

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ЗАКЛАД
«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Навчально-науковий інститут математики
та інформаційних технологій


Кафедра інформаційних технологій та систем


Вередін Микола Олександрович

**МОДЕЛЮВАННЯ ІНТЕРАКТИВНОГО СЕРЕДОВИЩА У VR-ІГРАХ
ІЗ ЗАСТОСУВАННЯМ VR-ШОЛОМУ META QUEST 2**

**кваліфікаційна робота
здобувача вищої освіти другого (магістерського) рівня
освітньої програми «Мультимедійні системи»
за спеціальністю 121 „Інженерія програмного забезпечення”**

Особистий підпис  Микола ВЕРЕДІН

Науковий керівник  Світлана ПЕРЕЯСЛАВСЬКА,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій та
систем

Завідувач кафедри  Микола СЕМЕНОВ,
кандидат педагогічних наук,
доцент кафедри інформаційних
технологій та систем

АНОТАЦІЯ

Вередін М.О.

Тема: Моделювання інтерактивного середовища у VR-іграх із застосуванням VR-шолому Meta Quest 2.

Спеціальність: 121 «Інженерія програмного забезпечення».

Установа: ДЗ «ЛНУ імені Тараса Шевченка», 2026 р.

Магістерська робота містить: 99 с., 12 рис., 3 табл., 1 додат., 59 джерел.

Об'єкт дослідження – VR-ігри як програмні системи віртуальної реальності.

Предмет дослідження – методи та засоби моделювання інтерактивного ігрового середовища у VR-іграх із використанням рушія Unity.

Мета роботи — дослідити та обґрунтувати підходи до моделювання інтерактивного середовища у VR-іграх, реалізувати й перевірити модель фізичної взаємодії користувача з об'єктами VR із застосуванням Meta Quest 2.

Методи дослідження. *Загальнонаукові:* аналіз наукових публікацій і технічної документації з питань віртуальної реальності, аналіз підходів до моделювання інтерактивного середовища, синтез теоретичних положень щодо проектування VR-ігор. *Практичні та експериментальні:* проектування архітектури VR-гри, програмна реалізація ігрового середовища у рушії Unity, моделювання фізичної взаємодії з об'єктами, тестування VR-додатку на VR-шоломі Meta Quest 2, аналіз продуктивності та стабільності роботи гри.

Результати роботи. Досліджено основи віртуальної реальності та принципи створення інтерактивних VR-ігор. Проаналізовано апаратні й програмні засоби розробки для Meta Quest 2. Спроектовано та реалізовано VR-гру в середовищі Unity, тестування якої підтвердило коректність інтерактивних механік і можливість використання розробки як прикладу моделювання інтерактивного VR-середовища.

Ключові слова: віртуальна реальність, VR-ігри, інтерактивне середовище, фізична взаємодія, моделювання ігрового середовища, Unity, XR Interaction Toolkit, Meta Quest 2, VR-контролери.

ABSTRACT

Veredin M.O.

Title: Modeling of an Interactive Environment in VR Games Using the Meta Quest 2 VR Headset.

Specialty: 121 “Software Engineering”.

Institution: Luhansk Taras Shevchenko National University, 2026.

The master’s thesis contains: 99 pages, 12 figures, 3 tables, 1 appendix, 59 sources.

Object of the study – VR games as software systems of virtual reality.

Subject of the study – methods and tools for modeling an interactive game environment in VR games using the Unity game engine.

Purpose of the work – to investigate and substantiate approaches to modeling interactive environments in VR games, as well as to implement and experimentally validate a model of physical user interaction with virtual objects using the standalone Meta Quest 2 VR headset.

Research methods. General scientific methods: analysis of scientific publications and technical documentation on virtual reality, analysis of existing approaches to modeling interactive environments, synthesis of theoretical provisions related to VR game design. Practical and experimental methods: design of the VR game architecture, software implementation of the game environment using the Unity engine, modeling of physical interaction with objects, testing of the VR application on the Meta Quest 2 VR headset, analysis of the game’s performance and stability.

Results of the work. The fundamentals of virtual reality and the principles of creating interactive VR games were studied. Hardware and software development tools for Meta Quest 2 were analyzed. A VR game was designed and implemented in Unity, and testing confirmed the correctness of the interactive mechanics and the possibility of using the development as an example of modeling an interactive VR environment.

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЮВАННЯ ІНТЕРАКТИВНОГО СЕРЕДОВИЩА У VR-ІГРАХ.....	11
1.1. Віртуальна реальність та VR-ігри: основні поняття і сфери застосування	13
1.1.1. Поняття віртуальної реальності та її ключові характеристики....	15
1.1.2. VR-ігри як різновид інтерактивних програмних систем	16
1.2. Апаратно-програмне забезпечення для розробки VR-ігор.....	18
1.2.1. Апаратні компоненти VR-систем.....	19
1.2.2. Програмні інструменти для розробки VR-ігор.....	21
1.2.3. VR-шолом Meta Quest 2: архітектура та технічні можливості.....	23
1.3. Принципи моделювання інтерактивного середовища у VR-іграх.....	26
1.3.1. Інтерактивність та фізична взаємодія у віртуальному просторі..	28
1.3.2. Особливості проєктування ігрових сцен та об'єктів у VR.....	29
Висновки до розділу 1	32
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ VR-ГРИ ІНТЕРАКТИВНИМ СЕРЕДОВИЩЕМ	34
2.1. Постановка задачі та концепція моделювання інтерактивного VR- середовища.....	35
2.2. Структурна модель інтерактивного середовища VR-гри	37
2.3. Моделювання взаємодії користувача з об'єктами середовища	40
2.4. Проєктування ігрових сцен як інтерактивних просторів.....	42
2.5. Візуалізація моделі інтерактивного середовища.....	45
Висновки до розділу 2	48
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРАКТИВНИХ МЕХАНІК VR-ГРИ.....	50
3.1. Реалізація керування користувачем у VR-середовищі	51
3.2. Реалізація фізичної взаємодії з об'єктами у VR-середовищі	55
3.3. Реалізація інтерактивних механік	62

3.4. Тестування VR-гри на платформі Meta Quest 2	68
3.4.1. Перевірка працездатності та стабільності гри.....	69
3.4.2. Аналіз продуктивності VR-додатку	72
Висновки до розділу 3	75
ВИСНОВКИ.....	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	80
ДОДАТКИ.....	86
Додаток А. Вихідний код додатку.....	86

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ

VR – Virtual Reality

FPS – Frames Per Second

SDK – Software Development Kit

XR – Extended Reality

API – Application Programming Interface

GPU – Graphics Processing Unit

RAM – Random Access Memory

UI – User Interface

ВСТУП

Актуальність дослідження. Сучасний етап розвитку інформаційних технологій характеризується активним впровадженням засобів віртуальної реальності у різні сфери діяльності людини, зокрема у сферу розваг, освіти, тренажерних систем та комп'ютерних ігор. Віртуальна реальність забезпечує принципово новий рівень взаємодії користувача з цифровим середовищем за рахунок ефекту присутності та можливості безпосередньої фізичної взаємодії з віртуальними об'єктами. У цьому контексті VR-ігри виступають однією з найбільш технологічно складних та динамічно розвиваних категорій програмних продуктів.

Особливого значення у VR-іграх набуває **моделювання інтерактивного середовища**, оскільки саме інтерактивність визначає ступінь залученості користувача та реалістичність ігрового процесу. На відміну від традиційних комп'ютерних ігор, VR-ігри потребують точного поєднання тривимірної графіки, фізичного моделювання та обробки рухів користувача в реальному часі. Це висуває підвищені вимоги до методів проєктування і реалізації ігрового середовища, а також до оптимізації програмних рішень.

Серед сучасних VR-платформ значне поширення отримали автономні VR-шоломи, зокрема Meta Quest 2, які поєднують у собі апаратні засоби відображення, відстеження рухів та обчислювальні ресурси. Використання таких пристроїв розширює доступність VR-технологій, проте водночас накладає обмеження на продуктивність і потребує ретельного підходу до моделювання інтерактивного середовища та фізичної взаємодії об'єктів. У зв'язку з цим актуальним є дослідження практичних підходів до створення VR-ігор, адаптованих до можливостей автономних VR-шоломів.

Питанням віртуальної реальності, комп'ютерної графіки та інтерактивних систем присвячені праці таких вітчизняних і зарубіжних науковців, як М. В. Костюк, О. М. Спірін, В. В. Литвин, С. Лаваль, J. Jerald, M. Slater, S. Bryson, A. Steed та інших. У їх роботах розглядаються теоретичні засади VR, питання імерсії та взаємодії користувача з віртуальним

середовищем. Водночас значна частина наявних досліджень має загальнотеоретичний характер і не повною мірою охоплює практичні аспекти моделювання інтерактивного ігрового середовища у VR-іграх, розроблених з використанням сучасних ігрових рушіїв та автономних VR-пристроїв.

Недостатній рівень систематизованого аналізу методів моделювання інтерактивного середовища у VR-іграх, а також потреба у практичних прикладах реалізації фізичної взаємодії з об'єктами у середовищі Unity для VR-шолому Meta Quest 2 зумовили вибір теми магістерської роботи: **«Моделювання інтерактивного середовища у VR-іграх із застосуванням VR-шолому Meta Quest 2»**.

Об'єктом дослідження є VR-ігри як програмні системи віртуальної реальності.

Предмет дослідження – методи та засоби моделювання інтерактивного ігрового середовища у VR-іграх із використанням рушія Unity та VR-шолому Meta Quest 2.

Мета роботи — дослідити та обґрунтувати підходи до моделювання інтерактивного середовища у VR-іграх, а також реалізувати й експериментально перевірити модель фізичної взаємодії користувача з об'єктами віртуального простору із застосуванням автономного VR-шолому Meta Quest 2.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати теоретичні основи віртуальної реальності та принципи побудови інтерактивних VR-середовищ;
- дослідити апаратні та програмні засоби створення інтерактивних VR-застосунків рушія Unity;
- розробити концептуальну модель інтерактивного середовища VR-гри;
- спроектувати архітектуру VR-застосунку з урахуванням інтерактивних механік;
- реалізувати модель фізичної взаємодії користувача з об'єктами віртуального середовища;

- реалізувати інтерактивні елементи середовища, зокрема механіку фізичних кнопок та логіку відкривання дверей;
- провести тестування реалізованого інтерактивного VR-середовища на платформі Meta Quest 2 та проаналізувати отримані результати.

Методи дослідження. Наукові положення магістерської роботи базуються на використанні сукупності загальнонаукових та спеціальних методів, зокрема:

- аналізу наукових публікацій і технічної документації з питань віртуальної реальності та VR-ігор;
- синтезу теоретичних положень щодо моделювання інтерактивного середовища;
- моделювання ігрового середовища та взаємодії об'єктів у VR;
- проєктування програмної архітектури VR-гри;
- експериментального тестування VR-додатку на VR-шоломі Meta Quest 2;
- аналізу продуктивності та стабільності роботи гри.

Наукова новизна дослідження полягає у практичному обґрунтуванні підходів до моделювання інтерактивного середовища у VR-іграх для автономних VR-платформ. У роботі систематизовано методи реалізації фізичної взаємодії з об'єктами та запропоновано комплексне рішення для створення інтерактивного ігрового середовища у рушії Unity з урахуванням обмежень VR-шолому Meta Quest 2.

Практичне значення роботи полягає у можливості використання отриманих результатів та розробленої VR-гри як прикладу реалізації інтерактивного середовища у VR-іграх. Матеріали роботи можуть бути використані у навчальному процесі, а також при подальшій розробці VR-додатків і ігор для автономних VR-пристроїв.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЮВАННЯ ІНТЕРАКТИВНОГО СЕРЕДОВИЩА У VR-ІГРАХ

Віртуальна реальність на сьогодні є одним із найбільш динамічно розвиваних напрямів інформаційних технологій, що суттєво впливає на підходи до створення програмних продуктів та способи взаємодії людини з цифровим середовищем. Розвиток апаратного забезпечення, графічних технологій та засобів тривимірного моделювання сприяв широкому впровадженню VR-рішень у різні сфери діяльності, зокрема у комп'ютерні ігри, освітні платформи, симуляційні тренажери та віртуальні середовища для навчання і тренування (рис 1.1).

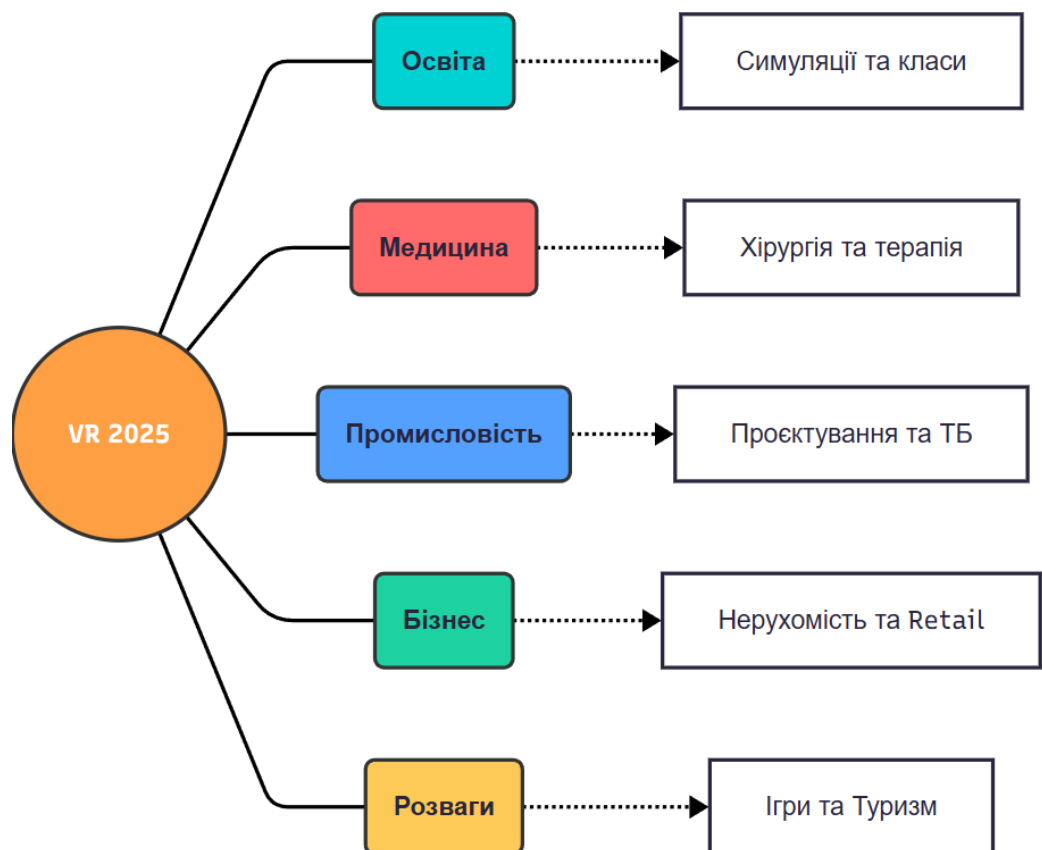


Рис. 1.1. Основні сфери застосування VR на 2025р.

