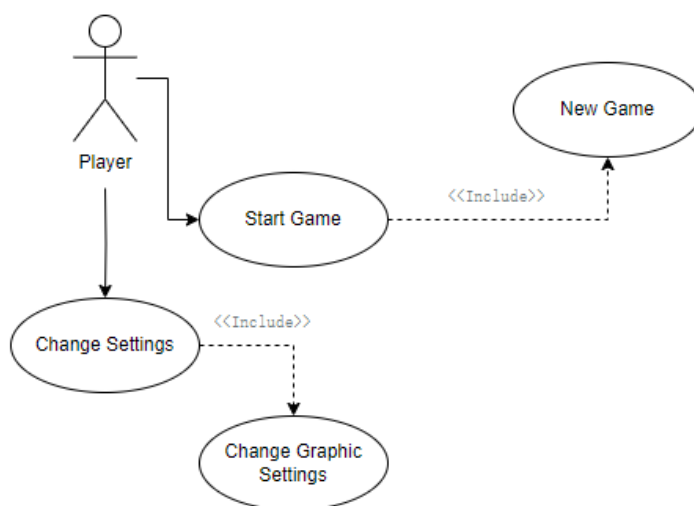
					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2 Розробка архітектури ігрового додатку

Діаграма варіантів використання — це графічне зображення взаємодії користувача з системою, що відображає взаємозв'язок між користувачами (акторами) та функціями системи (варіантами використання). Вона допомагає зрозуміти, як користувачі взаємодіють з різними елементами системи та які можливості надаються їм у межах цієї взаємодії.

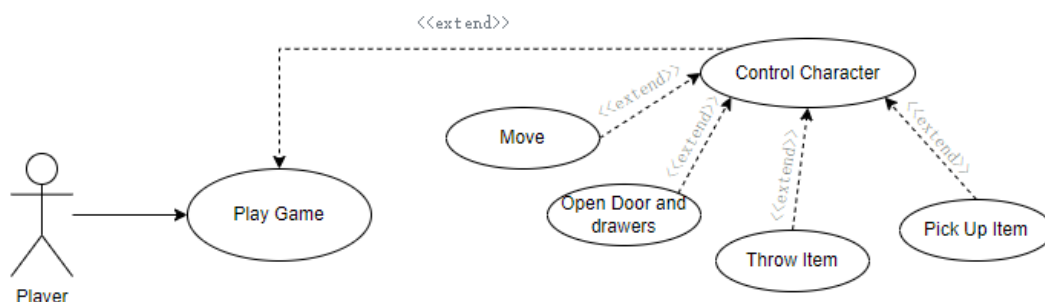
Побудовану діаграму варіантів використання можна бачити на рисунку 2.9

Main Menu



а)

The Game



б)

Рис. 2.9 – Діаграма варіантів використання:

а – перша частина, б – друга частина

На першій діаграмі варіантів використання відображено взаємодію гравця з головним меню гри "Fear Within". Головне меню - це перше, що бачить гравець після запуску гри. Звідси він може почати нову гру або змінити налаштування.

Гравець має можливість вибрати опцію "Start Game". Ця опція включає в себе початок нової гри, що позначено зв'язком з варіантом використання "New Game". Це означає, що кожен раз, коли гравець вибирає "Start Game", нова гра починається автоматично.

Друга основна опція у головному меню - це "Change Settings". Вибравши цю опцію, гравець може змінити параметри гри, такі як графічні налаштування. Це відображено зв'язком з варіантом використання "Change Graphic Settings". Таким чином, гравець може налаштувати графіку відповідно до своїх уподобань перед тим, як почати гру.

Друга діаграма варіантів використання зображує взаємодію гравця з основними елементами ігрового процесу в грі "Fear Within". Після того, як гравець вибирає "Play Game" у головному меню, він переходить до ігрового процесу, де може керувати своїм персонажем.

Варіант використання "Control Character" включає всі дії, які гравець може виконувати, керуючи персонажем у грі. Це включає в себе рух, відкривання дверей та ящиків, кидання предметів та підбирання предметів.

Прикладом може бути рух персонажем (Move). Коли гравець обирає рух, він керує своїм персонажем у просторі гри, використовуючи контролери. Аналогічно, коли гравець підходить до дверей або ящика, він може вибрати опцію "Open Door and Drawers" для взаємодії з цими об'єктами.

Кидання предметів дозволяє гравцю підбирати об'єкти з ігрового середовища та кидати їх у будь-якому напрямку. Це корисно для вирішення певних головоломок або взаємодії з оточенням. Підбирання предметів дозволяє гравцю брати корисні об'єкти, які можуть знадобитися для просування по сюжету гри.

Діаграма класів для відображає основні класи, які використовуються для реалізації гри, їхні атрибути, методи, а також взаємозв'язки між класами. Вона

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

допомагає зрозуміти структуру коду та взаємодію між різними компонентами гри.

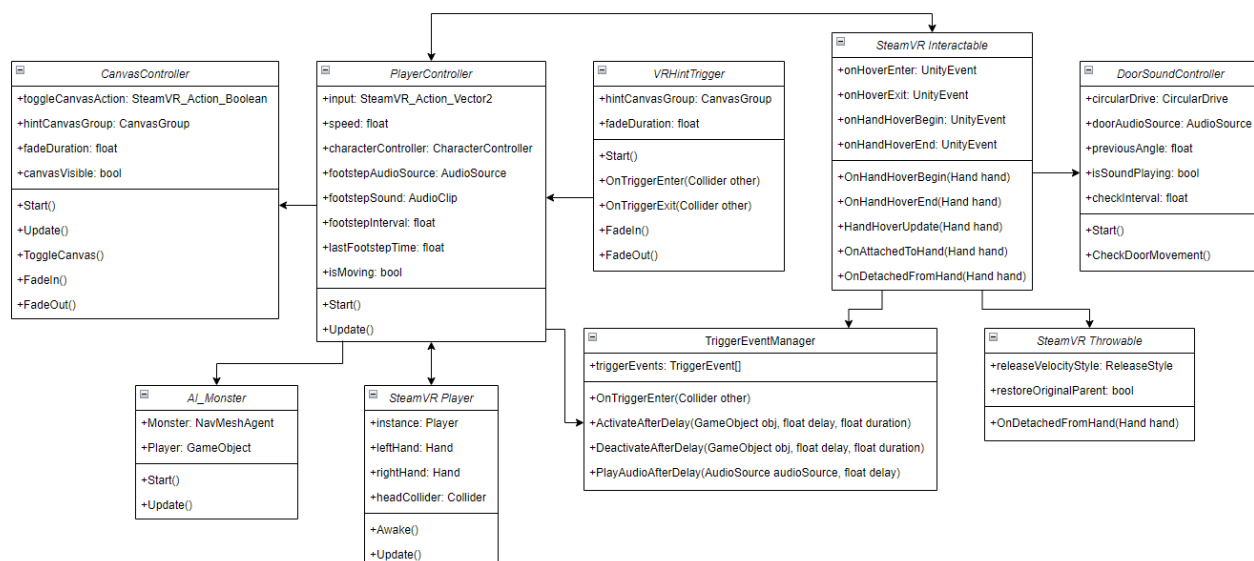


Рис 2.10 – Діаграма класів

CanvasController відповідає за керування відображенням і взаємодією з Canvas у грі. Цей клас дозволяє перемикає стан Canvas, а також плавно показувати і приховувати його.

PlayerController керує рухом персонажа гравця. Він отримує вхідні дані від контролера VR, визначає напрямок і швидкість руху, а також відтворює звуки кроків, коли гравець рухається.

VRHintTrigger відповідає за показ підказок гравцю, коли той входить або виходить із тригерної зони. Він використовує анімацію для плавного з'явлення і зникнення підказок.

AI_Monster керує поведінкою монстра у грі. Він використовує компонент NavMeshAgent для навігації та слідкування за гравцем, створюючи відчуття переслідування.

SteamVR Player є центральним класом для управління компонентами VR, включаючи руки і голову гравця. Він забезпечує зв'язок між різними частинами VR-системи.

SteamVR Interactable забезпечує взаємодію гравця з об'єктами у грі. Цей клас використовує події для обробки дій, таких як підйом предметів або відкриття дверей.

TriggerEventManager керує активацією різних подій у грі, таких як скримери або зміна стану об'єктів. Він може активувати або деактивувати об'єкти, а також відтворювати звукові ефекти за допомогою тригерів.

DoorSoundController відповідає за відтворення звуків дверей, коли вони рухаються. Він перевіряє, чи двері рухаються, і запускає або зупиняє звукові ефекти відповідно.

SteamVR Throwable дозволяє гравцю кидати об'єкти у грі. Цей клас керує фізичними властивостями об'єктів під час їх викидання з рук гравця.