Особливе місце серед VR-додатків займають VR-ігри, оскільки саме вони найбільш повно демонструють потенціал віртуальної реальності як

інтерактивного середовища. На відміну від традиційних ігор, VR-ігри передбачають безпосередню участь користувача у віртуальному просторі, де кожен рух голови або рук має відповідний відгук у цифровому середовищі. У такому форматі ключову роль відіграє не лише якість візуального відображення, але й коректне моделювання взаємодії з об'єктами, фізичних властивостей середовища та логіки поведінки ігрових елементів.

Зростання популярності автономних VR-пристроїв, зокрема VR-шоломів серії Meta Quest, значно розширило аудиторію користувачів віртуальної реальності. За рахунок поєднання в одному пристрої засобів відображення, відстеження рухів та обчислювальних ресурсів, такі шоломи забезпечують доступ до VR-контенту без необхідності використання зовнішніх комп'ютерів. Водночас автономність подібних пристроїв висуває додаткові вимоги до оптимізації графіки, фізичних розрахунків та інтерактивних механік у VR-іграх.

Однією з ключових проблем у розробці VR-ігор є **моделювання інтерактивного середовища**, у межах якого користувач може природно взаємодіяти з об'єктами віртуального простору. Інтерактивність у VR-іграх реалізується через фізичну взаємодію, захоплення та переміщення об'єктів, активацію елементів керування та зміну стану ігрового середовища у відповідь на дії гравця. Невідповідність поведінки віртуальних об'єктів очікуванням користувача суттєво знижує рівень імерсії та негативно впливає на ігровий досвід.

У зв'язку з цим особливої актуальності набуває дослідження теоретичних засад створення інтерактивних VR-середовищ, а також аналіз апаратних і програмних засобів, що використовуються у процесі розробки VR-ігор. Важливим інструментом у цій сфері є ігровий рушій Unity, який надає широкі можливості для створення тривимірних сцен, реалізації фізичної взаємодії та інтеграції з VR-платформами. Саме використання таких рушіїв дозволяє поєднати теоретичні підходи до моделювання інтерактивного середовища з їх практичною реалізацією.

Таким чином, розгляд теоретичних основ моделювання інтерактивного середовища у VR-іграх є необхідним етапом для подальшого проектування та реалізації VR-додатків. У межах даного розділу розглядаються базові поняття віртуальної реальності, особливості VR-ігор як інтерактивних систем, а також апаратно-програмні засоби, що забезпечують створення та функціонування інтерактивних VR-середовищ.

1.1. Віртуальна реальність та VR-ігри: основні поняття і сфери застосування

Термін «віртуальна реальність» (Virtual Reality, VR) вперше був запропонований в середині XX століття та відтоді набув широкого розвитку у сфері інформаційних технологій і розважальної індустрії. Віртуальна реальність визначається як «система, яка дозволяє користувачу взаємодіяти з комп'ютерно-згенерованим тривимірним середовищем у режимі реального часу, створюючи відчуття присутності» [3; 6]. Іншими словами, VR забезпечує повне занурення користувача у штучно створений простір, де він може виконувати різні дії та спостерігати їхні наслідки.

В Україні, як і у світі, термін «VR» поки що не має чіткої нормативної інтерпретації у законодавчих актах, проте в науковій літературі та практичній діяльності його тлумачення є досить однозначним. Науковці визначають VR як комплекс апаратних і програмних засобів, що забезпечують імітацію фізичного середовища та сенсорну взаємодію користувача з цим середовищем [17]. Основними компонентами VR-систем є: 3D-графіка, програмне забезпечення для відображення та взаємодії, пристрої відстеження рухів, а також інтерфейси для забезпечення тактильного та аудіо сприйняття [8; 9].

Особливе місце в сучасних VR-технологіях займають VR-ігри – інтерактивні цифрові продукти, призначені для занурення користувача у віртуальний світ. Вони поєднують елементи комп'ютерної графіки, гейміфікації, сенсорної взаємодії та сюжетного наповнення [5; 28]. Згідно з дослідженнями J. Jerald та P. Milgram, VR-ігри можна розглядати як окрему

підкатегорію ігрової індустрії, де основна мета полягає у створенні максимально реалістичного та інтерактивного середовища, що стимулює відчуття присутності та занурення [4; 15; 16].

Поширення VR-технологій у сучасному світі значно прискорилося завдяки появі масових VR-платформ, серед яких особливе значення має VR-шолом Meta Quest 2. Цей пристрій поєднує автономність, високу роздільну здатність дисплеїв, простий контроль рухів та можливість взаємодії без додаткових комп'ютерів [10; 11]. За даними компанії Meta, саме Meta Quest 2 відкрив нові горизонти для розробників VR-ігор, дозволяючи створювати інтерактивні середовища для різних жанрів, від симуляторів і пригодницьких ігор до освітніх і тренувальних застосунків.

Наукова спільнота активно досліджує різні аспекти VR-ігор. Так, А. Steuer пропонує оцінювати ефективність VR через параметри «присутності» та «іммерсивності», які визначають психологічний рівень занурення користувача у віртуальне середовище [20]. Дослідження R. Slater та M. Usoh показують, що чим вища інтерактивність та реалістичність середовища, тим сильніше проявляється ефект «присутності», що є ключовим показником якості VR-досвіду [2].

В Україні дослідження VR та VR-ігор також поступово набувають розвитку. Зокрема, дослідження VR-технологій здійснюються переважно у сфері освіти, тренажерних систем та симуляцій.

Таким чином, віртуальна реальність та VR-ігри є інтеграцією апаратних і програмних рішень, що забезпечують занурення користувача у штучно створене середовище. У контексті сучасних технологій VR-шоломи, зокрема Meta Quest 2, надають широкі можливості для створення інтерактивних середовищ, які застосовуються у розважальній, освітній та професійній сферах, а також слугують предметом активного наукового дослідження.

1.1.1. Поняття віртуальної реальності та її ключові характеристики

Віртуальна реальність (Virtual Reality, VR) — це технологія створення штучного середовища, яке імітує фізичний світ та дозволяє користувачу взаємодіяти з ним у режимі реального часу [1; 17]. Основна відмінність VR від традиційних комп'ютерних симуляцій полягає у високому рівні занурення та інтерактивності, що створює у користувача відчуття присутності («presence») у віртуальному просторі [3; 40].

Ключові характеристики VR визначаються поєднанням апаратних та програмних компонентів, а також принципами взаємодії користувача з середовищем. До основних характеристик належать:

1. **Імерсивність (Immersion)** — здатність системи створювати відчуття повного занурення користувача у віртуальне середовище. Рівень імерсивності залежить від якості графіки, поля зору, звукового супроводу та тактильного зворотного зв'язку [8; 28].

2. **Присутність (Presence)** — психологічний ефект сприйняття користувачем віртуального середовища як реального. Присутність формується через синхронізацію сенсорної інформації (зір, слух, дотик) та реакції користувача на події у VR [2; 54].

3. **Інтерактивність (Interactivity)** — здатність користувача виконувати дії у віртуальному середовищі, отримуючи миттєвий зворотний зв'язок. Інтерактивність забезпечується контролерами руху, сенсорними панелями та системами відстеження положення тіла [29; 53].

4. **Сенсорна адаптивність (Sensory Adaptation)** — можливість системи коректно реагувати на рухи та дії користувача, відтворюючи зміни середовища у відповідь на його поведінку. Це включає відстеження положення голови, рук, а також використання тактильних та аудіо ефектів [41].

5. **Реалістичність (Fidelity)** — рівень деталізації графічного, звукового та фізичного відтворення віртуального світу, що визначає якість візуального та сенсорного сприйняття користувача. Висока реалістичність підвищує ефект присутності та загальну ефективність VR-досвіду [25; 40].

Важливо зазначити, що VR-системи можуть бути як автономними, так і підключеними до комп'ютера або консолі. Автономні VR-шоломи, такі як Meta Quest 2, забезпечують інтегровану графіку, сенсори та обчислювальні потужності, що дозволяє користувачу взаємодіяти з середовищем без зовнішніх пристроїв [10; 11]. Підключені системи можуть надавати вищу якість графіки та складніші симуляції завдяки використанню ресурсів зовнішніх комп'ютерів, проте потребують додаткового обладнання.

Таким чином, віртуальна реальність є комплексною технологією, що поєднує апаратні та програмні засоби для створення інтерактивного середовища. Основні характеристики VR — імерсивність, присутність, інтерактивність, сенсорна адаптивність та реалістичність — визначають ефективність занурення та користувацький досвід, роблячи VR потужним інструментом для ігор, освіти та професійного тренування.

1.1.2. VR-ігри як різновид інтерактивних програмних систем

VR-ігри є специфічним різновидом інтерактивних програмних систем, основним завданням яких є створення повного занурення користувача у віртуальне середовище та забезпечення активної взаємодії з ним [30]. Від традиційних комп'ютерних ігор VR-ігри відрізняються високим рівнем імерсивності, присутності та сенсорної інтерактивності, що досягається завдяки використанню VR-шоломів, контролерів рухів та спеціальних трекінгових систем [3; 29].

Як інтерактивні програмні системи, VR-ігри мають комплексну архітектуру [6; 17], що включає:

1. **Графічний модуль** — забезпечує рендеринг тривимірного середовища з високою роздільною здатністю, реалістичним освітленням та анімацією об'єктів. Високоякісна графіка підвищує імерсивність та створює враження реалістичності середовища [25].

2. **Модуль взаємодії користувача** — відповідає за обробку рухів, дій та команд користувача. Включає відстеження положення голови, рук та контролерів, а також обробку жестів та команд голосу [41].

3. **Логічний модуль гри** — реалізує сценарії, правила та алгоритми поведінки об'єктів у середовищі. Саме цей компонент визначає, як середовище реагує на дії користувача та як розвивається ігровий процес [6; 17].

4. **Модуль звуку та тактильних ефектів** — формує аудіо та haptic-зворотний зв'язок, що сприяє ефекту присутності та покращує сприйняття віртуального середовища [8; 41].

VR-ігри класифікуються за різними критеріями, серед яких: жанр (екшн, пригодницькі, симулятори, освітні програми), рівень імерсивності (часткова або повна інтеграція з сенсорними системами), спосіб управління (контролери, рукавички, голосові команди) та платформа (автономні VR-шоломи або підключені до ПК системи) [5; 28].

Важливою особливістю VR-ігор є взаємозв'язок між інтерактивністю та присутністю користувача. Дослідження R. Slater та M. Usoh показують, що чим точніше система відтворює фізичні взаємодії та сенсорні сигнали, тим сильніше проявляється ефект занурення та підвищується користувацький досвід [20].

Meta Quest 2 є однією з найбільш поширених платформ для створення VR-ігор, забезпечуючи автономну роботу, точне відстеження рухів і можливість масштабування складності середовища. Це дозволяє розробникам створювати інтерактивні симуляції для розваг, освіти та професійного тренування [11; 52].

Таким чином, VR-ігри є інтерактивними програмними системами нового покоління, які поєднують графіку, звук, логіку та сенсорну взаємодію. Їх основні властивості — імерсивність, присутність та інтерактивність — визначають якість занурення користувача та ефективність використання VR-технологій у різних сферах [6].

1.2. Апаратно-програмне забезпечення для розробки VR-ігор

Розробка VR-ігор потребує комплексного поєднання апаратних і програмних компонентів, які забезпечують високий рівень занурення, інтерактивності та реалістичності середовища. Апаратно-програмне забезпечення (АПЗ) для VR-ігор визначає можливості створення інтерактивного контенту, його продуктивність та якість користувацького досвіду [6; 7].

Основною складовою АПЗ є апаратна платформа, що включає VR-шоломи, сенсорні контролери, системи відстеження рухів, високопродуктивні графічні процесори та комп'ютери. Вона відповідає за генерацію зображення, обробку рухів користувача та забезпечення зворотного зв'язку через аудіо та тактильні пристрої [8; 11]. Високі вимоги до апаратного забезпечення обумовлені необхідністю обробки великого обсягу даних у режимі реального часу [45; 46], що критично для підтримки ефекту присутності та мінімізації затримок у VR-середовищі.

Другим ключовим компонентом є **програмне забезпечення**, що включає ігрові рушії, SDK (Software Development Kit), API (Application Programming Interface), а також інструменти для моделювання, рендерингу, фізичної симуляції та оптимізації VR-контенту. Програмне забезпечення забезпечує логіку ігрового процесу, обробку взаємодії користувача з об'єктами середовища та інтеграцію апаратних даних від трекінгових систем [40].

Сучасні VR-платформи, такі як Meta Quest 2, поєднують в собі обидва аспекти АПЗ — автономність апаратної частини та широкі програмні можливості для розробників. Використання таких пристроїв дозволяє створювати VR-ігри, які не потребують підключення до зовнішніх комп'ютерів, зберігаючи при цьому високу якість графіки, точне відстеження рухів і ефективну взаємодію користувача з середовищем [11; 52].

Для ефективної розробки VR-ігор важливо забезпечити **сумісність апаратної та програмної частин**. Високопродуктивні графічні карти та

центральні процесори дозволяють рендерити складні тривимірні сцени та підтримувати стабільну частоту кадрів. SDK та рушії, такі як Unity або Unreal Engine, надають розробникам інструменти для реалізації фізики об'єктів, взаємодії користувача та оптимізації ресурсів [25; 28].

Крім того, в розробці VR-ігор важливу роль відіграють **системи відстеження рухів та контролери**. Вони дозволяють користувачу здійснювати точні дії у віртуальному середовищі, а розробникам — інтегрувати ці дії у логіку гри. Використання сенсорних рукавичок, трекінгових камер та акселерометрів підвищує рівень інтерактивності та створює більш природні сценарії взаємодії [11; 41].

Важливим аспектом є також **інтеграція аудіо та тактильного зворотного зв'язку**, яка дозволяє підвищити ефект присутності користувача. Просторове аудіо, вібраційні сигнали та тактильні ефекти формують повний сенсорний досвід та допомагають користувачу сприймати події у віртуальному середовищі більш реалістично [8; 41].