У грі використано декілька шаблонів проектування для забезпечення ефективної та гнучкої архітектури.

Один із основних патернів, що використовується, це **Singleton**. Клас SteamVR Player реалізує цей патерн для зберігання єдиного екземпляра гравця в грі, що забезпечує централізований доступ до компонентів VR-системи.

Інший важливий патерн - **Event-Driven**. Класи SteamVR Interactable, TriggerEventManager та DoorSoundController використовують події для управління взаємодією між об'єктами. Це дозволяє децентралізовано керувати поведінкою об'єктів, що підвищує гнучкість та масштабованість системи.

Також у грі активно використовується **Component-Based** архітектура, що є основною парадигмою в Unity. Кожен клас представляє собою компонент, який може бути доданий до будь-якого ігрового об'єкта для розширення його функціональності. Це дозволяє легко додавати нові можливості та змінювати поведінку об'єктів, не порушуючи загальної структури коду.

Таким чином, діаграма класів відображає взаємозв'язок між різними компонентами гри "Fear Within" та ілюструє, як використання шаблонів проектування сприяє створенню ефективної та гнучкої архітектури.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3 Опис роботи з анімаціями

Анімації є невід'ємною частиною гри "Fear Within", оскільки вони додають реалістичності та динаміки ігровому процесу, підсилюючи емоційний вплив на гравця. У цьому розділі буде описано, як були створені та інтегровані анімації в гру, використовуючи різні інструменти та технології.

Використання Mixamo

Більшість анімацій для гри "Fear Within" були створені за допомогою сервісу Mixamo. Mixamo пропонує широкий вибір готових анімацій, які можна легко адаптувати для різних персонажів та ситуацій у грі.

Mixamo дозволив створити такі важливі анімації для медсестри, як ходьба, зачинення дверей, заховати об'єкт, а також танець, який виступає як пасхалка для гравців. Використання Mixamo дозволило отримати якісні анімації, які безперебійно інтегруються у Unity та працюють з різними персонажами гри.

Створення анімацій в Unity

Окрім використання Mixamo, деякі анімації були створені власноруч за допомогою Unity. Це дозволило створити унікальні анімації, які точно відповідають потребам гри та додають унікальності. Наприклад, анімація переміщення персонажа та анімація скримера були розроблені саме в Unity. Використання вбудованого редактора анімацій Unity забезпечило можливість тонкого налаштування рухів та дій персонажів, що дозволило досягти високого рівня реалістичності.

Анімація скримера, де медсестра виглядає із-за дверей, створює момент несподіваного жаху та тривоги, що є важливим елементом для підсилення емоційного впливу гри. Ця анімація була створена вручну, щоб забезпечити максимально реалістичний та страшний ефект.

Інтеграція анімацій у гру

Інтеграція анімацій у гру здійснювалася за допомогою механізмів анімації Unity. Animation та Animator дозволили створити плавні переходи між різними станами персонажів. Для кожного типу анімації був створений відповідний

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

анімаційний кліп, який потім прив'язувався до відповідного стану в Animator Controller. Це дозволило легко керувати анімаціями та створювати логіку для їх активації в залежності від дій гравця чи подій у грі.

Анімації для медсестри

Особлива увага приділялася анімаціям медсестри, оскільки вона є головним ворогом гри. Важливо було забезпечити її рухи та дії такими, що викликатимуть страх та тривогу у гравця. Анімація ходьби медсестри, створена за допомогою Міхато, дозволяє їй плавно та реалістично пересуватися по коридорах та кімнатах лікарні, створюючи постійне відчуття переслідування.

Анімація зачинення дверей та анімація, в якій медсестра ховає предмет, були створені для кат-сцен, таких як запис з камер спостереження. Ці анімації використовуються тільки в кат-сценах та в геймплейній складовій не беруть участь. Вони додають глибині сюжету, показуючи гравцеві важливі моменти через відео з камер спостереження.

Танець медсестри, який виступає як пасхалка, додає грі розважальну складову. Ця анімація була створена для того, щоб забезпечити невеликий відпочинок від напруженого ігрового процесу та додати елемент несподіванки.

Анімація скримера, де медсестра виглядає із-за дверей, була створена вручну у Unity для досягнення максимальної реалістичності та ефекту несподіваного налякування. Ця анімація є важливою складовою атмосфери жаху, що панує у грі.

Idle анімації для поліції

Окрім медсестри, у грі присутні й інші персонажі, такі як поліція, яка з'являється наприкінці гри. Для цих персонажів були створені idle анімації, які показують, як поліцейські стоять та чекають, створюючи атмосферу напруженості та очікування. Ці анімації були створені за допомогою Міхато та забезпечують реалістичність поведінки персонажів у грі.

Повішене тіло

Одним з наймоторошніших елементів гри є повішене тіло з молотком у голові. Для цієї моделі не використовувалися анімації, але був застосований

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

підхід з використанням ragdoll (технологія фізичної симуляції скелету). Це забезпечило найреалістичніший ефект падіння трупа з петлі, коли гравець взаємодіє з проектором. Завдяки ragdoll, рухи тіла виглядають природно і додають до загальної атмосфери жаху.

Таким чином, робота з анімаціями в грі "Fear Within" включала як використання готових анімацій з Mixamo, так і створення унікальних анімацій в Unity. Це дозволило забезпечити високу якість та реалістичність ігрового процесу, підсилюючи атмосферу жаху та інтерактивність гри. Кожна анімація була ретельно розроблена та інтегрована, щоб максимально підсилити емоційний вплив на гравця та забезпечити захоплюючий ігровий досвід.

2.4 Основні етапи реалізації гри

Для розробки гри "Fear Within" більша частина механік реалізована не тільки за допомогою коду, але й за допомогою стандартних компонентів SteamVR, таких як Interactable, Throwable, Linear Drive, Circular Drive та інші. Ці компоненти дозволяють легко реалізувати основні механіки, що відображають дії з "реального життя", такі як брати та кидати предмети, відчиняти і зачиняти двері, шафи та багато іншого.

Основна механіка: Рух персонажа

Основною механікою гри є, без сумнівів, управління діями головного героя, що включає в себе: переміщення персонажем по ігровому середовищу та інтеракцію з ігровими об'єктами. Воно здійснюється за допомогою скрипту PlayerController. Залежно від поточного стану героя, він може виконувати різні дії.

Основний код для руху персонажа був написаний на C# і використовував SteamVR для інтеграції з VR-контролерами.

Опис роботи коду PlayerController

Скрипт PlayerController відповідає за управління рухом персонажа у віртуальній реальності, використовуючи VR-контролери SteamVR. Він також обробляє звукові ефекти кроків для підвищення реалістичності ігрового процесу.

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

Ключові компоненти та їх роль:

- **SteamVR_Action_Vector2 input:** Це вхідний вектор, який отримує дані від VR-контролера про напрямок і силу натискання стіка.
- **float speed:** Швидкість, з якою гравець рухається в ігровому середовищі.
- **CharacterController characterController:** Компонент Unity, який відповідає за фізичне переміщення персонажа в ігровому просторі, обробляючи зіткнення і гравітацію.
- **AudioSource footstepAudioSource:** Аудіоджерело, яке використовується для відтворення звуків кроків персонажа.
- **AudioClip footstepSound:** Звуковий кліп, який містить звуки кроків.
- **float footstepInterval:** Інтервал часу між відтворенням звуків кроків.
- **float lastFootstepTime:** Час останнього відтворення звуку кроків, використовується для контролю інтервалу.
- **bool isMoving:** Прапорець, який вказує, чи рухається гравець в даний момент.

Основний код складається з двох методів:

• **Start():** Цей метод викликається один раз при запуску скрипта. Він ініціалізує CharacterController для управління переміщенням персонажа і налаштовує AudioSource для відтворення звуків кроків. Звук кроків завантажується і готується до відтворення, але ставиться на паузу, оскільки на початку гри гравець не рухається.

• **Update():** У цьому коді використовуються різні математичні концепції для забезпечення плавного та реалістичного пересування головного героя.

Vector2 та Vector3: Клас SteamVR_Action_Vector2 використовується для зберігання двовимірних векторів, які представляють вхідні сигнали від контролера VR. Вектори є основою для опису напрямку та величини руху персонажа.

```
Vector3 direction = Player.instance.hmdTransform.TransformDirection(new Vector3(input.axis.x, 0, input.axis.y));
```

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

Тут ми конвертуємо двовимірний вектор `input.axis` у тривимірний вектор для використання у просторі гри. `input.axis.x` та `input.axis.y` представляють координати на площині, які відповідають напрямку руху гравця.

Проекція на площину: Для обчислення руху персонажа з урахуванням гравітації ми використовуємо метод `Vector3.ProjectOnPlane`, який проєціює вектор руху на горизонтальну площину, усуваючи вертикальну компоненту.

```
characterController.Move(speed * Time.deltaTime *  
Vector3.ProjectOnPlane(direction, Vector3.up) - new Vector3(0, 9.81f, 0) *  
Time.deltaTime);
```

Це забезпечує природне пересування персонажа по поверхні без урахування вертикальних змін, що відповідає реалістичному досвіду руху.

Магнітуда вектора: Ми визначаємо, чи рухається персонаж, перевіряючи величину вектора напрямку. Якщо величина (магнітуда) вектора більше нуля, це означає, що персонаж рухається.

```
isMoving = direction.magnitude > 0 && characterController.isGrounded;
```

Магнітуда вектора `direction` розраховується як квадратний корінь з суми квадратів його компонент, що дозволяє точно визначити, чи є рух.

Управління подіями тригерів

Другим за важливістю скриптом у грі є `TriggerEventManager`. Цей скрипт забезпечує управління подіями тригерів у грі. Він дозволяє активувати та деактивувати різні об'єкти, а також відтворювати аудіо при вході гравця у зону тригера. Це забезпечує динамічність і інтерактивність ігрового процесу.

Опис роботи коду `TriggerEventManager`

Скрипт `TriggerEventManager` є універсальним інструментом для управління подіями тригерів у грі "Fear Within". Він використовується як для створення скрімерів, так і для просування по сюжету, вмикаючи та вимикаючи певні ігрові об'єкти. Це забезпечує глибше занурення в гру та додає елементи несподіванки та інтерактивності.

Ключові компоненти та їх роль:

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

- **TriggerEvent:** Внутрішній клас, який містить налаштування для кожної події тригера. Він включає:

- requiredTag: Тег, який повинен мати об'єкт для спрацювання тригера.
- gameObjectsToActivate: Масив об'єктів, які потрібно активувати.
- gameObjectsToDeactivate: Масив об'єктів, які потрібно деактивувати.
- audioSourcesToPlay: Масив аудіо джерел, які потрібно відтворити.
- activationDelays: Затримка перед активацією кожного об'єкта.
- deactivationDelays: Затримка перед деактивацією кожного об'єкта.
- audioPlayDelays: Затримка перед відтворенням кожного аудіо.
- activationDurations: Тривалість, на яку кожен об'єкт буде активований.
- deactivationDurations: Тривалість, на яку кожен об'єкт буде деактивований.

Основні методи:

- **OnTriggerEnter(Collider other):** Цей метод викликається, коли об'єкт входить у зону тригера. Він перевіряє, чи відповідає тег об'єкта заданому в TriggerEvent. Якщо тег збігається або не заданий, запускаються відповідні події:

- Активація об'єктів з урахуванням затримок і тривалості активації.
- Деактивація об'єктів з урахуванням затримок і тривалості деактивації.
- Відтворення аудіо з урахуванням затримок.

- **ActivateAfterDelay(GameObject obj, float delay, float duration):** Корутіна, яка активує об'єкт після заданої затримки і, якщо вказано, деактивує його через певний час. Це дозволяє динамічно змінювати стан об'єктів у грі.

- **DeactivateAfterDelay(GameObject obj, float delay, float duration):** Корутіна, яка деактивує об'єкт після заданої затримки і, якщо вказано, активує його через певний час. Це забезпечує гнучкість у управлінні об'єктами.

- **PlayAudioAfterDelay(AudioSource audioSource, float delay):** Корутіна, яка відтворює аудіо після заданої затримки. Це дозволяє синхронізувати звукові ефекти з іншими подіями в грі.

Управління звуками дверей

Скрипт DoorSoundController забезпечує управління звуками дверей у грі. Він контролює відтворення звуку, коли двері відкриваються або закриваються, використовуючи компонент CircularDrive для визначення кута повороту дверей. Це додає реалістичності та підсилює атмосферу гри.

Опис роботи коду DoorSoundController

Скрипт DoorSoundController є універсальним інструментом для управління звуками дверей у грі "Fear Within". Він використовується для створення реалістичного звукового супроводу при відкриванні та закриванні дверей. Це забезпечує додаткову глибину занурення у гру, створюючи більш правдоподібну атмосферу.

Ключові компоненти та їх роль:

- **CircularDrive:** Компонент, який визначає кут повороту дверей. Він використовується для визначення руху дверей.
- **AudioSource:** Компонент, який відтворює звукові ефекти дверей. Він запускає або зупиняє звук залежно від руху дверей.
- **float previousAngle:** Змінна для зберігання попереднього кута повороту дверей.
- **bool isSoundPlaying:** Прапорець, який вказує, чи відтворюється звук дверей в даний момент.
- **float checkInterval:** Інтервал перевірки руху дверей.

Основні методи:

- **Start():** Метод викликається один раз при запуску скрипта. Він ініціалізує компоненти CircularDrive та AudioSource, якщо вони не призначені в інспекторі, і переконується, що звук спочатку не відтворюється. Далі запускається повторювана перевірка руху дверей через заданий інтервал.

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

• **CheckDoorMovement():** Метод перевіряє, чи рухаються двері, визначаючи, чи змінився кут повороту дверей. Якщо двері рухаються і звук не відтворюється, він запускає звук. Якщо двері перестають рухатися і звук відтворюється, він ставить звук на паузу. Цей метод оновлює попередній кут повороту поточним кутом для наступної перевірки.

Обчислення кута повороту: У цьому коді використовується компонент CircularDrive, який відповідає за обчислення кута повороту дверей. CircularDrive.outAngle зберігає поточний кут повороту дверей.

```
float currentAngle = circularDrive.outAngle;
```

Обчислення зміни кута: Для визначення, чи рухаються двері, обчислюється абсолютна різниця між поточним кутом currentAngle та попереднім кутом previousAngle

```
if (Mathf.Abs(currentAngle - previousAngle) > 0.01f)
```

Використання функції Mathf.Abs гарантує, що різниця обчислюється як абсолютне значення, що дозволяє виявляти рух незалежно від напрямку повороту дверей.

Запуск та зупинка звуку: Якщо зміна кута перевищує порогове значення 0.01f, двері вважаються такими, що рухаються, і звук починає відтворюватися. Якщо зміна кута менша за порогове значення, звук ставиться на паузу.

Оновлення попереднього кута: Після перевірки стану руху дверей та відповідного запуску або зупинки звуку, значення попереднього кута оновлюється поточним кутом.

```
previousAngle = currentAngle;
```

Активація меню в грі

Ще одним важливим аспектом є активація меню в грі. Цей функціонал реалізується за допомогою скрипту CanvasController, який дозволяє гравцеві відкривати та закривати меню, використовуючи VR-контролер.

Опис роботи коду CanvasController

Скрипт CanvasController відповідає за активацію та деактивацію меню в грі "Fear Within". Він дозволяє гравцеві використовувати VR-контролер для

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

відкривання та закривання меню, що забезпечує зручний доступ до налаштувань та інших функцій гри.

Ключові компоненти та їх роль:

- **SteamVR_Action_Boolean toggleCanvasAction:** Дія, яка відповідає за перемикання стану меню при натисканні кнопки на VR-контролері.

- **CanvasGroup hintCanvasGroup:** Група канвасу, яка використовується для керування прозорістю та взаємодією з меню.

- **float fadeDuration:** Тривалість плавного переходу між станами видимості меню.

- **bool canvasVisible:** Змінна, яка зберігає поточний стан видимості меню.

Основний код складається з наступних методів:

- **Start():** Метод викликається один раз при запуску скрипта. Він ініціалізує CanvasGroup, встановлюючи початкову прозорість і відключаючи взаємодію та блокування променів.

- **Update():** Метод викликається кожного кадру і перевіряє, чи була натиснута кнопка на VR-контролері. Якщо так, він викликає метод ToggleCanvas() для перемикання стану меню.

- **ToggleCanvas():** Метод перемикає стан меню між видимим і прихованим. Якщо меню має стати видимим, запускається корутина FadeIn(), інакше - FadeOut().

- **FadeIn():** Корутина, яка забезпечує плавну появу меню, поступово збільшуючи прозорість CanvasGroup до 1 і вмикаючи взаємодію та блокування променів.

- **FadeOut():** Корутина, яка забезпечує плавне зникнення меню, поступово зменшуючи прозорість CanvasGroup до 0 і вимикаючи взаємодію та блокування променів.

Відображення підказок в VR

Для підвищення взаємодії гравця з грою та забезпечення орієнтації в ігровому світі використовуються підказки. Скрипт VRHintTrigger відповідає за

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

відображення підказок у VR, використовуючи групу Canvas для плавного відображення та зникнення підказок при вході та виході гравця з тригерної зони.