Таким чином, апаратно-програмне забезпечення для VR-ігор є комплексною системою, що поєднує високопродуктивні апаратні компоненти та програмні інструменти. Воно забезпечує ефективну генерацію та відображення віртуального середовища, інтерактивність, точне відстеження рухів користувача та високий рівень занурення, що робить VR-технології ефективним інструментом для розваг, освіти та професійного тренування [6; 17].

1.2.1. Апаратні компоненти VR-систем

Апаратні компоненти є основою функціонування VR-систем, забезпечуючи відтворення віртуального середовища та інтерактивність між користувачем і середовищем. Вони включають VR-шоломи, сенсорні контролери, системи відстеження рухів, обчислювальні пристрої та допоміжні периферійні пристрої [7; 8].

1. VR-шоломи (Head-Mounted Displays, HMD)

VR-шолом — головний пристрій, який відповідає за візуальне та частково аудіо сприйняття користувача. Основні характеристики шоломів включають:

- **Дисплей** з високою роздільною здатністю для кожного ока, що забезпечує реалістичне зображення та мінімізацію ефекту «екрану дверей» (screen-door effect).
- **Поле зору (Field of View, FOV)**, яке визначає кут охоплення зображення та безпосередньо впливає на ефект занурення.
- **Частота оновлення (Refresh Rate)**, що забезпечує плавність рухів та зменшує ймовірність виникнення нудоти користувача.
- **Вбудовані датчики** (гіроскопи, акселерометри, сенсори наближення), що дозволяють відстежувати положення голови та рухи користувача у просторі [17].

2. Контролери та пристрої відстеження рухів

Сенсорні контролери забезпечують безпосередню взаємодію користувача з об'єктами у VR-середовищі. Вони включають кнопки, тригери, сенсорні панелі та датчики положення рук. Контролери можуть передавати інформацію про позицію, орієнтацію та рухи пальців, що дозволяє реалізувати природну взаємодію [11; 41].

Системи трекінгу, у свою чергу, відстежують положення голови, рук, а іноді і всього тіла користувача у просторі. Вони бувають:

- **Оптичні (камери, інфрачервоні сенсори)** — визначають положення об'єктів за допомогою маркерів або LED-трекерів.
- **Інерційні (IMU, акселерометри, гіроскопи)** — визначають рухи користувача у просторі без зовнішніх камер.
- **Складені системи (inside-out / outside-in)** — поєднують оптичні та інерційні методи для точного трекінгу [47; 48]

3. Обчислювальні компоненти

Розробка та відтворення VR-ігор потребує значних обчислювальних ресурсів. До них належать:

- **Центральний процесор (CPU)**, що забезпечує обробку логіки гри, фізики та взаємодії об'єктів.
- **Графічний процесор (GPU)**, який відповідає за рендеринг 3D-сцен у реальному часі та підтримку високої частоти кадрів.
- **Оперативна пам'ять (RAM)** для швидкого доступу до даних та ресурсів середовища [3; 17].

4. Додаткові периферійні пристрої

Для підвищення рівня занурення використовуються додаткові пристрої:

- **Тактичні рукавички та вібраційні сенсори** для передачі відчуття дотику.
- **Просторові аудіосистеми** для формування об'ємного звуку.
- **Біометричні сенсори** для відстеження фізіологічної реакції користувача [8; 54].

Синергія цих апаратних компонентів забезпечує ефект високого рівня занурення, інтерактивності та присутності у VR-середовищі. Вибір та конфігурація апаратних компонентів визначає можливості створення VR-ігор з високою деталізацією, плавністю відображення та реалістичною взаємодією користувача з віртуальним світом [6; 7].

1.2.2. Програмні інструменти для розробки VR-ігор

Програмне забезпечення є ключовим компонентом створення VR-ігор, забезпечуючи логіку взаємодії, рендеринг тривимірних сцен, обробку даних від сенсорів та оптимізацію продуктивності. Розробка VR-контенту потребує поєднання спеціалізованих ігрових рушіїв, SDK, API та додаткових інструментів для моделювання, анімації та фізичного симулювання [6; 7].

1. Ігрові рушії

Ігрові рушії (Game Engines) є основою для розробки VR-ігор. Вони забезпечують інтеграцію графічного, фізичного та аудіо модулів, а також управління сценаріями взаємодії. Найпопулярнішими рушіями для VR є:

- **Unity** — забезпечує кросплатформену розробку, підтримку VR-шоломів (Meta Quest 2, HTC Vive, Valve Index), інтеграцію з SDK та плагінами для відстеження рухів і фізики. Unity надає інструменти для створення тривимірних сцен, анімації об'єктів, просторового аудіо та інтерактивного UI [9; 18].

- **Unreal Engine** — дозволяє створювати VR-ігри з високою реалістичністю графіки завдяки рушійу рендерингу на основі фізично коректного освітлення (PBR) та підтримці Blueprints для візуального програмування. Unreal Engine забезпечує високу продуктивність та гнучкість у створенні інтерактивних середовищ [10].

2. SDK та API для VR-платформ

Для взаємодії з апаратними компонентами VR-шоломів використовуються спеціалізовані SDK (Software Development Kit) та API (Application Programming Interface). Вони дозволяють обробляти дані трекінгу, контролерів та сенсорів, а також забезпечують доступ до вбудованих функцій пристроїв [11; 41]. Для Meta Quest 2 найбільш поширеними є:

- **Oculus SDK** — надає інструменти для відстеження рухів, рендерингу та інтеграції з VR-рушіями [41; 42].

- **OpenXR** — стандартний кросплатформний API, що дозволяє розробникам створювати VR-додатки, сумісні з різними VR-пристроями [12; 43].

3. Інструменти 3D-моделювання та анімації

Створення контенту для VR-ігор неможливе без програм для тривимірного моделювання та анімації. Найпоширеніші інструменти:

- **Blender** — безкоштовна платформа для моделювання, текстурування та анімації 3D-об'єктів.

- **Maya та 3ds Max** — комерційні інструменти для професійного 3D-моделювання та створення складних анімацій [13; 14].

4. Інструменти для фізичної симуляції та оптимізації

Для реалізації реалістичної поведінки об'єктів та фізики середовища використовуються фізичні рушії:

- **PhysX** — рушій фізики від NVIDIA, інтегрований у Unity та Unreal Engine, забезпечує моделювання гравітації, зіткнень та динаміки об'єктів [56].
- **Havok Physics** — комерційний фізичний рушій для високоякісного моделювання руху та взаємодії об'єктів у VR [57].

5. Інструменти для аудіо та інтерактивного UI

Просторове аудіо та користувацький інтерфейс є важливими елементами VR-ігор:

- **FMOD та Wwise** — інструменти для створення просторового звуку та інтерактивної музики [58].
- **VR Interaction Framework та VRTK** — набори інструментів для інтерактивного управління об'єктами, жестами та контролерами.

Таким чином, програмні інструменти для розробки VR-ігор забезпечують комплексне середовище для створення інтерактивних 3D-сцен, обробки даних від сенсорів та контролерів, а також інтеграції графіки, звуку та фізики. Поєднання рушіїв, SDK, API та додаткових програмних засобів дозволяє створювати VR-ігри високої якості, які ефективно використовують можливості сучасних апаратних платформ.

1.2.3. VR-шолом Meta Quest 2: архітектура та технічні можливості

Meta Quest 2 є однією з найбільш поширених автономних VR-платформ, яка поєднує апаратні та програмні компоненти для створення високоякісного VR-досвіду. Шолом розроблений компанією Meta (раніше Oculus) і призначений для використання як у розважальних, так і в освітніх та професійних застосунках [51].

1. Архітектура апаратної частини

Meta Quest 2 має інтегровану апаратну платформу, що включає дисплеї, процесори, датчики та акумуляторну систему:

- **Дисплеї:** OLED-панелі з роздільною здатністю 1832×1920 пікселів на око, що забезпечує чітке та контрастне зображення. Поле зору складає приблизно 90° – 100° , що сприяє ефекту занурення. Частота оновлення екрану може досягати 120 Гц, що зменшує відчуття розмитості рухів.
- **Процесор та графічний чип:** Qualcomm Snapdragon XR2 з вбудованим GPU забезпечує обробку складної 3D-графіки та виконання VR-додатків у реальному часі без підключення до зовнішнього комп'ютера. Цей чип підтримує багатозадачність та високий рівень оптимізації для VR-середовищ.
- **Датчики:** комбінація IMU (акселерометр, гіроскоп), сенсорів наближення та камер для відстеження рухів користувача у просторі. Вбудована система **inside-out tracking** дозволяє відстежувати положення користувача та контролерів без використання зовнішніх базових станцій.
- **Акумулятор:** літієво-іонна батарея забезпечує автономну роботу шолома протягом 2–3 годин при інтенсивному використанні, з можливістю підзарядки через USB-C.

2. Контролери та системи взаємодії

Meta Quest 2 комплектується бездротовими контролерами Oculus Touch, що забезпечують:

- точне відстеження положення та орієнтації рук у 3D-просторі;
- наявність кнопок, тригерів та сенсорних панелей для реалізації різних сценаріїв взаємодії;
- вібраційний зворотний зв'язок (haptics) для підвищення ефекту присутності.

Система відстеження підтримує повну інтеграцію з VR-ігровими рушіями, забезпечуючи природні та точні рухи користувача у середовищі. Детальніше архітектуру шолому Meta Quest 2 описано у [59].

3. Програмні можливості та інтеграція

Meta Quest 2 підтримує роботу з основними VR-рушіями, такими як Unity та Unreal Engine, а також сумісна з SDK Oculus та стандартом OpenXR

[49; 50]. Це дозволяє розробникам створювати кросплатформені VR-додатки та адаптувати ігровий контент під автономні шоломи без додаткового комп'ютерного обладнання.

Ключові програмні можливості включають:

- підтримку високоякісного 3D-графічного рендерингу та просторового аудіо;
- обробку даних трекінгу у реальному часі для забезпечення плавної взаємодії;
- інтеграцію систем захисту та оптимізації продуктивності для стабільного функціонування VR-додатків.

4. Переваги Meta Quest 2

- автономність роботи та простота використання без додаткового обладнання;
- високий рівень імерсивності завдяки високій роздільній здатності дисплея та точному трекінгу;
- підтримка широкого спектру VR-додатків, включаючи ігри, освітні симуляції та професійні тренажери;
- інтеграція з програмними інструментами для швидкої розробки та тестування VR-контенту [44; 52].

Таким чином, Meta Quest 2 є потужною та універсальною платформою для розробки VR-ігор, яка поєднує автономну апаратну архітектуру, точні системи відстеження рухів, високоякісний графічний та аудіо відтворювач, а також широкий спектр програмних інструментів для створення інтерактивного контенту. Ця платформа забезпечує високу якість занурення та дозволяє розробникам реалізовувати складні сценарії взаємодії у віртуальному середовищі.

1.3. Принципи моделювання інтерактивного середовища у VR-іграх

Моделювання інтерактивного середовища у VR-іграх є ключовим етапом розробки, що визначає рівень занурення користувача, ефективність взаємодії та реалістичність віртуального світу. Основною метою є створення середовища, в якому користувач може вільно переміщуватися, взаємодіяти з об'єктами та відчувати присутність, що максимально наближена до реального досвіду [18; 19].

1. Принцип імерсивності та присутності

Ефект імерсивності визначається тим, наскільки користувач відчуває себе частиною віртуального світу. Присутність забезпечується через поєднання високоякісної графіки, точного трекінгу рухів користувача, просторового аудіо та тактильного зворотного зв'язку. Чим точніше система відтворює фізичні взаємодії та сенсорні сигнали, тим сильніше проявляється ефект занурення [18; 21].

2. Принцип інтерактивності

Інтерактивність забезпечує можливість активної взаємодії користувача з об'єктами середовища. Основні аспекти інтерактивності включають:

- об'єкти, що реагують на дії користувача (натискання, переміщення, підйом, кидок);
- адаптивну поведінку NPC (неігрових персонажів) та середовища;
- сценарії подій, що змінюються залежно від дій користувача [22].

3. Принцип реалістичності фізики та поведінки об'єктів

Реалістична фізика забезпечує природну поведінку об'єктів, їх зіткнення, гравітацію та динаміку руху. Для цього використовуються фізичні рушії (PhysX, Havok), що дозволяють моделювати взаємодію об'єктів з урахуванням маси, сили, тертя та інших фізичних параметрів. Реалістична фізика підвищує правдоподібність середовища та сприяє більш природній взаємодії користувача [22; 24].

4. Принцип масштабованості та оптимізації

VR-середовище повинно забезпечувати високу продуктивність на різних апаратних конфігураціях. Оптимізація включає зменшення кількості полігонів, використання LOD-моделей (Level of Detail), ефективне управління текстурами та рендерингом. Масштабованість дозволяє розробникам адаптувати VR-контент під автономні шоломи, ПК та інші платформи без втрати якості занурення [6; 26].

5. Принцип когнітивної зручності та ергономіки

Інтерфейс користувача у VR-іграх повинен бути зрозумілим та природним. Важливими аспектами є:

- розташування елементів UI у полі зору користувача;
- інтуїтивні жести для взаємодії з об'єктами;
- відсутність перевантаження інформацією, що може викликати когнітивну втоми [28].

6. Принцип адаптивності та сценарного моделювання

Для підвищення інтерактивності та динамічності середовища застосовуються адаптивні алгоритми, які змінюють поведінку NPC, об'єктів та освітлення залежно від дій користувача. Це дозволяє створювати унікальні сценарії для кожного сеансу та підвищує реалістичність взаємодії.

7. Принцип багатосенсорної інтеграції

Ефективне моделювання VR-середовища передбачає синхронізацію різних сенсорних каналів:

- візуального (графіка, світло, тіні);
- аудіального (просторове аудіо, звукові ефекти);
- тактильного (вібрація, haptics);
- рухового (відстеження положення голови та рук).

Багатосенсорна інтеграція забезпечує глибокий ефект занурення та реалістичність сприйняття [18; 31; 32].

Таким чином, моделювання інтерактивного середовища у VR-іграх ґрунтується на принципах імерсивності, інтерактивності, реалістичності фізики, оптимізації та багатосенсорної інтеграції. Дотримання цих принципів

дозволяє створювати VR-додатки високого рівня занурення, що забезпечують ефективну взаємодію користувача з віртуальним світом, а також відповідають сучасним вимогам як до розваг, так і до освітніх та професійних симуляцій.