Опис роботи коду VRHintTrigger

Скрипт VRHintTrigger відповідає за відображення підказок у віртуальній реальності. Він використовується для надання гравцю інформації або орієнтації у грі, відображаючи підказки при вході гравця в певну зону та приховуючи їх при виході.

Ключові компоненти та їх роль:

- **CanvasGroup hintCanvasGroup**: Група Canvas, яка використовується для керування прозорістю підказки.

- **float fadeDuration**: Тривалість анімації появи та зникнення підказки.

Основний код складається з наступних методів:

- **Start()**: Метод викликається один раз при запуску скрипта. Він ініціалізує CanvasGroup, встановлюючи початкову прозорість на 0 (приховано).

- **OnTriggerEnter(Collider other)**: Метод викликається при вході об'єкта в зону тригера. Він перевіряє, чи має об'єкт тег "Player". Якщо так, запускається корутина FadeIn() для плавної появи підказки.

- **OnTriggerExit(Collider other)**: Метод викликається при виході об'єкта із зони тригера. Він перевіряє, чи має об'єкт тег "Player". Якщо так, запускається корутина FadeOut() для плавного зникнення підказки.

- **FadeIn()**: Корутина, яка забезпечує плавну появу підказки, поступово збільшуючи прозорість CanvasGroup.

- **FadeOut()**: Корутина, яка забезпечує плавне зникнення підказки, поступово зменшуючи прозорість CanvasGroup.

Управління поведінкою монстра

Однією з ключових механік гри є переслідування гравця монстром, що додає напруженості і створює атмосферу жаху. Це реалізовано за допомогою скрипта AI_Monster, який використовує компонент NavMeshAgent для навігації монстра до гравця.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

Опис роботи коду AI_Monster

Скрипт AI_Monster відповідає за управління поведінкою монстра у грі "Fear Within". Він використовує компонент NavMeshAgent для навігації монстра до позиції гравця, що створює ефект переслідування і підтримує напругу у грі.

Ключові компоненти та їх роль:

- **NavMeshAgent Monster:** Компонент, який відповідає за навігацію і рух монстра по ігровому середовищу.

- **GameObject Player:** Об'єкт гравця, який є метою переслідування монстра.

Основний код складається з наступних методів:

- **Start():** Метод викликається один раз при запуску скрипта. Він ініціалізує компонент NavMeshAgent для монстра і знаходить об'єкт гравця за тегом "Player".

- **Update():** Метод викликається кожного кадру і встановлює мету для NavMeshAgent у позиції гравця. Він також обчислює відстань до гравця для подальшого використання в інших механіках гри (наприклад, для визначення моменту атаки або скримера).

Використання NavMeshAgent: NavMeshAgent є компонентом Unity, який використовується для реалізації навігації персонажів у грі. Він дозволяє монстру переміщуватися у напрямку до заданої точки у просторі.

```
Monster = gameObject.GetComponent<NavMeshAgent>();
```

Знаходження гравця: У методі Start монстр знаходить об'єкт гравця за тегом "Player". Це дозволяє монстру завжди знати місцезнаходження гравця.

```
Player = GameObject.FindGameObjectWithTag("Player");
```

Встановлення цілі переслідування: У методі Update монстр постійно оновлює свою ціль переслідування, встановлюючи її на поточну позицію гравця.

```
Monster.SetDestination(Player.transform.position);
```

Використовується метод SetDestination, який приймає вектор позиції гравця та встановлює його як нову ціль для переміщення монстра.

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

Обчислення відстані до гравця: Для обчислення відстані між монстром і гравцем використовується метод `Vector3.Distance`. Цей метод обчислює евклідову відстань між двома точками у тривимірному просторі.

```
float    Dist_Player    =    Vector3.Distance(Player.transform.position,
gameObject.transform.position);
```

Евклідова відстань обчислюється за формулою:

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad 2.1$$

де (x_1, y_1, z_1) та (x_2, y_2, z_2) – координати монстра та гравця відповідно.

Висновки до другого розділу

У другому розділі детально розглянуто проектування та розробку гри "Fear Within" у жанрі horror для віртуальної реальності. Спершу було створено дизайн-документ, в якому описано сюжет гри, призначення та цільову аудиторію. Основні ігрові механіки та їх реалізація були детально продумані для забезпечення максимального занурення гравців.

При розробці архітектури ігрового додатку було використано діаграми варіантів використання та класів. Це дозволило чітко визначити взаємодію між компонентами гри та забезпечити структурований підхід до розробки.

Робота з анімаціями включала використання інструментів Mixamo та Unity для створення та інтеграції анімацій персонажів і об'єктів. Це значно підвищило реалістичність і динамічність ігрового процесу.

Основні етапи реалізації гри включали розробку коду для управління персонажем, реалізацію взаємодії з об'єктами, таких як підбирання, кидання предметів та відкривання дверей. Для забезпечення навігації ворогів у грі було створено код `AI_Monster`, який дозволяє монстрам переслідувати гравця, використовуючи алгоритми навігації.

Також у розділі описано створення 3D моделей, включаючи моделі головного героя, медсестри, та об'єктів лікарні. Використання Blender для створення цих моделей дозволило досягти високої деталізації та реалістичності, що значно підвищило візуальну якість гри.

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

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1 Опис роботи з програмним додатком

Гра "Fear Within" розпочинається з головного меню, яке дозволяє гравцеві налаштувати графіку. Після налаштування, гравець може почати нову гру.



Рис 3.1 – Головне меню



Рис 3.2 – Налаштування графіки

Після завантаження основної сцени гравець прокидається в операційній кімнаті психіатричної лікарні. На початку гравець знаходить записку з підказкою та ключ від дверей. Ця частина гри реалізована за допомогою скриптів, що активуються після взаємодії гравця з об'єктами на сцені. Взаємодія з об'єктами

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

реалізована за допомогою стандартних компонентів SteamVR, таких як Interactable, що дозволяє гравцю взяти ключ і відчинити двері.



Рис 3.3 – Стартова локація

Гравець управляє головним героєм за допомогою VR-контролерів, що дозволяють плавно пересуватися по ігровому середовищу. Основним механізмом переміщення є скользяння за допомогою стіка контролера, що забезпечує реалістичність переміщення. Також у грі реалізована можливість телепортуватися, яка використовується в певні моменти для подолання перешкод, наприклад, для залізання у розбите вікно, що зображено на рисунку 3.4.



Рис 3.4 – Точки для телепортації

Після виходу з операційної кімнати і по шляху до класної кімнати спрацьовує скрипт `TriggerEventManager`, що викликає появу медсестри, яка виходить з класної кімнати.



Рис 3.5 – Перша поява медсестри

Далі гравець знаходить записку на трупі доктора, у якій дані підказки. Знайшовши ключ від офісу, гравець прямує туди, дивиться відео на телевізорі, після чого з'являється підказка, визвана скриптом `VRHintTrigger`, що треба подзвонити по телефону і викликати поліцію.



Рис 3.6 – Підказка у офісі

Скрипт `TriggerEventManager` вмикає телевізор і запускає відтворення відео при взаємодії з об'єктом. Зробивши це, гравець виходить з кімнати, і його лякає скрімер медсестри, яка виглядає із-за дверей. Скрипт `TriggerEventManager` активується в цей момент, відтворюючи звук і анімацію появи медсестри.



Рис 3.7 – Відтворення відео на телевізорі

Спустившись на перший поверх, гравець йде до місця з відео, щоб забрати ключ від фінальних дверей, але двері в цю кімнату зачинені. Скрипт VRHintTrigger активується, коли гравець підходить до скла, показуючи підказку, що скло в цю кімнату можна розбити важким предметом.

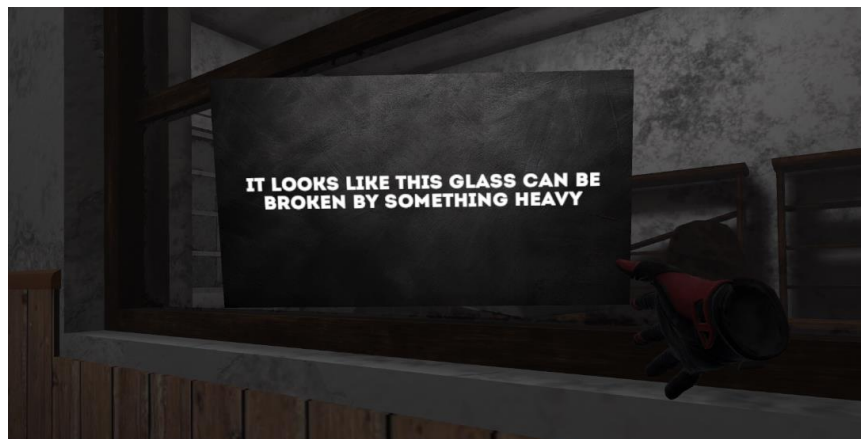


Рис 3.8 – Підказка у кімнаті охоронця

Досліджуючи локацію, гравець потрапляє до актової зали, де бачить проектор та повішений труп з молотком у голові.



Рис 3.9 - Актова зала

Прочитавши записку, головний герой розуміє, що треба спуститися у морг та дістати касету для відтворення відео. Після отримання касети за гравцем починає бігти медсестра. В цей момент активується `TriggerEventManager` який вмикає медсестру, у якої спрацьовує скрипт `AI_Monster`, що забезпечує переслідування гравця. Якщо медсестра наздожене гравця, гра перезапуститься з початку сцени.

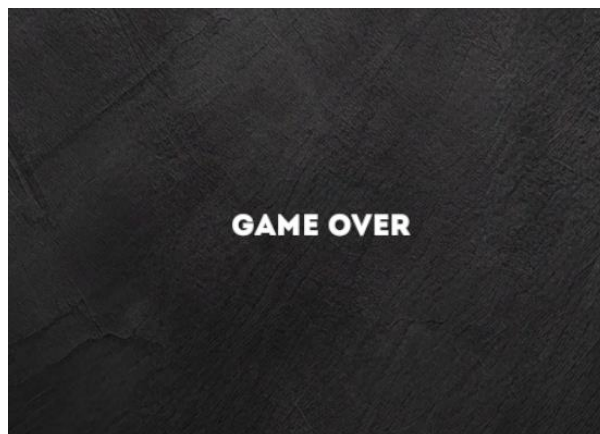


Рис 3.10 – Кінець гри

Втікши від неї, гравець вставляє касету у проектор, відтворюється страшне відео, після чого труп падає, що дає змогу забрати молоток і розбити вікно у кімнаті охоронця. Скрипт для падіння трупа (`TriggerEventManager`) активується після перегляду касети, викликаючи падіння трупа і доступ до молотка.

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



Рис 3.11 – Перегляд касети

Забравши у кімнаті охоронця фінальний ключ від головних дверей, гравець потрапляє до фінальної локації на вулицю лікарні, де його чекає поліція.

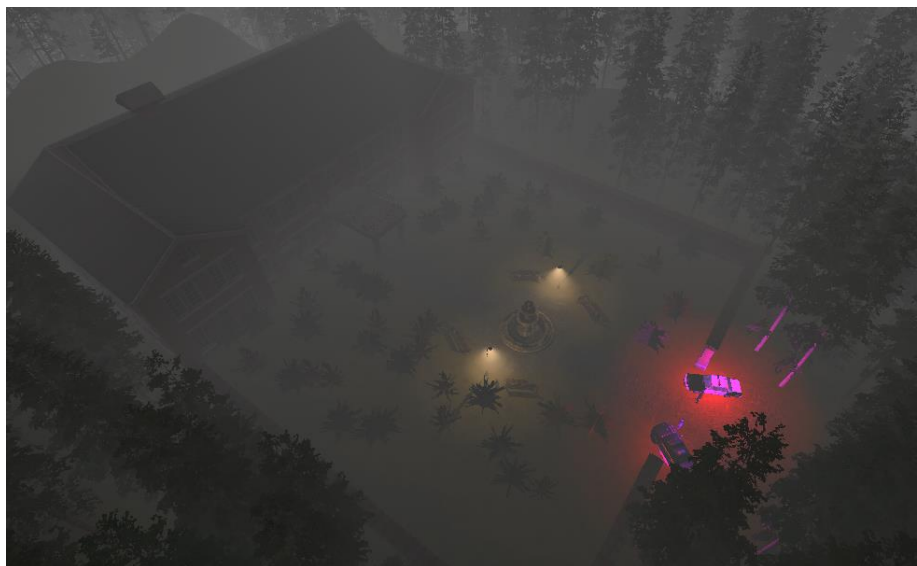


Рис 3.12 – Фінальна локація гри

Підійшовши до поліції, спрацьовує скрипт GameEndController, що відтворює канवास з фінальними словами у грі, де розповідається, що далі сталося з головним героєм. Після цього гравець знову опиняється у головному меню, що дає змогу заново почати гру.

**ALEX WAS ABLE TO ESCAPE FROM THE
ABANDONED ST. MARY'S ASYLUM.**

**THE POLICE ARRIVED ON TIME AND
SAVED HIM FROM THIS NIGHTMARE.**

a)

**ALEX WAS TAKEN TO THE POLICE
STATION, WHERE HE GAVE HIS
TESTIMONY ABOUT THE TERRIBLE
EVENTS THAT OCCURRED IN THE
HOSPITAL. HIS STORY WAS FULL OF
HORROR AND UNIMAGINABLE EVENTS
THAT SHOCKED INVESTIGATORS.**

**AFTER SEVERAL MONTHS OF
REHABILITATION IN A PSYCHIATRIC
CLINIC, ALEX BEGAN A NEW LIFE,**

б)

Рис 3.14 – Фінальні слова

a – перша частина, *б* – друга частина

Таким чином, гра "Fear Within" пропонує захоплюючий і напружений ігровий досвід, використовуючи VR для глибокого занурення в атмосферу жаху. Кожен скрипт і механіка гри сприяють створенню інтерактивного і реалістичного світу, де гравець може відчувати себе частиною сюжету і стати свідком страшних подій, що відбуваються в психіатричній лікарні.

3.2 Тестування роботи програмного забезпечення

Для забезпечення якості гри "Fear Within" було проведено ретельне тестування всіх її компонентів. Це тестування дозволило виявити та усунути можливі дефекти, а також переконатися в правильній роботі програмного забезпечення на різних етапах гри.

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

Результати тестування

Було протестовано основні сцени гри, включаючи головне меню, ігровий процес, взаємодію з об'єктами, активацію скриптів та підказок, а також кінцеву сцену. Тестування проводилось шляхом введення різних даних і порівняння отриманих результатів з очікуваними.

Таблиця 3.1

Результати тестування			
Сцена	Вхідні дані	Очікувані дані	Спостережені дані
Main Menu	Натиск на кнопку початку гри	Завантаження сцени Asylum	Завантаження сцени Asylum
	Налаштування графіки	Зміна параметрів графіки	Зміна параметрів графіки
	Переміщення персонажа	Гравець рухається	Гравець рухається
Asylum	Переміщення персонажа	Гравець рухається	Гравець рухається
	Звук переміщення персонажа	При ходьбі відтворюється звук, при зупинці переривається	Звуки відтворюються з затримкою
	Взаємодія з інтерактивними предметами	Предмети можна підняти, кинути	Предмети можна підняти, кинути
	Активація тригерів	Скрипт тригера активується, івенти спрацьовують	Скрипт тригера активується, івенти спрацьовують
	Скример медсестри	Скрипт активується, відтворюється анімація	Скрипт активується, в деяких випадках анімація медсестри не відтворювалася коректно
	Відкривання дверей	Двері відчиняються, відтворюється звук дверей	Двері відчиняються, відтворюється звук дверей

	Взаємодія з ключем	Взяти ключ, відчинити двері	Взяти ключ, відчинити двері
	Взаємодія з телевизором	Відтворення відео, показ підказки про дзвінок у поліцію	Відтворення відео, показ підказки про дзвінок у поліцію
	Активація підказки	В певній області активується підказка	У деяких місцях тригери для показу підказок не активувалися
	Взаємодія з проектором	Відтворення відео, падіння трупа, можливість взяти молоток	Відтворення відео, падіння трупа, можливість взяти молоток
	Отримання касети	Переслідування медсестрою, можливість уникнення її, гра перезапускається при контакті	Переслідування медсестрою, можливість уникнення її, гра перезапускається при контакті
	Смерть	Вмикається камера з екраном програшу	Вмикається камера з екраном програшу
	Використання молотка	Розбиття скла, активація телепортів, взяття фінального ключа	Розбиття скла, активація телепортів, взяття фінального ключа
	Телепортація	Гравець телепортується на певну позицію	Гравець телепортується на певну позицію
Street	Підхід до поліції	Відтворення фінальної сцени, показ канвасу з кінцевими словами, повернення в меню	Відтворення фінальної сцени, показ канвасу з кінцевими словами, повернення в меню

Під час тестування було виявлено кілька проблем, які потребували вирішення.

Однією з проблем була затримка відтворення звуків кроків. Виявлено, що звуки кроків відтворюються з затримкою, що знижувало реалістичність гри. Це питання було вирішено шляхом налаштування інтервалу відтворення звуків у

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

скрипті PlayerController, зокрема зменшенням значення параметру footstepInterval, що дозволило синхронізувати звуки кроків з рухом гравця.