1.3.1. Інтерактивність та фізична взаємодія у віртуальному просторі

Як зазначають дослідники, «одним із найважливіших аспектів віртуальної реальності є інтерактивність, яка визначає здатність користувача активно взаємодіяти з об'єктами і середовищем у цифровому просторі» — це дозволяє не лише спостерігати, а й безпосередньо впливати на віртуальне середовище та отримувати зворотний зв'язок у реальному часі [18; 21; 33].

Інтерактивність у VR-іграх є складним психофізичним процесом, що включає взаємодію *користувача – апаратний пристрій – програмне середовище*. Саме ця взаємодія забезпечує активну участь користувача в симульованому світі, створюючи ефект присутності та контроль над подіями у віртуальному просторі. Інтерактивність реалізується як через візуальні й аудіальні реакції на дії користувача, так і через фізичну взаємодію із віртуальними об'єктами за допомогою контролерів, трекінгу рухів та технологій тактильного зворотного зв'язку (haptics). [22; 34]

Фізична взаємодія — це здатність користувача не лише сприймати віртуальне середовище, а й фізично взаємодіяти з ним через рухи тіла, маніпуляції руками або іншими ввідними пристроями. Завдяки трекінговим системам положення голови, рук та тіла визначається у тривимірному просторі, що дозволяє відтворювати природні рухи користувача у віртуальний контекст. Сучасні VR-системи підтримують як *традиційні контролери*, так і дані щодо просторового позиціонування, включаючи hand-tracking (відстеження рук без контролерів) для більш природних жестів, що підвищує якість сприйняття та рівень присутності. [35; 36]

Фізична взаємодія у VR також пов'язана з концепцією сенсомоторної інтеграції: дії користувача (рухи тіла, жести) перетворюються на зміни у віртуальному середовищі, а реакції середовища (зміна позиції об'єктів, звукові

ефекти, тактильний зворотний зв'язок) повертаються до користувача, створюючи циклічний контур взаємодії. Такий сенсомоторний зв'язок підсилює ефект *embodiment* (втілення), коли користувач відчуває своє тіло віртуально представленим та діє у взаємодії з об'єктами на основі реальних рухів. [19; 21; 37]

Ключовими компонентами інтерактивності та фізичної взаємодії у VR-іграх є:

- **Ввідні пристрої** — контролери, рукавички, сенсори трекінгу, що забезпечують точне відображення рухів користувача у віртуальному просторі.
- **Візуальний та аудіальний зворотний зв'язок** — реалістичні зміни у сцені, які відповідають діям користувача.
- **Фізичні властивості об'єктів** — моделі поведінки, що імітують фізичні закони (гравітація, зіткнення, інерція), забезпечуючи природну взаємодію з віртуальними об'єктами.
- **Haptics і тактильний зворотний зв'язок** — передача відчуттів дотику та опору, що робить взаємодію більш наближеною до реального сприйняття.

Детальніше описано у джерелах [23; 24; 34]

Інтерактивність і фізична взаємодія відіграють ключову роль у підвищенні *занурення* та *присутності*: активна взаємодія з VR-середовищем не лише збільшує ефективність користувацького досвіду, а й сприяє кращому навчанню, прийняттю рішень та емоційному залученню у віртуальні сценарії. Такий багатовимірний підхід до взаємодії є одним із фундаментальних принципів моделювання інтерактивних VR-середовищ, що забезпечує їхню ефективність як у розважальних, так і у професійних застосуваннях [27; 38]

1.3.2. Особливості проєктування ігрових сцен та об'єктів у VR

Проєктування ігрових сцен та об'єктів у VR-іграх має значні відмінності від традиційного 2D чи 3D-геймдизайну через необхідність забезпечення повного занурення користувача, високої інтерактивності та відповідності

фізичним законам у віртуальному просторі [18; 39]. Як зазначає А. Slater, «ефект присутності у віртуальному середовищі залежить не лише від якості графіки, а й від правильного проєктування об'єктів та взаємодії з ними [21]»

Проєктування VR-сцен передбачає врахування таких основних аспектів:

- **Масштабування та пропорції об'єктів**, що забезпечує правильне відчуття простору;
- **Оптимізація полігональної структури та текстур**, для підтримки стабільного фреймрейту;
- **Розташування інтерактивних об'єктів**, що дозволяє користувачу легко взаємодіяти без дискомфорту;
- **Фізична коректність поведінки об'єктів**, щоб реакція на дії користувача виглядала природною;
- **Сценарна композиція та навігація**, що забезпечує логічний порядок подій і зручність переміщення у VR-середовищі [26;].

Для наочності порівняємо ключові особливості проєктування ігрових об'єктів у VR з традиційним 3D-дизайном у таблиці:

Таблиця 1.1

Порівняльна таблиця особливостей проєктування об'єктів у VR та традиційних 3D-іграх

Параметр	Традиційний 3D-дизайн	VR-дизайн
Масштаб об'єктів	Орієнтований на екран монітора; масштаб часто умовний	Орієнтований на реальне відчуття простору; точні пропорції критично важливі
Інтерактивність	Обмежена взаємодія через мишу та клавіатуру	Висока інтерактивність через контролери, трекінг рухів та hand-tracking
Оптимізація	Оптимізація графіки для зручного рендерингу на ПК/консолях	Обов'язкова висока оптимізація для підтримки стабільного FPS у VR (зазвичай ≥ 90)
Фізична поведінка	Фізика може бути умовною, часто стилізована	Реалістична фізика об'єктів для правильного відчуття взаємодії та занурення

Параметр	Традиційний 3D-дизайн	VR-дизайн
Навігація	Переміщення через клавіатуру або контролер	Природна навігація користувача у тривимірному просторі з урахуванням обмежень VR
Сенсорний зворотний зв'язок	Відсутній або обмежений	Тактильний, аудіальний та візуальний зворотний зв'язок для підвищення присутності
Композиція сцени	Орієнтована на плоский екран та композиційні правила камери	Орієнтована на очі користувача, його поле зору та перспективу у просторі

Джерело: узагальнено за [18; 26; 39; 41]. Як ми бачимо, проєктування об'єктів у VR-іграх значно відрізняється від традиційного 3D-дизайну. У VR критично важливими є реалістичні пропорції об'єктів, висока інтерактивність та точне відтворення фізичної поведінки, тоді як у традиційних 3D-іграх ці параметри часто умовні і орієнтовані на екран монітора. Також у VR особлива увага приділяється сенсорному зворотному зв'язку та природній навігації користувача, що забезпечує ефект присутності та комфортне занурення у віртуальне середовище. Проєктування VR-сцен також передбачає використання принципу **«центр уваги користувача»**, коли важливі об'єкти та ключові елементи розташовуються у зоні прямого спостереження користувача. Додатково необхідно враховувати обмеження VR-платформи, такі як дальність трекінгу контролерів, площа фізичного простору користувача та потенційні проблеми з морською хворобою (VR sickness) [55].

Таким чином, проєктування об'єктів і сцен у VR-іграх потребує інтегрованого підходу, який поєднує фізичну правдоподібність, високу інтерактивність, оптимізацію продуктивності та ергономічну навігацію. Лише комплексне врахування цих аспектів дозволяє створювати VR-додатки, що забезпечують глибоке занурення користувача та природну взаємодію з віртуальним середовищем.

Висновки до розділу 1

У першому розділі дипломної роботи було розглянуто ключові теоретичні та технічні аспекти створення інтерактивних VR-середовищ та VR-ігор. Проведений аналіз дозволив сформулювати наступні висновки:

1. **Поняття віртуальної реальності та VR-ігор.** Віртуальна реальність визначається як технологія створення ілюзії присутності користувача у штучно змодельованому тривимірному середовищі, що підтримує інтерактивну взаємодію та сенсомоторний зворотний зв'язок. VR-ігри є різновидом інтерактивних програмних систем, які поєднують візуальні, аудіальні та фізичні компоненти для створення високого рівня занурення та активної участі користувача.

2. **Апаратно-програмне забезпечення для VR.** Для створення VR-ігор необхідні як потужні апаратні компоненти (VR-шоломи, контролери, системи трекінгу, комп'ютерне обладнання), так і програмні інструменти (движки Unity та Unreal Engine, SDK для VR, системи фізичного моделювання). Особлива увага приділяється VR-шолому Meta Quest 2, який забезпечує автономну роботу, високу роздільну здатність, трекінг рухів користувача та підтримку hand-tracking.

3. **Принципи моделювання інтерактивного середовища.** Моделювання VR-середовища ґрунтується на принципах інтерактивності, фізичної взаємодії та реалізації присутності. Інтерактивність у VR включає циклічний зв'язок «користувач – середовище», де дії користувача впливають на віртуальні об'єкти, а система надає зворотний сенсорний, візуальний та аудіальний відгук.

4. **Проектування сцен і об'єктів у VR.** Проектування VR-сцен та об'єктів має враховувати масштаб, пропорції, фізичну поведінку об'єктів, оптимізацію графіки, логіку навігації та розташування інтерактивних елементів. Як показано у порівняльній таблиці, VR-дизайн відрізняється від традиційного 3D-дизайну підвищеною інтерактивністю, природністю рухів та підтримкою сенсорного зворотного зв'язку, що забезпечує ефект присутності.

5. Висновок загального характеру. Розуміння апаратних та програмних компонентів VR, принципів моделювання інтерактивного середовища та особливостей проєктування ігрових сцен є фундаментом для ефективної розробки VR-ігор. Усі розглянуті аспекти взаємопов'язані: правильний вибір обладнання, оптимізація програмного середовища та логіка взаємодії користувача визначають якість занурення, продуктивність системи та рівень інтерактивності, що є ключовими показниками успішності сучасних VR-додатків.

Таким чином, перший розділ створює теоретичну та методологічну базу для подальшого дослідження та практичної реалізації інтерактивного VR-середовища у VR-іграх із використанням шолома Meta Quest 2.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ VR-ГРИ З ІНТЕРАКТИВНИМ СЕРЕДОВИЩЕМ

У попередньому розділі було розглянуто теоретичні основи віртуальної реальності, особливості VR-ігор як інтерактивних програмних систем, а також апаратно-програмні засоби, що застосовуються для створення VR-додатків. Отримані теоретичні положення є підґрунтям для практичної реалізації VR-гри з інтерактивним середовищем, орієнтованої на використання автономного VR-шолому Meta Quest 2.

Даний розділ присвячений процесу проєктування та реалізації VR-гри жанру *escape room*, у якій основний акцент зроблено на фізичній взаємодії користувача з об'єктами віртуального середовища. Особливістю розроблюваної гри є поєднання масштабного ігрового простору з можливістю маніпуляції великими об'єктами, що зумовлено ігровою концепцією та безпосередньо впливає на архітектуру проєкту й підходи до моделювання середовища.

У межах розділу розглядаються етапи формування ігрової концепції, визначення функціональних вимог та проєктування архітектури VR-гри у середовищі Unity. Окрему увагу приділено організації ігрових сцен, моделюванню віртуальних приміщень та налаштуванню фізичних властивостей об'єктів, які забезпечують можливість їх захоплення, переміщення та використання у процесі проходження гри.

Реалізація VR-гри здійснюється з урахуванням технічних обмежень автономного VR-шолому Meta Quest 2, що вимагає оптимального використання обчислювальних ресурсів та коректного налаштування фізичних і графічних параметрів. У процесі розробки особлива увага приділяється забезпеченню стабільної роботи додатку, природної взаємодії користувача з віртуальним середовищем та збереженню відчуття імерсії.

Таким чином, у другому розділі здійснюється перехід від теоретичного аналізу до практичної реалізації VR-гри, яка виступає прикладом

моделювання інтерактивного середовища у VR-іграх. Результати, отримані на цьому етапі, є основою для подальшого аналізу інтерактивних механік та тестування розробленого VR-додатку

2.1. Постановка задачі та концепція моделювання інтерактивного VR-середовища

У межах магістерської роботи основною метою є дослідження та практична реалізація процесу моделювання інтерактивного середовища у VR-іграх із використанням автономного VR-шолому Meta Quest 2. Розроблена VR-гра не розглядається як самостійний комерційний продукт, а використовується як **експериментальне середовище**, у якому реалізуються та перевіряються принципи інтерактивності, фізичної взаємодії та просторової організації віртуального світу.

Постановка задачі полягає у створенні такого віртуального середовища, де ігровий процес формується безпосередньо через взаємодію користувача з об'єктами простору, а не через заздалегідь визначені сценарії або абстрактні інтерфейси. Основний акцент робиться на фізично обґрунтованій взаємодії, що базується на реальних рухах користувача, захопленні предметів, зміні їх положення та використанні властивостей середовища для досягнення ігрових цілей.

Концепція інтерактивного середовища

Для дослідження процесу моделювання інтерактивного VR-середовища було розроблено концепцію гри **Little Strongman**, яка дозволяє зосередити увагу саме на взаємодії користувача з простором. Ігрове середовище представлено у вигляді замкненого будинку, що складається з послідовно з'єднаних кімнат, кожна з яких є окремою інтерактивною сценою з власним набором об'єктів та умов взаємодії.

Ключовою особливістю середовища є зміна масштабу персонажа відносно навколишніх об'єктів. Гравець керує персонажем значно меншого розміру, ніж стандартні елементи інтер'єру, що дозволяє:

- підкреслити просторову структуру сцени;
- змінити сприйняття знайомих об'єктів;
- створити умови, за яких взаємодія з середовищем стає основним механізмом ігрового процесу.

Такий підхід дає змогу дослідити, як **масштаб, пропорції та фізичні властивості об'єктів** впливають на зручність, інтуїтивність і ефективність взаємодії у VR.

Моделювання фізичної взаємодії з об'єктами

Інтерактивне середовище гри побудоване на принципі **фізичної взаємодії користувача з об'єктами віртуального простору**. Усі ключові елементи середовища — предмети інтер'єру, кнопки, двері — мають фізичні властивості, що підпорядковуються законам гравітації, зіткнень та інерції.

Користувач, використовуючи VR-контролери Meta Quest 2, може:

- захоплювати об'єкти різної маси та розміру;
- переміщувати їх у просторі;
- складати предмети один на один;
- використовувати об'єкти як опору або інструмент для досягнення інтерактивних елементів сцени.

Таким чином, логіка проходження гри формується безпосередньо через середовище, а не через текстові підказки або традиційні ігрові інтерфейси. Це дозволяє на практиці реалізувати принцип «environment-driven gameplay», де саме середовище виступає основним носієм ігрових механік.

Інтерактивні елементи як частина середовища

Особливу роль у моделюванні інтерактивного середовища відіграють **фізичні кнопки та двері**, які не є абстрактними тригерами, а інтегровані у простір як повноцінні об'єкти. Активація таких елементів можлива лише через

фізичний вплив — натискання, досягнення або використання предметів середовища.

Це дозволяє:

- дослідити поведінку користувача у VR-просторі;
- оцінити інтуїтивність розташування інтерактивних об'єктів;
- перевірити ефективність фізичних механік у контексті

автономного VR-шолому.

Обмеження та умови реалізації середовища

Проектування інтерактивного середовища здійснюється з урахуванням **апаратних обмежень Meta Quest 2**, зокрема автономного режиму роботи, обмеженої обчислювальної потужності та необхідності підтримки стабільної частоти кадрів. Віртуальний простір адаптований до обмеженого фізичного простору користувача, що дозволяє уникнути дискомфорту та зменшити ризик виникнення VR-sickness.

Окрема увага приділяється:

- стабільності трекінгу рухів рук;
- точності захоплення об'єктів;
- оптимізації фізичних розрахунків у реальному часі.

2.2. Структурна модель інтерактивного середовища VR-гри

Інтерактивне середовище у VR-грі є складною системою взаємопов'язаних компонентів, які забезпечують активну участь користувача у віртуальному просторі, фізичну взаємодію з об'єктами та отримання зворотного зв'язку в режимі реального часу. На відміну від традиційних ігрових середовищ, у VR користувач не лише спостерігає за подіями, а виступає безпосереднім елементом системи, впливаючи на стан віртуального світу через власні рухи та дії.

У межах даної магістерської роботи інтерактивне середовище VR-гри розглядається як **структурна модель**, що складається з низки логічних підсистем, об'єднаних у єдиний цикл взаємодії «користувач – система –

середовище – зворотний зв'язок».

Основні компоненти інтерактивного середовища:

1) Користувач

Користувач є центральним елементом інтерактивного середовища. Його фізичні рухи, жести рук та дії з контролерами формують вхідні дані для VR-системи. На цьому рівні відбувається прийняття рішень, просторове мислення та ініціація взаємодії з об'єктами віртуального світу.

2) VR-шолом Meta Quest 2

VR-шолом виконує роль апаратного інтерфейсу між користувачем та віртуальним середовищем. Він забезпечує:

- відображення тривимірного зображення;
- відстеження положення голови;
- передачу сенсорних даних у програмну частину гри.

Meta Quest 2 дозволяє реалізувати автономну модель інтерактивного середовища без підключення до зовнішнього комп'ютера.

3) VR-контролери

Контролери відповідають за відстеження рухів рук та ініціацію дій користувача, таких як захоплення, переміщення, натискання кнопок і взаємодія з об'єктами. Дані з контролерів передаються до системи взаємодії у реальному часі.

4) Система керування та обробки подій

Цей компонент відповідає за інтерпретацію дій користувача та перетворення їх у логічні події. Система подій визначає, яка дія має бути виконана у відповідь на рух або натискання, та передає відповідні команди іншим підсистемам.

5) Ігрова сцена

Ігрова сцена є просторовою основою інтерактивного середовища. Вона включає геометрію приміщень, зони взаємодії, точки навігації та розміщення інтерактивних об'єктів. Сцена не є статичною — вона реагує на дії користувача та змінює свій стан у процесі гри.

6) Інтерактивні об'єкти

До інтерактивних об'єктів належать предмети, з якими користувач може фізично взаємодіяти: кнопки, двері, рухомі об'єкти, елементи середовища. Кожен об'єкт має набір фізичних та логічних властивостей, що визначають його поведінку у VR-просторі.

7) Фізичний рушій

Фізичний рушій забезпечує коректну симуляцію фізичних законів: гравітації, зіткнень, інерції та маси об'єктів. Саме він відповідає за правдоподібність взаємодії користувача з елементами середовища.

8) Система зворотного зв'язку

Зворотний зв'язок реалізується через:

- візуальні зміни сцени;
- звукові ефекти;
- тактильні сигнали контролерів.

Він дозволяє користувачу миттєво відчутти результат своїх дій, що підсилює ефект присутності.

Цикл взаємодії в інтерактивному середовищі

Взаємодія у VR-середовищі має циклічний характер:

1. Користувач виконує дію (рух, жест, натискання).
2. VR-шолом і контролери фіксують цю дію.
3. Система керування обробляє подію.
4. Фізичний рушій та логіка гри змінюють стан об'єктів і сцени.
5. Користувач отримує зворотний зв'язок і коригує свої дії.

Для наочного представлення цієї структури була побудована узагальнена структурна модель інтерактивного VR-середовища (рис 2.2).

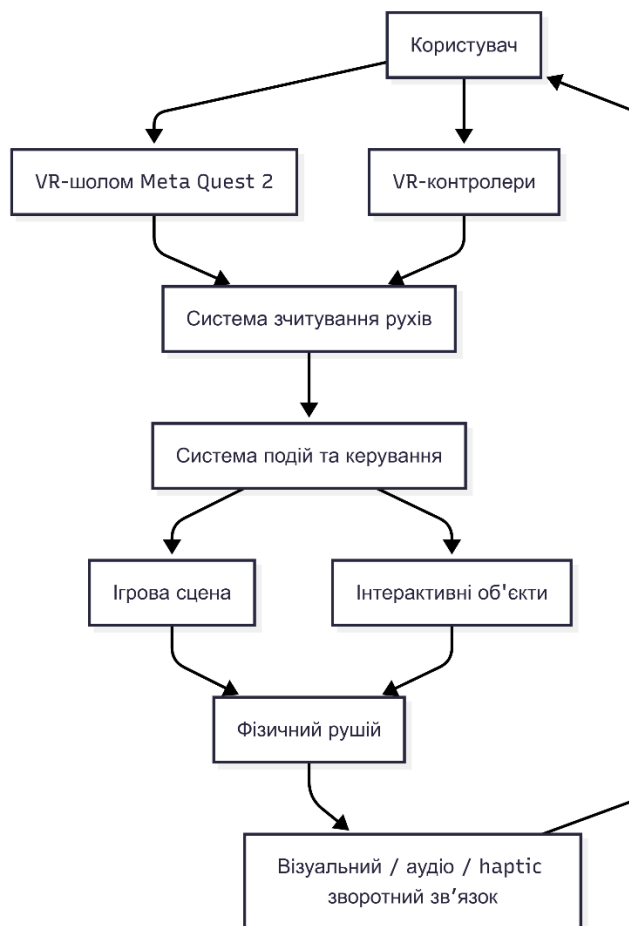


Рис. 2.2. Структурна модель інтерактивного середовища VR-гри

2.3. Моделювання взаємодії користувача з об'єктами середовища

На відміну від традиційних ігрових систем, у віртуальній реальності користувач взаємодіє з об'єктами безпосередньо через власні фізичні рухи, що потребує точного відображення цих дій у цифровому середовищі.

У межах даної роботи взаємодія користувача з об'єктами розглядається не як окрема механіка, а як процес перетворення фізичних дій у логічні та фізичні зміни стану віртуального середовища.

Інтерактивна взаємодія у VR-середовищі реалізується у вигляді послідовного циклу (Рис 2.3), який включає такі етапи:

1. Фізична дія користувача

Користувач виконує реальну дію: рухає рукою, нахиляється, стискає контролер або торкається віртуального об'єкта.

2. Зчитування та трекінг руху

VR-шолом Meta Quest 2 та контролери фіксують положення голови і рук у тривимірному просторі за допомогою сенсорів та камер inside-out tracking.

3. Інтерпретація дії системою керування

Отримані дані передаються до системи керування, де визначається тип взаємодії: захоплення, переміщення, натискання, активація або зіткнення.

4. Фізична обробка взаємодії

Фізичний рушій обчислює наслідки дії з урахуванням маси об'єкта, гравітації, колізій та обмежень руху.

5. Зміна стану об'єкта або середовища

Об'єкт змінює свій стан (переміщується, активується, відкриває доступ до іншої зони), що впливає на загальний стан ігрової сцени.

6. Зворотний зв'язок для користувача

Користувач отримує візуальний, аудіальний або тактильний відгук, який підтверджує успішність або неможливість виконання дії.

Циклічність цього процесу дозволяє користувачу постійно коригувати свої дії, формуючи безперервну інтерактивну взаємодію з середовищем.

Однією з базових форм фізичної взаємодії у VR-середовищі є захоплення та переміщення об'єктів. Для цього об'єкти середовища наділяються такими властивостями:

- фізичним колайдером для визначення зіткнень;
- параметрами маси та гравітації;
- логікою обмеження доступності (об'єкт можна або не можна захопити).

Після активації захоплення об'єкт тимчасово підпорядковується рухам контролера, але залишається частиною фізичної системи, що запобігає порушенню просторової логіки сцени.

Іншим типом взаємодії є натискання фізичних елементів, таких як кнопки або важелі. У цьому випадку взаємодія моделюється не через логічну подію, а через фізичне зміщення об'єкта у просторі.

Активация відбувається лише тоді, коли об'єкт досягає заданого порогу переміщення, що імітує реальну механічну дію. Такий підхід підвищує реалістичність взаємодії та змушує користувача виконувати точні рухи.

Важливою складовою моделювання взаємодії є введення обмежень, які запобігають нелогічним або нереалістичним діям користувача. До таких обмежень належать:

- недосяжність об'єктів без використання допоміжних предметів;
- залежність результату взаємодії від положення об'єкта у просторі;
- вплив гравітації та маси на можливість переміщення.

Завдяки цьому інтерактивне середовище перестає бути набором сценарних тригерів і набуває властивостей цілісної фізично обґрунтованої системи.

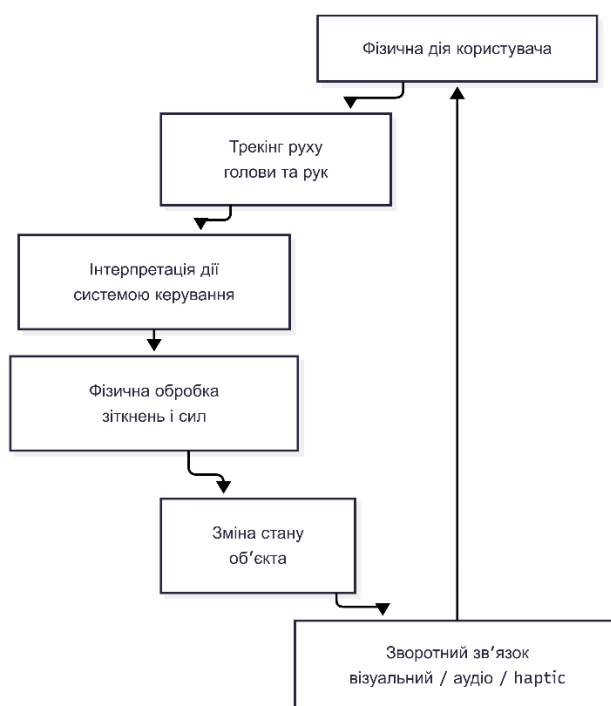


Рис 2.3. Схема взаємодії користувача з об'єктом у VR-середовищі

2.4. Проектування ігрових сцен як інтерактивних просторів

У VR-іграх ігрова сцена перестає виконувати роль пасивної декорації та перетворюється на **активне інтерактивне середовище**, з яким користувач постійно взаємодіє. У межах даної магістерської роботи ігрова сцена

розглядається як **цілісна система об'єктів, просторових зон та правил взаємодії**, що формують користувацький досвід у віртуальній реальності.

Проектування інтерактивного середовища передбачає не лише візуальне оформлення сцени, а й **логічну організацію простору**, визначення доступності об'єктів та сценаріїв їх використання користувачем.

Ігрові сцени VR-гри Little Strongman побудовані за принципом **просторового зонування**, де кожне приміщення виконує окрему функціональну роль у загальній структурі інтерактивного середовища. Кімнати будинку не є ізольованими декораціями, а пов'язані між собою через систему умовного доступу, що активується діями користувача.

Кожна зона сцени містить:

- інтерактивні об'єкти (кнопки, рухомі предмети, опори);
- статичні обмеження (стіни, двері, стелі);
- логіку переходу до наступної зони.

Такий підхід дозволяє формувати **послідовну модель дослідження простору**, де користувач поступово освоює середовище шляхом фізичної взаємодії з його елементами.

Розміщення інтерактивних об'єктів у сцені здійснюється з урахуванням ергономіки VR-середовища та фізичних можливостей користувача. Основними принципами є:

- розташування ключових об'єктів у зоні природного поля зору користувача;
- уникнення надмірної щільності інтерактивних елементів;
- створення необхідності використання допоміжних предметів для досягнення мети.

Зокрема, кнопки активації дверей навмисно розміщуються поза зоною прямої досяжності, що змушує користувача аналізувати простір та використовувати предмети середовища як інструменти взаємодії. Таким чином, інтерактивність закладена не в окремий сценарій, а в просторову структуру сцени.

Особливістю проектування сцен у VR-грі *Little Strongman* є зменшений масштаб ігрового персонажа, що безпосередньо впливає на сприйняття простору. Об'єкти, які у реальному житті є незначними, у VR-середовищі перетворюються на великогабаритні елементи, що формують складні просторові задачі.

Навігація користувача у сцені базується на:

- фізичному переміщенні у межах доступної зони;
- використанні об'єктів як сходинок або опор;
- візуальних орієнтирах у просторі сцени.

Такий підхід дозволяє мінімізувати використання штучних способів пересування та зменшити ризик виникнення дискомфорту, пов'язаного з VR sickness.

У контексті даної роботи ігрова сцена розглядається як модель інтерактивного середовища, у якій:

- користувач є активним елементом системи;
- об'єкти реагують на фізичні дії;
- простір визначає можливі сценарії взаємодії.

Для наочного представлення структури інтерактивного середовища у VR-грі *Little Strongman* було побудовано схематичну модель ігрової сцени (рисунок 2.4). Дана схема відображає основні компоненти інтерактивного середовища та логіку їх взаємодії у процесі ігрового сценарію.

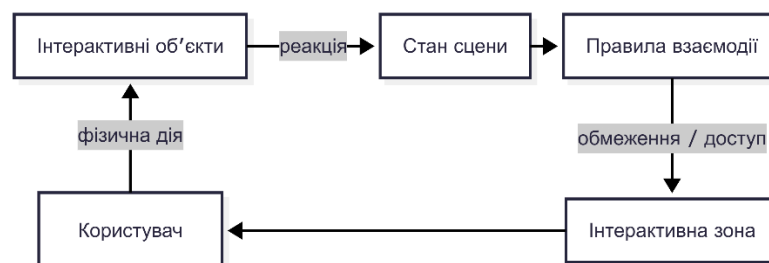


Рис 2.4. Схематичне представлення ігрової сцени як інтерактивного середовища

На рисунку 2.4 користувач розглядається як активний елемент системи, який здійснює фізичні дії у віртуальному просторі. Ці дії спрямовані на інтерактивні об'єкти сцени, що, у свою чергу, викликають зміну стану ігрового середовища. Зміна стану сцени підпорядковується заданим правилам взаємодії, які визначають доступність зон, можливість переходу між приміщеннями та подальший розвиток ігрового процесу.