Ще одна проблема стосувалася неактивних тригерів для показу підказок у деяких місцях гри. Це призводило до того, що гравці не отримували необхідної інформації в критичні моменти. Проблему було вирішено шляхом перевірки та виправлення колайдерів у скрипті VRHintTrigger, що забезпечило правильне розміщення та розміри тригерних зон.

Було також виявлено проблеми з анімацією медсестри. В деяких випадках анімація не відтворювалася коректно, що призводило до порушення ігрового процесу. Це було виправлено шляхом оновлення анімаційних файлів та коригування параметрів анімації у Unity, що забезпечило плавність і правильність анімацій.

Таким чином, тестування гри "Fear Within" показало, що всі основні компоненти програмного забезпечення працюють коректно, забезпечуючи належну якість ігрового процесу та задоволення від гри. Виявлені під час тестування дефекти були оперативно виправлені, що дозволило досягти стабільної роботи гри.

Функціональне тестування

Функціональне тестування було проведено для перевірки коректної роботи основних функцій гри, таких як переміщення персонажа, взаємодія з об'єктами, активація скриптів та тригерів, а також відтворення звукових та візуальних ефектів.

Чеклист для тестування переміщення персонажа

1. Перевірити плавне переміщення персонажа за допомогою стіка контролера.
2. Перевірити можливість телепортації у вказаних точках гри.
3. Перевірити коректність анімацій при переміщенні.
4. Перевірити відтворення звуків кроків під час руху персонажа.
5. Перевірити реакцію персонажа на перешкоди та колізії.

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

Тест-кейс для перевірки взаємодії з об'єктами

ID тест-кейсу	Опис	Кроки	Очікуваний результат	Фактичний результат	Статус
ТС_001	Перевірка підбирання предметів	1. Підійти до об'єкта. 2. Натиснути кнопку взаємодії.	Предмет піднімається та з'являється в руці персонажа.	Предмет піднімається та з'являється в руці персонажа.	Пройдено
ТС_002	Перевірка кидання предметів	1. Підняти предмет. 2. Кинути предмет.	Предмет кидається та взаємодіє з іншими об'єктами.	Предмет кидається та взаємодіє з іншими об'єктами.	Пройдено
ТС_003	Перевірка відкривання дверей	1. Підійти до дверей. 2. Натиснути кнопку взаємодії.	Двері відчиняються або зачиняються.	Двері відчиняються або зачиняються.	Пройдено

Опис проведеного тестування

Під час функціонального тестування було виявлено декілька проблем. Наприклад, було виявлено, що при переміщенні персонажа іноді виникають затримки, що негативно впливає на занурення у гру. Це було виправлено шляхом оптимізації коду та зменшення навантаження на процесор. Також було виявлено проблему з киданням предметів, яка була вирішена шляхом коригування компоненту Throwable на кожному предметі. Після усунення всіх виявлених дефектів гра була знову протестована, і всі функції працювали коректно.

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

Висновки до третього розділу

У третьому розділі було детально описано роботу з програмним додатком "Fear Within", включаючи основні механіки гри, сюжетні моменти та активацію скриптів. Було проведено функціональне тестування гри, розроблено чеклист та тест-кейс для перевірки роботи основних функцій гри. Результати тестування показали, що всі основні механіки гри працюють коректно, а виявлені дефекти були успішно усунуті.

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

ВИСНОВКИ

Ця кваліфікаційна робота була присвячена розробці комп'ютерної гри у жанрі horror для віртуальної реальності під назвою "Fear Within". У процесі виконання роботи було досягнуто основну мету – створення гри, яка забезпечує гравцям повне занурення у страхітливий світ психіатричної лікарні за допомогою сучасних VR технологій.

У першому розділі проведено детальний аналіз проблематики розробки VR гри у жанрі horror. Визначено ключові аспекти, які необхідно враховувати, включаючи психологічний вплив на гравців, технічні обмеження VR технологій та інтерактивність ігрового процесу. Було розглянуто аналогічні ігри та обрано відповідні інструменти для реалізації проекту, такі як Unity, SteamVR та гарнітура Pico 4.

У другому розділі розроблено дизайн-документ гри, що включає механіки управління головним героєм, реалізацію плавного переміщення та телепортації, а також взаємодію з об'єктами і розгадування головоломок. Було створено архітектуру ігрового додатку, що забезпечує ефективну роботу компонентів гри та оптимізацію продуктивності для стабільного геймплею у VR середовищі.

У третьому розділі проведено тестування гри "Fear Within". Описано ігровий процес, активацію скриптів і тригерів у ключових моментах сюжету. Тестування виявило ряд проблем, які були успішно усунені, що забезпечило стабільну роботу гри та покращило загальний геймплей.

Результати роботи підтвердили можливість створення високоякісного VR проекту у жанрі horror, який забезпечує гравцям захоплюючий та занурювальний ігровий досвід. Розроблена гра може бути використана як комерційний продукт, а також як навчальний матеріал для студентів, що вивчають розробку відеоігор та VR технологій. Результати дослідження можуть стати основою для подальших досліджень у сфері VR та геймдизайну.

Таким чином, виконання цієї роботи дозволило досягти поставлених цілей, забезпечити високу якість кінцевого продукту та підготувати проект "Fear Within" до релізу та подальшого комерційного використання.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ігрові жанри. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: <https://training.qatestlab.com/blog/technical-articles/games-genres/> (дата звернення: 14.02.2024).

2. Жанри комп'ютерних ігор: список. Класифікація комп'ютерних ігор за жанрами. UAEU Українська ТОП Газета. URL: <https://uaeu.top/digital-online/zhanri-komp-yuternikh-igor-spisok-klasifikatsiya-komp-yuternikh-igor-za-zhanrami.html> (дата звернення: 14.02.2024).

3. NSDG. How to Create a Horror Game in Unity. Sharp Coder Blog. URL: <https://www.sharpcoderblog.com/blog/how-to-make-a-horror-game-in-unity-3d> (date of access: 20.02.2024).

4. Project Tutorial: How to Make a Horror Game In Unity C#. Education Ecosystem Blog. URL: <https://educationecosystem.com/blog/project-tutorial-how-to-make-a-horror-game-in-unity-c/> (date of access: 20.02.2024).

5. SteamVR Plugin | Integration | Unity Asset Store. Unity Asset Store - The Best Assets for Game Making. URL: <https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647> (date of access: 29.02.2024).

6. Quickstart | SteamVR Unity Plugin. GitHub. URL: https://valvesoftware.github.io/steamvr_unity_plugin/articles/Quickstart.html (date of access: 01.03.2024).

7. Create with VR - Unity Learn. Unity Learn. URL: <https://learn.unity.com/course/create-with-vr> (date of access: 04.03.2024).

8. Unity - Manual: VR development in Unity. Unity - Manual: Unity User Manual 2022.3 (LTS). URL: <https://docs.unity3d.com/Manual/VROverview.html> (date of access: 05.03.2024).

9. Sketchfab. Sketchfab. URL: <https://sketchfab.com/feed> (date of access: 18.03.2024).

10. Introduction to Optimization in Unity - Unity Learn. *Unity Learn*. URL: <https://learn.unity.com/tutorial/introduction-to-optimization-in-unity> (date of access: 25.03.2024).
11. Unity - Manual: Understanding optimization in Unity. *Unity - Manual: Unity User Manual 2022.3 (LTS)*. URL: <https://docs.unity3d.com/Manual/UnderstandingPerformance.html> (date of access: 25.03.2024).
12. Mixamo. *Mixamo*. URL: <https://www.mixamo.com/#/> (date of access: 02.04.2024).
13. What is Use Case Diagram?. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> (date of access: 06.04.2024).
14. Unity - Manual: Use Animation Events. *Unity3d*. URL: <https://docs.unity3d.com/Manual/script-AnimationWindowEvent.html> (date of access: 10.04.2024).
15. Unity - Scripting API: Rigidbody. *Unity3d*. URL: <https://docs.unity3d.com/ScriptReference/Rigidbody.html> (date of access: 17.04.2024).
16. Unity - Scripting API: Coroutine. *Unity3d*. URL: <https://docs.unity3d.com/ScriptReference/Coroutine.html> (date of access: 22.04.2024).
17. Unity - Manual: Animator Controller. *Unity3d*. URL: <https://docs.unity3d.com/Manual/class-AnimatorController.html> (date of access: 25.04.2024).
18. EventSystems.EventTrigger - Unity Scripting API. *Unity3d*. URL: <https://docs.unity3d.com/ru/2019.3/ScriptReference/EventSystems.EventTrigger.html> (date of access: 01.05.2024).
19. Trigger Event | Spatial Creator Toolkit. *Spatial Creator Toolkit - Powered by Unity*. URL: <https://toolkit.spatial.io/docs/components/trigger-event> (date of access: 06.05.2024).