Таким чином, схема ілюструє циклічний характер взаємодії «користувач — об'єкт — середовище», де кожна дія користувача призводить до реакції середовища, а оновлений стан сцени впливає на подальші можливості взаємодії. Саме така модель лежить в основі проєктування інтерактивних VR-сцен у даній роботі.

2.5. Візуалізація моделі інтерактивного середовища

Для підтвердження практичної реалізації розробленої моделі інтерактивного середовища у VR-грі *Little Strongman* було виконано її візуалізацію безпосередньо у середовищі розробки Unity. Метою даного підрозділу є демонстрація відповідності між концептуальною моделлю взаємодії користувача з віртуальним середовищем та її фактичним втіленням у вигляді ігрової сцени.

Розроблена модель інтерактивного середовища ґрунтується на взаємодії трьох ключових компонентів: користувача, інтерактивних об'єктів та правил функціонування сцени. У процесі реалізації кожен з цих компонентів було представлено конкретними елементами ігрової сцени Unity, що забезпечує відтворення повного циклу взаємодії у VR-просторі.

На рисунку 2.5 наведено приклад реалізації моделі інтерактивного середовища у середовищі Unity. Сцена містить інтерактивні об'єкти, доступні для фізичної взаємодії, а також елементи, що визначають логіку переходу між станами середовища.

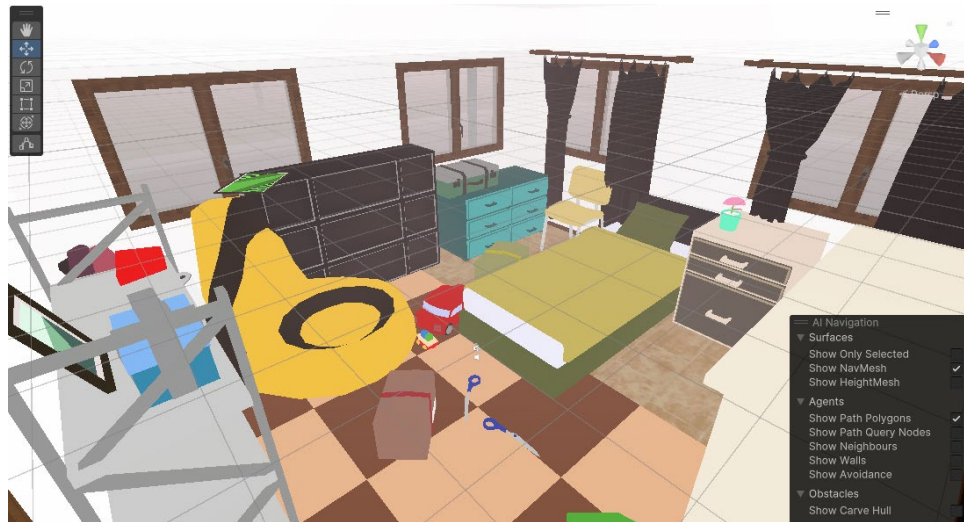


Рис. 2.5. Реалізація моделі інтерактивного середовища у Unity

Користувач у даній моделі представлений VR-камерою та контролерами Meta Quest 2, які забезпечують відстеження положення голови та рук у тривимірному просторі. Це дозволяє користувачу безпосередньо взаємодіяти з об'єктами сцени шляхом захоплення, переміщення та маніпуляції предметами, що відповідає концепції фізичної взаємодії, описаній у попередніх підрозділах.

Інтерактивні об'єкти сцени реалізовані у вигляді тривимірних моделей з прикріпленими компонентами колайдерів та фізичних властивостей. Кожен такий об'єкт реагує на дії користувача відповідно до заданих правил взаємодії: може бути піднятий, переміщений або використаний для досягнення функціональних елементів сцени, зокрема фізичних кнопок активації.

Правила функціонування середовища реалізовані через логіку ігрових скриптів, які визначають зміну стану сцени залежно від дій користувача. Наприклад, активація кнопки призводить до відкривання дверей і переходу сцени у новий стан, що наочно демонструє причинно-наслідковий зв'язок між фізичною дією та реакцією середовища.

Таким чином, кожен елемент концептуальної моделі має пряме відображення у реалізації сцени:

- користувач - VR-камера та контролери Meta Quest 2;

- інтерактивні об'єкти - фізичні 3D-моделі з колайдерами;
- правила взаємодії - програмна логіка сцени;
- зміна стану середовища - відкриття доступу до нових зон.

Візуалізація моделі інтерактивного середовища у Unity підтверджує коректність обраного підходу до проєктування VR-гри та демонструє, що інтерактивне середовище є активним елементом ігрового процесу. Середовище не лише відображає дії користувача, а й динамічно змінюється у відповідь на них, забезпечуючи високий рівень занурення та логічну послідовність ігрового сценарію.

Висновки до розділу 2

У другому розділі магістерської роботи було виконано практичне дослідження процесу моделювання інтерактивного середовища у VR-іграх на прикладі розробки VR-гри *Little Strongman* з використанням VR-шолому Meta Quest 2. Основну увагу зосереджено не на створенні ігрового продукту як такого, а на формуванні та реалізації моделі взаємодії користувача з віртуальним середовищем.

У процесі роботи сформульовано постановку задачі та визначено концепцію VR-гри, в якій інтерактивне середовище виступає центральним елементом ігрового процесу. Було обґрунтовано вибір жанру escape room як такого, що найкраще підходить для дослідження фізичної взаємодії, просторового мислення та маніпуляції об'єктами у віртуальному просторі. Застосування зміненого масштабу персонажа дозволило створити нестандартні умови взаємодії та підкреслити значущість фізичних дій користувача у VR-середовищі.

У розділі детально розглянуто процес проектування інтерактивного середовища, який включає визначення ролей користувача, інтерактивних об'єктів та правил функціонування сцени. Побудовано логічну модель середовища, що описує причинно-наслідкові зв'язки між діями користувача та реакціями віртуального простору, зокрема зміну станів середовища внаслідок фізичної взаємодії з об'єктами.

Особливу увагу приділено реалізації фізичної взаємодії у VR, що охоплює механіки захоплення, переміщення та використання об'єктів як елементів ігрової логіки. Продемонстровано, що інтерактивність у VR-іграх є результатом узгодженої роботи апаратних засобів (VR-шолом і контролери), програмної логіки та фізичних моделей об'єктів. Такий підхід забезпечує формування сенсомоторного контуру взаємодії, який є основою ефекту присутності у віртуальному середовищі.

Також було здійснено візуалізацію розробленої моделі інтерактивного середовища у середовищі Unity, що підтверджує відповідність між

концептуальною схемою та її практичним втіленням у вигляді ігрової сцени. Представлена реалізація наочно демонструє, як абстрактна модель взаємодії трансформується у конкретні елементи сцени, правила поведінки об'єктів і сценарії взаємодії користувача з віртуальним середовищем.

У результаті виконання другого розділу доведено, що процес моделювання інтерактивного середовища у VR-іграх є багаторівневим і включає концептуальне проєктування, формалізацію взаємодії та практичну реалізацію у VR-рушії. Отримані результати створюють методологічну основу для подальшої реалізації та аналізу ефективності інтерактивного середовища, що буде розглянуто у наступному розділі магістерської роботи.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРАКТИВНИХ МЕХАНІК VR-ГРИ

У третьому розділі магістерської роботи розглядається практична реалізація інтерактивних механік VR-гри та результати їх тестування на цільовій платформі Meta Quest 2. Даний розділ є логічним продовженням етапу проектування та спрямований на опис програмної реалізації основних елементів взаємодії користувача з ігровим середовищем у імерсивній віртуальній реальності.

Основна увага зосереджена на реалізації керування користувачем у VR-середовищі, організації фізичної взаємодії з об'єктами та впровадженні базових інтерактивних механік, які безпосередньо формують ігровий процес. Реалізація зазначених механік здійснювалася з використанням рушія Unity та стандартних засобів підтримки VR, адаптованих до особливостей автономного VR-шолома Meta Quest 2.

У межах розділу детально описано роботу з VR-контролерами, механізми захоплення та переміщення об'єктів, а також реалізацію фізичних кнопок і логіки відкривання дверей як ключових елементів проходження гри. Обрані рішення спрямовані на забезпечення інтуїтивної взаємодії користувача з ігровим світом та підтримання стабільної роботи фізичної симуляції в умовах обмежених обчислювальних ресурсів автономної VR-платформи.

Окрему частину розділу присвячено тестуванню VR-гри безпосередньо на пристрої Meta Quest 2. Проведено перевірку працездатності основних механік, оцінено стабільність роботи додатку та проаналізовано продуктивність гри в умовах реального використання. Отримані результати дозволяють зробити висновки щодо доцільності застосованих технічних рішень та їх відповідності вимогам до VR-ігор з інтерактивним середовищем.

Таким чином, у третьому розділі здійснено практичне підтвердження можливості реалізації інтерактивних механік VR-гри з використанням

стандартних інструментів рушія Unity та апаратних можливостей VR-шолома Meta Quest 2.

3.1. Реалізація керування користувачем у VR-середовищі

Керування користувачем у VR-середовищі реалізовано з використанням пакету **XR Interaction Toolkit**, який забезпечує стандартний набір компонентів для роботи з VR-пристроями у рушії Unity. Даний підхід дозволяє поєднати готові інструменти для відстеження рухів користувача з власними скриптами, що розширюють базовий функціонал керування відповідно до ігрової концепції.

Основою керування є об'єкт **XR Origin**, який відповідає за просторову прив'язку гравця до сцени. До його складу входять камера, що представляє VR-шолом, та об'єкти рук, які синхронізуються з VR-контролерами (рис. 3.1).

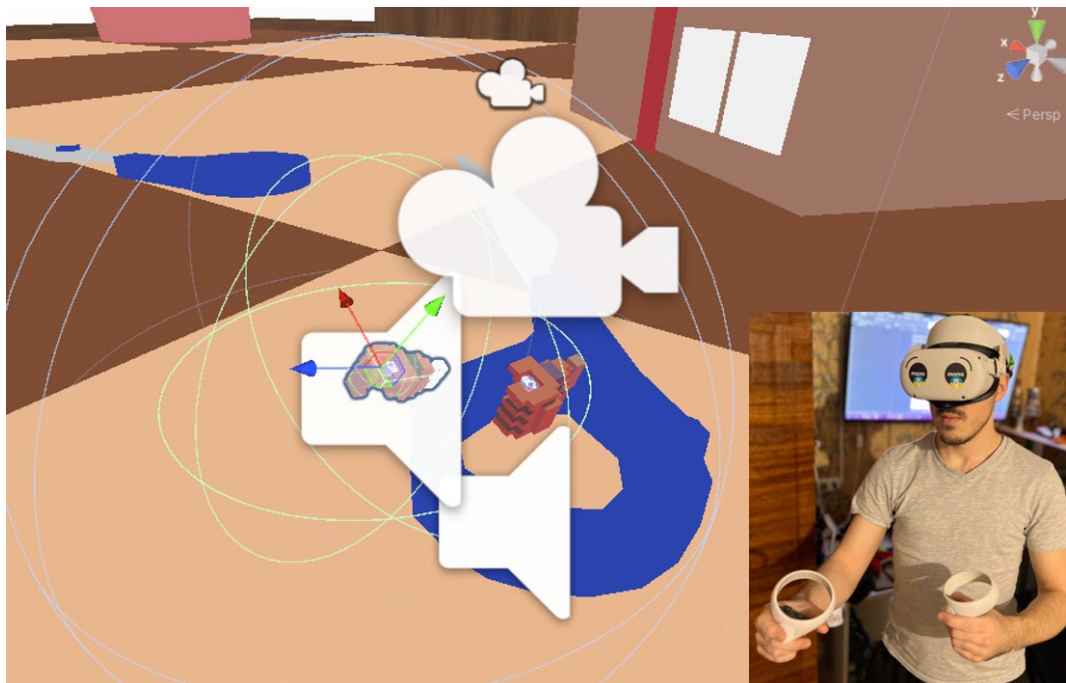


Рис. 3.1. Представлення аватара гравця у VR-середовищі

Для реалізації переміщення та обмеження колізій використовується компонент **Character Controller** (рис 3.2.), що дозволяє керувати рухом гравця без прямого використання фізичних сил.

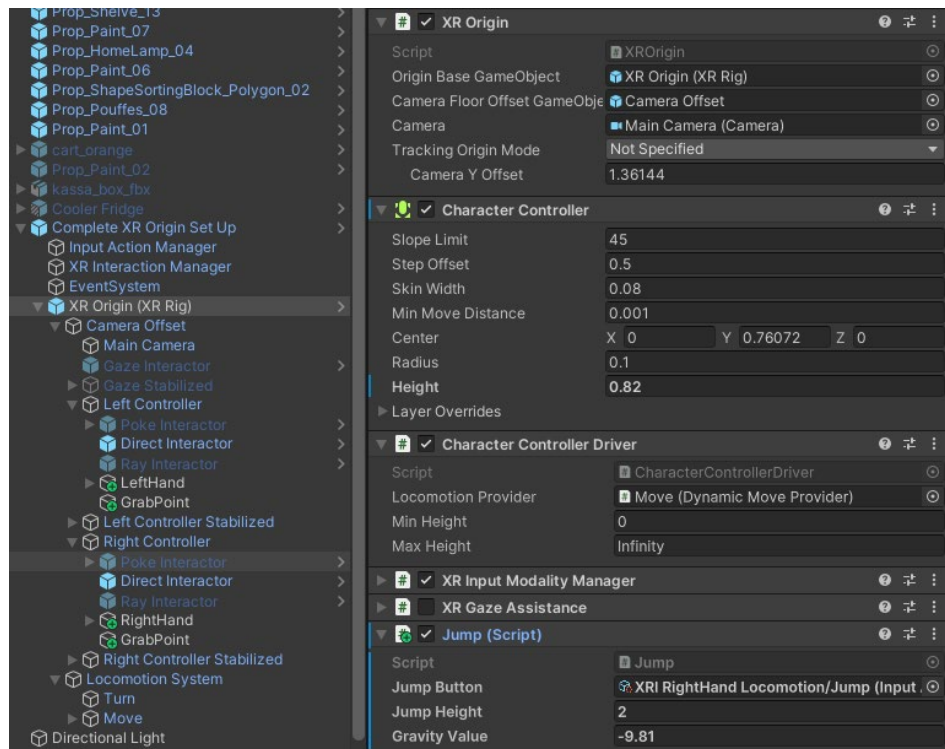


Рис 3.2. Ієрархія об'єкта XR Origin у середовищі Unity

Для навігації у віртуальному просторі використовується компонент **Dynamic Move Provider**, який входить до складу XR Interaction Toolkit та працює у зв'язці з **Locomotion System**. Він забезпечує плавне переміщення гравця у напрямку, заданому джойстиком VR-контролера, з урахуванням орієнтації камери.