					KH.KP.Б – 122 – 24 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

20. Unity - Scripting API: SerializeField. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/SerializeField.html> (date of access: 10.05.2024).
21. Unity - Scripting API: Vector3. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/Vector3.html> (date of access: 18.05.2024).
22. Unity - Scripting API: SceneManager. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html> (date of access: 22.05.2024).
23. Unity - Scripting API: AudioSource. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/AudioSource.html> (date of access: 26.05.2024).
24. Canvas | Unity UI | 1.0.0. *Unity3d*.
URL: <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html> (date of access: 28.05.2024).
25. Unity - Scripting API: NavMesh. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/AI.NavMesh.html> (date of access: 02.06.2024).
26. Unity - Manual: Create a ragdoll. *Unity3d*.
URL: <https://docs.unity3d.com/Manual/wizard-RagdollWizard.html> (date of access: 04.06.2024).
27. Unity - Scripting API: EditorWindow. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/EditorWindow.html> (date of access: 05.06.2024).
28. Unity - Scripting API: Debug.Log. *Unity3d*.
URL: <https://docs.unity3d.com/ScriptReference/Debug.Log.html> (date of access: 11.06.2024).

ДОДАТКИ

Найбільш важливі функції

PlayerController:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;
using Valve.VR.InteractionSystem;

public class PlayerController : MonoBehaviour
{
    public SteamVR_Action_Vector2 input; // Вхідний вектор від контролера VR
    public float speed = 1.0f; // Швидкість руху гравця
    private CharacterController characterController; // Компонент CharacterController для
керування пересуванням
    private AudioSource footstepAudioSource; // Аудіоджерело для звуків кроків
    public AudioClip footstepSound; // Звуковий кліп для звуків кроків
    public float footstepInterval = 1.0f; // Інтервал між кроками
    private float lastFootstepTime; // Час останнього кроку
    private bool isMoving = false; // Чи рухається гравець

    private void Start()
    {
        // Ініціалізація CharacterController
        characterController = GetComponent<CharacterController>();

        // Додавання та налаштування AudioSource
        footstepAudioSource = gameObject.AddComponent<AudioSource>();
        footstepAudioSource.spatialBlend = 1.0f; // Просторовий звук
        footstepAudioSource.volume = 0.5f; // Гучність звуку
        footstepAudioSource.loop = true; // Повторення звуку
        footstepAudioSource.clip = footstepSound; // Призначення звукового кліпу для кроків
        footstepAudioSource.Play(); // Початок відтворення звуку відразу
        footstepAudioSource.Pause(); // Ставимо на паузу, тому що гравець не рухається

        lastFootstepTime = Time.time; // Ініціалізація часу останнього кроку
    }

    private void Update()
    {
        // Отримання напрямку руху на основі вхідного вектора від контролера VR
        Vector3 direction = Player.instance.hmdTransform.TransformDirection(new
        Vector3(input.axis.x, 0, input.axis.y));

        // Рух гравця з урахуванням гравітації
        characterController.Move(speed * Time.deltaTime * Vector3.ProjectOnPlane(direction,
        Vector3.up) - new Vector3(0, 9.81f, 0) * Time.deltaTime);

        // Визначення, чи рухається гравець
```

```

isMoving = direction.magnitude > 0 && characterController.isGrounded;

if (isMoving)
{
    footstepAudioSource.UnPause(); // Продовжуємо відтворення, якщо гравець рухається

    // Відтворення звуку кроків з інтервалом
    if (Time.time - lastFootstepTime > footstepInterval)
    {
        lastFootstepTime = Time.time;
    }
}
else
{
    footstepAudioSource.Pause(); // Ставимо на паузу відтворення, якщо гравець припиняє
    рух
}
}
}

```

TriggerEventManager

```

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

public class TriggerEventManager : MonoBehaviour
{
    // Клас для опису подій тригера
    [System.Serializable]
    public class TriggerEvent
    {
        public string requiredTag; // Тег, який повинен мати об'єкт, щоб тригер спрацював. Якщо
        порожній, спрацює як зазвичай.
        public GameObject[] gameObjectsToActivate; // Масив об'єктів, які потрібно активувати
        public GameObject[] gameObjectsToDeactivate; // Масив об'єктів, які потрібно
        деактивувати
        public AudioSource[] audioSourcesToPlay; // Масив аудіо джерел, які потрібно відтворити
        public float[] activationDelays; // Затримка для кожного об'єкта в gameObjectsToActivate
        public float[] deactivationDelays; // Затримка для кожного об'єкта в
        gameObjectsToDeactivate
        public float[] audioPlayDelays; // Затримка для кожного об'єкта в audioSourcesToPlay
        public float[] activationDurations; // Час, через який кожен об'єкт буде активований
        public float[] deactivationDurations; // Час, через який кожен об'єкт буде деактивований
    }

    public TriggerEvent[] triggerEvents; // Масив подій тригера

    // Метод, який викликається при вході в тригер
    private void OnTriggerEnter(Collider other)
    {

```

```

foreach (TriggerEvent triggerEvent in triggerEvents)
{
    // Перевіряємо, чи тег об'єкта відповідає потрібному тегу, якщо він заданий
    if (string.IsNullOrEmpty(triggerEvent.requiredTag) ||
other.CompareTag(triggerEvent.requiredTag))
    {
        // Активуємо об'єкти з затримкою та на вказаний час
        for (int i = 0; i < triggerEvent.gameObjectsToActivate.Length; i++)
        {
            GameObject obj = triggerEvent.gameObjectsToActivate[i];
            float delay = triggerEvent.activationDelays.Length > i ? triggerEvent.activationDelays[i] :
Of;
            float duration = triggerEvent.activationDurations.Length > i ?
triggerEvent.activationDurations[i] : Of;
            StartCoroutine(ActivateAfterDelay(obj, delay, duration));
        }

        // Деактивуємо об'єкти з затримкою та на вказаний час
        for (int i = 0; i < triggerEvent.gameObjectsToDeactivate.Length; i++)
        {
            GameObject obj = triggerEvent.gameObjectsToDeactivate[i];
            float delay = triggerEvent.deactivationDelays.Length > i ?
triggerEvent.deactivationDelays[i] : Of;
            float duration = triggerEvent.deactivationDurations.Length > i ?
triggerEvent.deactivationDurations[i] : Of;
            StartCoroutine(DeactivateAfterDelay(obj, delay, duration));
        }

        // Відтворюємо аудіо з затримкою
        for (int i = 0; i < triggerEvent.audioSourcesToPlay.Length; i++)
        {
            AudioSource audioSource = triggerEvent.audioSourcesToPlay[i];
            float delay = triggerEvent.audioPlayDelays.Length > i ? triggerEvent.audioPlayDelays[i] :
Of;
            StartCoroutine(PlayAudioAfterDelay(audioSource, delay));
        }
    }
}

// Корутіна для активації об'єкта з затримкою та на вказаний час
IEnumerator ActivateAfterDelay(GameObject obj, float delay, float duration)
{
    yield return new WaitForSeconds(delay);
    obj.SetActive(true);
    if (duration > 0)
    {
        yield return new WaitForSeconds(duration);
        obj.SetActive(false);
    }
}

```

```

    }

    // Корутіна для деактивації об'єкта з затримкою та на вказаний час
    IEnumerator DeactivateAfterDelay(GameObject obj, float delay, float duration)
    {
        yield return new WaitForSeconds(delay);
        obj.SetActive(false);
        if (duration > 0)
        {
            yield return new WaitForSeconds(duration);
            obj.SetActive(true);
        }
    }

    // Корутіна для відтворення аудіо з затримкою
    IEnumerator PlayAudioAfterDelay(AudioSource audioSource, float delay)
    {
        yield return new WaitForSeconds(delay);
        audioSource.Play();
    }
}