Такий спосіб переміщення дозволяє:

- адаптувати гру до обмеженого фізичного простору користувача;
- зменшити різкі зміни положення камери;
- забезпечити стабільну роботу на автономному пристрої Meta Quest

2.

1) Реалізація механіки стрибка

Оскільки стандартні компоненти XR Interaction Toolkit не передбачають механіку стрибка, у проєкті реалізовано власний скрипт **Jump**, який розширює можливості переміщення гравця. Скрипт використовує систему введення **Input System Unity** та компонент **Character Controller**.

Фрагмент програмного коду механіки стрибка наведено нижче:

```

using UnityEngine;
using UnityEngine.InputSystem;

public class Jump : MonoBehaviour
{
    [SerializeField] private InputActionReference jumpButton;
    [SerializeField] private float jumpHeight = 2.0f;
    [SerializeField] private float gravityValue = -9.81f;

    private CharacterController _characterController;
    private Vector3 _playerVelocity;

    private void Awake() => _characterController =
GetComponent<CharacterController>();

    private void OnEnable() => jumpButton.action.performed += Jumping;

    private void OnDisable() => jumpButton.action.performed -= Jumping;

    private void Jumping(InputAction.CallbackContext obj)
    {
        if (!_characterController.isGrounded) return;
        _playerVelocity.y += Mathf.Sqrt(jumpHeight * -3.0f * gravityValue);
    }

    private void Update()
    {
        if (_characterController.isGrounded && _playerVelocity.y < 0)
        {

```

```

        _playerVelocity.y = 0f;
    }

    _playerVelocity.y += gravityValue * Time.deltaTime;
    _characterController.Move(_playerVelocity * Time.deltaTime);
}
}

```

У представленому скрипті реалізовано базову логіку стрибка з урахуванням гравітації та стану контакту гравця з поверхнею.

- **InputActionReference jumpButton** — посилання на дію введення, прив'язану до кнопки VR-контролера Meta Quest 2.
- **jumpHeight** — параметр, що визначає максимальну висоту стрибка.
- **gravityValue** — значення гравітації, яке використовується для імітації падіння.
- **CharacterController** — компонент, що забезпечує переміщення гравця та обробку колізій.
- Метод **Jumping()** — виконує перевірку стану `isGrounded` і ініціює вертикальний імпульс.
- Метод **Update()** — відповідає за застосування гравітації та переміщення гравця у просторі.

Такий підхід дозволяє інтегрувати механіку стрибка у VR-середовище без використання повноцінної фізичної симуляції для персонажа, що позитивно впливає на продуктивність гри.

Адаптація керування до особливостей VR

Реалізована система керування поєднує стандартні компоненти XR Interaction Toolkit з власними програмними рішеннями, що дозволяє:

- забезпечити інтуїтивне керування;
- зберегти стабільність фізичної симуляції;

- адаптувати рух гравця до специфіки VR-середовища та ігрової концепції.

Таким чином, у підрозділі 3.1 реалізовано комплексну систему керування користувачем у VR-середовищі, яка базується на XR Interaction Toolkit та доповнюється авторськими скриптами. Представлений програмний код підтверджує практичну реалізацію механік переміщення та може бути використаний як приклад побудови системи керування у VR-іграх для платформи Meta Quest 2.

3.2. Реалізація фізичної взаємодії з об'єктами у VR-середовищі

У межах проєкту фізична взаємодія реалізована з використанням стандартної фізичної системи рушія Unity у поєднанні з авторськими скриптами, що забезпечують захоплення, утримання та переміщення об'єктів у VR-просторі.

На відміну від повністю готових механік XR Interaction Toolkit, у проєкті використано власну реалізацію захоплення предметів, що дозволило точніше контролювати поведінку об'єктів та адаптувати її до обмежень автономного VR-шолома Meta Quest 2.

1) Загальна схема фізичної взаємодії

Фізична взаємодія з об'єктами у грі базується на таких принципах:

- взаємодія ініціюється активною дією користувача;
- захоплення можливе лише у межах зони досяжності руки;
- під час утримання об'єкта фізична симуляція частково вимикається;
- після відпускання об'єкт повністю повертається під контроль фізичної системи.

Схема фізичної взаємодії показана на малюнку 3.3.



Рисунок 3.3. Схема фізичної взаємодії гравця з об'єктами у VR-середовищі

2) Реалізація механіки захоплення об'єктів

Для реалізації захоплення предметів використовується авторський скрипт VRGrab, який підключається до віртуальної руки гравця (Лістинг 3.1). Скрипт відповідає за визначення об'єктів у зоні взаємодії, обробку натискання кнопки захоплення та коректне позиціонування предмета відносно руки.

Фрагмент програмного коду механіки захоплення наведено нижче:

Лістинг 3.1. Механіка захоплення предметів

```

using UnityEngine;
using UnityEngine.XR;

public class VRGrab : MonoBehaviour
{
    public XRNode controllerNode;
    public string pickupTag = "Pickup";
    public Transform grabPoint;

    private InputDevice device;
    private GameObject objectInHand;
    private Collider objectInTrigger;

    private Vector3 localPosOffset;
    private Quaternion localRotOffset;
  
```

```

void Start()
{
    device = InputDevices.GetDeviceAtXRNode(controllerNode);
}

void Update()
{
    if (device.TryGetFeatureValue(CommonUsages.gripButton, out bool
gripPressed))
    {
        if (gripPressed && objectInHand == null && objectInTrigger != null)
        {
            GrabObject(objectInTrigger.gameObject);
        }
        else if (!gripPressed && objectInHand != null)
        {
            ReleaseObject();
        }
    }

    if (objectInHand != null)
    {
        UpdateHeldObjectPose();
    }
}

```

Опис ключових елементів механіки захоплення:

- **XRNode controllerNode** — визначає, до якого VR-контролера (лівого або правого) прив'язана взаємодія.

- **pickupTag** — тег, що використовується для маркування об'єктів, які можуть бути захоплені.
- **grabPoint** — опорна точка, відносно якої обчислюється положення предмета в руці.
- **InputDevice device** — об'єкт, що забезпечує доступ до стану VR-контролера.
- **objectInTrigger** — об'єкт, що перебуває у зоні досяжності руки.
- **objectInHand** — об'єкт, який у поточний момент утримується гравцем.

Захоплення активується при натисканні кнопки хвату (grip), за умови, що у зоні взаємодії присутній допустимий об'єкт.

3) Фіксація об'єкта у руці

Після захоплення об'єкта виконується вимкнення фізичної симуляції для запобігання неконтрольованим коливанням та конфліктам з фізикою сцени (Лістинг 3.2). Для цього Rigidbody об'єкта переводиться у кінематичний режим, а дія гравітації тимчасово вимикається:

Лістинг 3.2. Фіксація об'єктів у руці грація

```
private void GrabObject(GameObject obj)
{
    objectInHand = obj;
    Rigidbody rb = obj.GetComponent<Rigidbody>();

    if (rb != null)
    {
        rb.isKinematic = true;
        rb.useGravity = false;
    }
}
```

```
Transform reference = grabPoint != null ? grabPoint : transform;
```

```

        localPosOffset =
reference.InverseTransformPoint(obj.transform.position);
        localRotOffset = Quaternion.Inverse(reference.rotation) *
obj.transform.rotation;
    }

```

Використання локальних зсувів позиції та обертання дозволяє зберігати природне положення предмета у руці незалежно від його початкової орієнтації, як показано на рисунку 3.7.

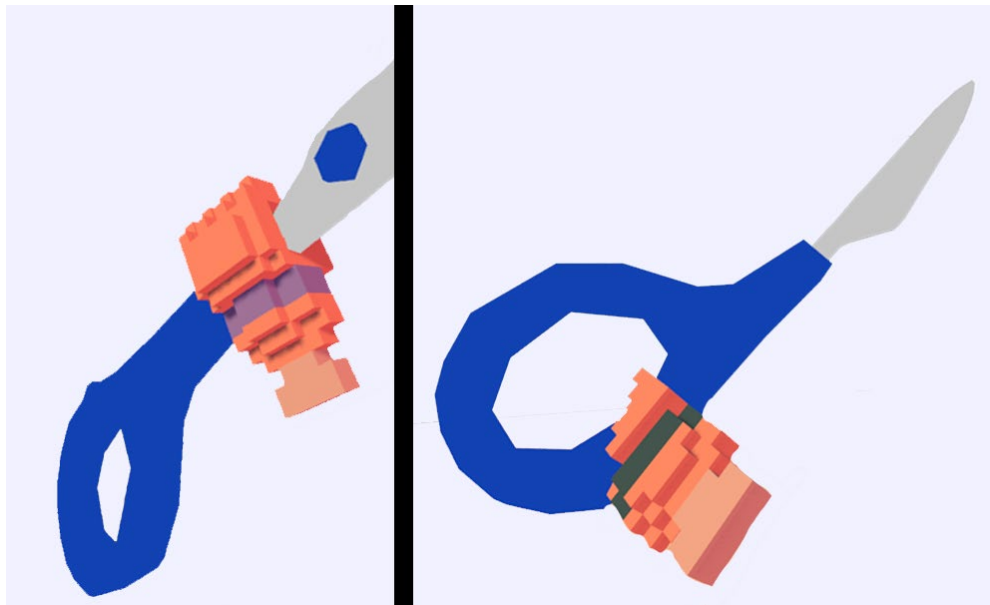


Рис. 3.7. Позиціонування об'єкта відносно точки захоплення у VR-руці.

4) Підтримка позиції утримуваного об'єкта

Під час утримання об'єкта його положення та орієнтація постійно оновлюються відповідно до рухів руки користувача (Лістинг 3.3). Це забезпечує візуальну стабільність та створює відчуття фізичного контролю над предметом.

Лістинг 3.3. Оновлення орієнтації та положення предметів у руці

```

private void UpdateHeldObjectPose()
{

```

```

Transform reference = grabPoint != null ? grabPoint : transform;

objectInHand.transform.position =
reference.TransformPoint(localPosOffset);
objectInHand.transform.rotation = reference.rotation * localRotOffset;
}

```

5) Відпускання об'єкта та повернення до фізичної симуляції

Після відпускання кнопки захоплення фізична симуляція для об'єкта повністю відновлюється (Лістинг 3.4). Це дозволяє предмету взаємодіяти з оточенням згідно з законами фізики, зокрема падати під дією гравітації або стикатися з іншими об'єктами сцени. Це реалізовано у наступному фрагменті коду:

Лістинг 3.4. Метод відпускання об'єкта

```

private void ReleaseObject()
{
    Rigidbody rb = objectInHand.GetComponent<Rigidbody>();

    if (rb != null)
    {
        rb.isKinematic = false;
        rb.useGravity = true;
    }

    objectInHand = null;
}

```

6) Виявлення об'єктів у зоні взаємодії

Для визначення можливості захоплення використовується тригерний колайдер, закріплений на віртуальній руці. Вхід та вихід об'єкта з цієї зони обробляються відповідними подіями (Лістинг 3.5):

Лістинг 3.5. Метод виявлення об'єктів у зоні взаємодії контролерів

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag(pickupTag))
        objectInTrigger = other;
}

private void OnTriggerExit(Collider other)
{
    if (objectInTrigger == other)
        objectInTrigger = null;
}
```

Цей підхід дозволяє мінімізувати кількість перевірок у сцені та забезпечує стабільну роботу механіки захоплення.

Таким чином було реалізовано авторську систему фізичної взаємодії з об'єктами у VR-середовищі, яка поєднує стандартну фізику Unity з подієвим підходом до захоплення предметів. Реалізована механіка забезпечує стабільну, передбачувану та інтуїтивну взаємодію користувача з ігровим середовищем і створює основу для реалізації інтерактивних механік, зокрема фізичних кнопок та логіки відкривання дверей, що розглядаються у наступному підрозділі.

3.3. Реалізація інтерактивних механік

У межах даного проєкту інтерактивність реалізована через фізичні елементи середовища, з якими гравець може взаємодіяти без використання абстрактних інтерфейсів. Основний акцент зроблено на механіці фізичних

кнопок та логіці відкривання дверей, що безпосередньо пов'язано з ігровою метою — послідовним відкриванням доступу до нових приміщень.

На відміну від класичних ігрових кнопок, реалізованих через натискання віртуальних UI-елементів, у даному проєкті застосовано підхід фізичної взаємодії, при якому кнопка реагує на реальне переміщення об'єкта в просторі під дією руки гравця або інших предметів.

1) Загальна логіка інтерактивної взаємодії

Реалізація інтерактивних механік у грі базується на подієвій архітектурі та складається з таких основних етапів:

1. фізичний контакт гравця з інтерактивним елементом;
2. зміна положення або стану об'єкта;
3. фіксація досягнення порогового значення взаємодії;
4. виклик події, що ініціює зміну стану іншого об'єкта;
5. виконання анімації або логіки реакції середовища.



Рисунок 3.8 Загальна схема реалізації інтерактивної механіки «кнопка — двері»

2) Реалізація фізичної кнопки

Фізична кнопка реалізована за допомогою авторського скрипта `PhysicalButton` (Лістинг 3.6), який аналізує локальне зміщення об'єкта кнопки вздовж однієї осі. Такий підхід дозволяє досягти відчуття реального натискання без використання складних фізичних з'єднань або додаткових обмежень.

Лістинг 3.6. Фрагмент коду реалізації роботи фізичних кнопок