CanvasController
using UnityEngine;
using Valve.VR;
using System.Collections;

public class CanvasController : MonoBehaviour
{
    public SteamVR_Action_Boolean toggleCanvasAction; // Дія для перемикання канвасу
    public CanvasGroup hintCanvasGroup; // Група канвасу для керування прозорістю
    public float fadeDuration = 0.5f; // Тривалість плавної появи/загасання

    private bool canvasVisible = false; // Поточний стан канвасу

    private void Start()
    {
        if (hintCanvasGroup != null)
        {
            hintCanvasGroup.alpha = 0; // Спочатку приховуємо канвас
            hintCanvasGroup.interactable = false; // Відключаємо взаємодію
            hintCanvasGroup.blocksRaycasts = false; // Відключаємо блокування променів
        }
        else
        {
            Debug.LogError("Hint Canvas Group is not assigned.");
        }
    }

    private void Update()

```

```

{
    // Перевіряємо, чи була натиснута кнопка Y на контролері
    if (toggleCanvasAction.GetStateDown(SteamVR_Input_Sources.Any))
    {
        ToggleCanvas(); // Перемикаємо стан канвасу
    }
}

private void ToggleCanvas()
{
    canvasVisible = !canvasVisible; // Інвертуємо стан канвасу
    if (canvasVisible)
    {
        StartCoroutine(FadeIn());
    }
    else
    {
        StartCoroutine(FadeOut());
    }
}

private IEnumerator FadeIn()
{
    float elapsedTime = 0f;
    hintCanvasGroup.interactable = true; // Включаємо взаємодію
    hintCanvasGroup.blocksRaycasts = true; // Включаємо блокування променів
    while (elapsedTime < fadeDuration)
    {
        hintCanvasGroup.alpha = Mathf.Clamp01(elapsedTime / fadeDuration);
        elapsedTime += Time.deltaTime;
        yield return null;
    }
    hintCanvasGroup.alpha = 1;
}

private IEnumerator FadeOut()
{
    float elapsedTime = 0f;
    while (elapsedTime < fadeDuration)
    {
        hintCanvasGroup.alpha = Mathf.Clamp01(1 - (elapsedTime / fadeDuration));
        elapsedTime += Time.deltaTime;
        yield return null;
    }
    hintCanvasGroup.alpha = 0;
    hintCanvasGroup.interactable = false; // Відключаємо взаємодію
    hintCanvasGroup.blocksRaycasts = false; // Відключаємо блокування променів
}
}

```

DoorSoundController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR.InteractionSystem;

public class DoorSoundController : MonoBehaviour
{
    public CircularDrive circularDrive; // Посилання на компонент CircularDrive
    public AudioSource doorAudioSource; // Посилання на компонент AudioSource

    private float previousAngle; // Змінна для зберігання попереднього кута повороту
    private bool isSoundPlaying = false; // Змінна для відстеження стану звуку

    public float checkInterval = 0.1f; // Інтервал перевірки за секунди

    void Start()
    {
        // Якщо circularDrive не призначений в інспекторі, намагаємося знайти його на цьому
        об'єкті
        if (circularDrive == null)
        {
            circularDrive = GetComponent<CircularDrive>();
        }

        // Якщо doorAudioSource не призначений в інспекторі, намагаємося знайти його на
        цьому об'єкті
        if (doorAudioSource == null)
        {
            doorAudioSource = GetComponent<AudioSource>();
        }

        // Переконаємося, що doorAudioSource призначено
        if (doorAudioSource == null)
        {
            Debug.LogError("DoorSoundController: AudioSource component is missing.");
            return;
        }

        // Переконаємося, що звук спочатку не відтворюється
        doorAudioSource.Stop();
        Debug.Log("DoorSoundController: Initialized.");

        // Запускаємо перевірку, що повторюється, через заданий інтервал
        InvokeRepeating("CheckDoorMovement", 0, checkInterval);
    }

    void CheckDoorMovement()
    {

```

```

// Перевіряємо, чи призначений circularDrive і чи активний він
if (circularDrive == null || !circularDrive.isActiveAndEnabled)
{
    return;
}

// Отримуємо поточний кут повороту дверей
float currentAngle = circularDrive.outAngle;

// Перевіряємо, чи змінився кут повороту (двері рухаються)
if (Mathf.Abs(currentAngle - previousAngle) > 0.01f)
{
    // Якщо двері рухаються і звук не відтворюється, запускаємо звук
    if (!isSoundPlaying)
    {
        doorAudioSource.Play();
        isSoundPlaying = true;
        Debug.Log("DoorSoundController: Door is moving, playing sound.");
    }
}
else
{
    // Якщо двері не рухаються і звук відтворюється, ставимо його на паузу
    if (isSoundPlaying)
    {
        doorAudioSource.Pause();
        isSoundPlaying = false;
        Debug.Log("DoorSoundController: Door is not moving, pausing sound.");
    }
}

// Оновлюємо попередній кут повороту поточним кутом
previousAngle = currentAngle;
}
}

```

VRHintTrigger

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR.InteractionSystem;

public class VRHintTrigger : MonoBehaviour
{
    public CanvasGroup hintCanvasGroup; // Група Canvas для відображення підказки
    public float fadeDuration = 0.5f; // Тривалість анімації появи та зникнення

    // Метод, який викликається при старті
    private void Start()
    {

```

```

    if (hintCanvasGroup != null)
    {
        hintCanvasGroup.alpha = 0; // Встановлюємо початкову прозорість на 0 (приховано)
    }
    else
    {
        Debug.LogError("Hint Canvas Group is not assigned."); // Виводимо помилку, якщо Canvas
Group не призначено
    }
}

// Метод, який викликається при вході в тригер
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player")) // Перевіряємо, чи об'єкт має тег "Player"
    {
        if (hintCanvasGroup != null)
        {
            StartCoroutine(FadeIn()); // Запускаємо анімацію появи підказки
        }
    }
}

// Метод, який викликається при виході з тригера
private void OnTriggerExit(Collider other)
{
    if (other.CompareTag("Player")) // Перевіряємо, чи об'єкт має тег "Player"
    {
        if (hintCanvasGroup != null)
        {
            StartCoroutine(FadeOut()); // Запускаємо анімацію зникнення підказки
        }
    }
}

// Корутіна для анімації появи підказки
private IEnumerator FadeIn()
{
    float elapsedTime = 0f;
    while (elapsedTime < fadeDuration)
    {
        hintCanvasGroup.alpha = Mathf.Clamp01(elapsedTime / fadeDuration); // Поступово
збільшуємо прозорість
        elapsedTime += Time.deltaTime;
        yield return null;
    }
    hintCanvasGroup.alpha = 1; // Встановлюємо прозорість на 1 (видимо)
}

```

```

// Корутіна для анімації зникнення підказки
private IEnumerator FadeOut()
{
    float elapsedTime = 0f;
    while (elapsedTime < fadeDuration)
    {
        hintCanvasGroup.alpha = Mathf.Clamp01(1 - (elapsedTime / fadeDuration)); // Поступово
        зменшуємо прозорість
        elapsedTime += Time.deltaTime;
        yield return null;
    }
    hintCanvasGroup.alpha = 0; // Встановлюємо прозорість на 0 (приховано)
}
}

```

GameEndController

```

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

public class GameEndController : MonoBehaviour
{
    public GameObject playerToDeactivate; // Головний персонаж
    public GameObject activateCamera; // Об'єкт, який буде увімкнений (камера)
    public CanvasGroup firstEndGameCanvas; // Перший канвас
    public CanvasGroup secondEndGameCanvas; // Другий канвас
    public float firstDisplayTime = 5f; // Час відображення першого канвасу
    public float secondDisplayTime = 5f; // Час відображення другого канвасу
    public float fadeDuration = 1f; // Тривалість анімації згасання
    public string mainMenuSceneName = "MainMenu"; // Назва сцени головного меню

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            EndGame();
        }
    }

    private void EndGame()
    {
        // Відключаємо головного персонажа
        playerToDeactivate.SetActive(false);

        // Включаємо камеру
        activateCamera.SetActive(true);

        // Запускаємо корутину для відображення канвасів та переходу на сцену головного меню
        StartCoroutine(EndGameCoroutine());
    }
}

```

```

}

private IEnumerator EndGameCoroutine()
{
    // Включаємо перший канвас з анімацією появи
    yield return StartCoroutine(FadeCanvasGroup(firstEndGameCanvas, 0f, 1f));

    // Чекаємо на вказаний час
    yield return new WaitForSeconds(firstDisplayTime);

    // Відключаємо перший канвас з анімацією згасання
    yield return StartCoroutine(FadeCanvasGroup(firstEndGameCanvas, 1f, 0f));
    firstEndGameCanvas.gameObject.SetActive(false);

    // Включаємо другий канвас з анімацією появи
    secondEndGameCanvas.gameObject.SetActive(true);
    yield return StartCoroutine(FadeCanvasGroup(secondEndGameCanvas, 0f, 1f));

    // Чекаємо на вказаний час
    yield return new WaitForSeconds(secondDisplayTime);

    // Відключаємо другий канвас з анімацією згасання та переходимо на сцену головного
    меню
    yield return StartCoroutine(FadeCanvasGroup(secondEndGameCanvas, 1f, 0f));

    // Переходимо на сцену головного меню
    SceneManager.LoadScene(mainMenuSceneName);
}

private IEnumerator FadeCanvasGroup(CanvasGroup canvasGroup, float start, float end)
{
    float elapsedTime = 0f;
    canvasGroup.gameObject.SetActive(true);

    while (elapsedTime < fadeDuration)
    {
        elapsedTime += Time.deltaTime;
        canvasGroup.alpha = Mathf.Lerp(start, end, elapsedTime / fadeDuration);
        yield return null;
    }

    canvasGroup.alpha = end;
}
}

```