```

using UnityEngine;
using UnityEngine.Events;

```

```

public class PhysicalButton : MonoBehaviour
{
    public UnityEvent OnButtonPressed;
    public float pressThreshold = 0.015f;

    private Vector3 startPos;
    private bool isPressed = false;

    void Start()
    {
        startPos = transform.localPosition;
    }

    void Update()
    {
        float displacement = startPos.z - transform.localPosition.z;

        if (!isPressed && displacement > pressThreshold)
        {
            isPressed = true;
            OnButtonPressed?.Invoke();
        }

        if (isPressed && displacement < pressThreshold * 0.5f)
        {
            isPressed = false;
        }
    }
}

```

Основні елементи реалізації:

- **UnityEvent OnButtonPressed** — подієвий механізм, який дозволяє призначати реакцію на натискання кнопки без жорсткого зв'язування зі скриптами.
- **pressThreshold** — порогове значення переміщення, яке визначає момент активації кнопки.
- **startPos** — початкове локальне положення кнопки, відносно якого обчислюється зміщення.
- **isPressed** — прапорець стану, що запобігає багаторазовому спрацюванню події.

Використання гістерезису (зменшене порогове значення при відпусканні) дозволяє уникнути багаторазового спрацювання події внаслідок незначних коливань кнопки. Визначення моменту натискання фізичної кнопки за локальним зміщенням описано у схемі 3.9.

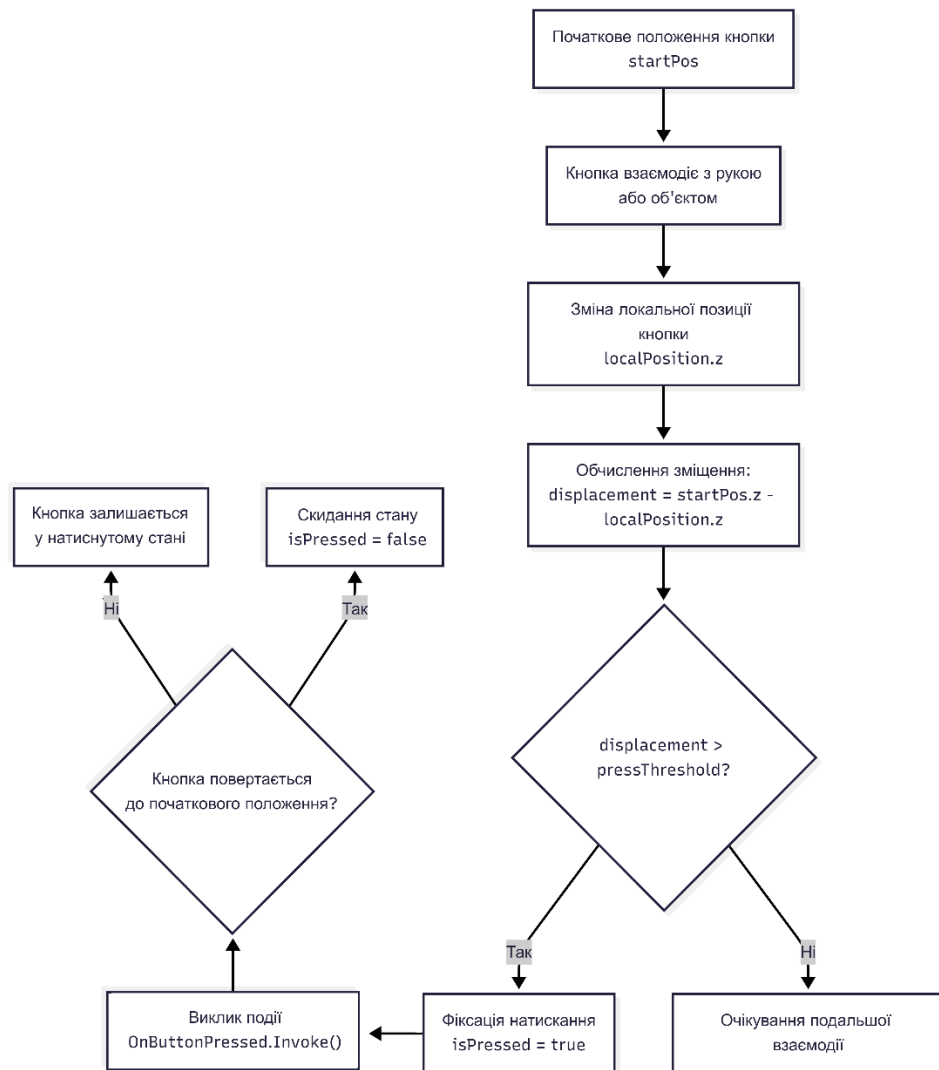


Рис. 3.9. Схема визначення моменту натискання фізичної кнопки за локальним зміщенням

3) Реалізація логіки відкривання дверей

Для реалізації реакції середовища на натискання кнопки використано скрипт **DoorController** (Лістинг 3.7), який відповідає за плавне відкривання дверей шляхом інтерполяції обертання.

Лістинг 3.7. Фрагмент коду реалізації роботи дверей
using UnityEngine;

```

public class DoorController : MonoBehaviour
{

```



```

public float openAngle = -40f;
public float speed = 2f;

private Quaternion closedRot;
private Quaternion openRot;
private bool isOpening = false;

void Start()
{
    closedRot = transform.rotation;
    openRot = Quaternion.Euler(transform.eulerAngles + new Vector3(0,
openAngle, 0));
}

public void OpenDoor()
{
    Debug.Log("BUTTON PRESSED → OpenDoor()");
    isOpening = true;
}

void Update()
{
    if (isOpening)
    {
        transform.rotation = Quaternion.Lerp(
            transform.rotation,
            openRot,
            Time.deltaTime * speed
        );
    }
}

```

```

    }
}

```

Основні компоненти реалізації:

- **closedRot / openRot** — кватерніони, що визначають закрите та відкрите положення дверей.
- **openAngle** — кут повороту дверей навколо вертикальної осі.
- **Quaternion.Lerp** — забезпечує плавний перехід між станами без різких рухів.
- **isOpening** — логічний прапорець, що активує процес відкривання.

Такий підхід дозволяє уникнути використання складної фізичної симуляції для дверей, що є важливим з огляду на обмежені обчислювальні ресурси автономного VR-пристрою Meta Quest 2.

4) Взаємозв'язок кнопки та дверей

Зв'язок між кнопкою та дверима реалізовано через **UnityEvent** (рис 3.11), що дозволяє налаштовувати логіку взаємодії без змін програмного коду. Подія `OnButtonPressed` кнопки прив'язується до методу `OpenDoor()` відповідного об'єкта дверей безпосередньо в інспекторі Unity.

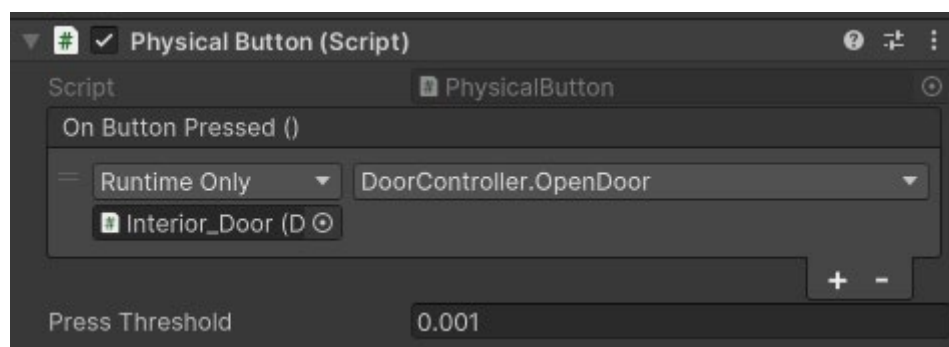


Рис. 3.11. Налаштування UnityEvent для зв'язку кнопки та дверей у середовищі Unity

Таким чином нами було реалізовано інтерактивні механіки, що базуються на фізичній взаємодії користувача з ігровим середовищем. Використання фізичних кнопок у поєднанні з подієвим керуванням дозволяє створювати логічні зв'язки між об'єктами сцени, забезпечуючи послідовний та інтуїтивно зрозумілий ігровий процес. Реалізований підхід є масштабованим і дозволяє легко розширювати кількість інтерактивних елементів у наступних рівнях гри.

3.4. Тестування VR-гри на платформі Meta Quest 2

Тестування VR-гри є обов'язковим етапом розробки, оскільки дозволяє перевірити коректність реалізації інтерактивних механік, стабільність роботи програмного забезпечення та відповідність гри апаратним обмеженням цільової платформи. У контексті VR-додатків тестування набуває особливого значення через поєднання високих вимог до продуктивності, чутливості до затримок та необхідності забезпечення комфортного користувацького досвіду.

У межах даної магістерської роботи тестування VR-гри **Little Strongman** проводилося безпосередньо на автономному VR-шоломі **Meta Quest 2**, який є цільовою платформою розробки. Такий підхід дозволив оцінити роботу гри в реальних умовах експлуатації, з урахуванням обмежень мобільного апаратного забезпечення, особливостей відстеження рухів користувача та специфіки взаємодії з VR-контролерами.

Процес тестування охоплював кілька ключових аспектів:

- перевірку працездатності основних ігрових механік (переміщення, стрибки, захоплення та маніпуляція об'єктами);
- аналіз коректності фізичної взаємодії з ігровими об'єктами та елементами середовища;
- оцінку стабільності роботи гри під час тривалих ігрових сесій;
- аналіз продуктивності VR-додатку, зокрема частоти кадрів, плавності відображення сцени та навантаження на апаратні ресурси.

Особлива увага приділялася тестуванню інтерактивних механік, пов'язаних із фізичними кнопками та логікою відкривання дверей, оскільки саме ці елементи визначають прогрес гравця та завершення ігрових рівнів. Також враховувалися особливості масштабування світу та сприйняття фізики об'єктів користувачем у VR-середовищі.

Для тестування використовувалися стандартні засоби розробника Unity, вбудовані інструменти профілювання, а також безпосереднє спостереження за поведінкою гри під час запуску на Meta Quest 2. Результати тестування фіксувалися у вигляді описів сценаріїв, таблиць результатів та скріншотів із середовища Unity і VR-пристрою, що дозволило систематизувати виявлені особливості та зробити обґрунтовані висновки.

Отримані результати тестування стали підґрунтям для оцінки готовності VR-гри до демонстрації на захисті магістерської роботи та визначення напрямків подальшого вдосконалення проєкту.

3.4.1. Перевірка працездатності та стабільності гри

Перевірка працездатності та стабільності VR-гри **Little Strongman** була спрямована на виявлення помилок у роботі основних ігрових механік, перевірку коректності взаємодії користувача з віртуальним середовищем та оцінку стабільності додатку під час тривалого використання на автономному VR-шоломі **Meta Quest 2**.

Тестування проводилося шляхом послідовного виконання визначених ігрових сценаріїв у підготовлених рівнях (дитяча кімната та прохідна зона), з використанням стандартних VR-контролерів Meta Quest 2. Особлива увага приділялася перевірці механік, які безпосередньо впливають на ігровий процес та прогрес гравця.

Основні сценарії тестування

У межах перевірки працездатності гри були реалізовані та протестовані такі сценарії:

- запуск гри та ініціалізація VR-середовища;

- коректне позиціонування гравця у віртуальному просторі;
- переміщення та орієнтація користувача за допомогою VR-контролерів;
- виконання стрибків у межах доступної ігрової площі;
- захоплення, утримання та переміщення фізичних об'єктів;
- взаємодія з фізичними кнопками;
- відкривання дверей після активації кнопок;
- переходи між ігровими зонами;
- завершення ігрового рівня.

Перевірка коректності інтерактивних механік

Під час тестування підтверджено, що всі ключові інтерактивні механіки працюють відповідно до закладеної логіки:

- захоплення об'єктів здійснюється за допомогою кнопки стискання контролера та є інтуїтивно зрозумілим для користувача;
- утримання предметів у руці відбувається стабільно без зміщень або втрати позиціонування;
- після відпускання предмети коректно повертаються до фізичної симуляції з увімкненою гравітацією;
- фізичні кнопки реагують на механічне натискання об'єктами або рукою гравця;
- логіка відкривання дверей активується виключно після коректного спрацювання кнопки.

Таким чином, інтерактивні елементи гри забезпечують логічний та послідовний ігровий процес без критичних помилок.

Стабільність роботи гри

Для оцінки стабільності VR-додатку проводилися ігрові сесії тривалістю від 10 до 25 хвилин без перезапуску гри. Під час тестування:

- не було зафіксовано аварійних завершень додатку;
- не виникало втрати відстеження контролерів;

- не спостерігалось зависань або критичних затримок у відповіді на дії користувача;
- ігровий стан зберігав коректність після багаторазових взаємодій з об'єктами.

Окремо перевірялася поведінка гри у випадках інтенсивної взаємодії з великою кількістю об'єктів, що підтвердило стабільну роботу фізичної системи без втрати керованості.

За результатами тестування було розроблено наступну таблицю:

Таблиця 3.1

Результати перевірки працездатності VR-гри

№	Перевірюваний елемент	Очікуваний результат	Фактичний результат
1	Запуск гри	Коректне завантаження сцени	Завантажено без помилок
2	Переміщення гравця	Плавне керування	Працює коректно
3	Стрибок	Реакція на натискання кнопки	Виконується коректно
4	Захоплення предметів	Предмет фіксується у руці	Працює стабільно
5	Відпускання предметів	Відновлення фізики	Коректно
6	Фізична кнопка	Реакція на натискання	Спрацьовує
7	Відкривання дверей	Плавне відкривання	Реалізовано

Результати перевірки працездатності та стабільності свідчать про коректну реалізацію основних ігрових механік VR-гри **Little Strongman**. Гра стабільно працює на платформі Meta Quest 2, забезпечує надійну фізичну взаємодію з об'єктами та не містить критичних помилок, які могли б перешкоджати ігровому процесу або демонстрації проекту.

3.4.2. Аналіз продуктивності VR-додатку

Аналіз продуктивності VR-гри **Little Strongman** проводився з метою оцінки відповідності розробленого додатку апаратним обмеженням автономного VR-шолому **Meta Quest 2**, а також визначення рівня плавності та комфортності ігрового процесу для користувача.

Оскільки VR-додатки висувають підвищені вимоги до стабільності частоти кадрів та затримки відображення, забезпечення належної продуктивності є критично важливим фактором для уникнення дискомфорту, дезорієнтації та VR-нудоти.

Цільові показники продуктивності для Meta Quest 2

Для платформи Meta Quest 2 рекомендованими є такі орієнтовні показники:

- частота оновлення зображення: **72 FPS** (стандартний режим);
- стабільний час кадру без різких піків;
- мінімальні затримки при обробці введення з VR-контролерів;
- відсутність різких просідань продуктивності під час активної фізичної взаємодії.

Дотримання зазначених параметрів дозволяє забезпечити комфортний досвід взаємодії з VR-середовищем.

Методика оцінки продуктивності

Оцінка продуктивності VR-гри здійснювалася в реальних умовах роботи додатку без використання емуляції. Для аналізу використовувалися:

- вбудовані інструменти профілювання Unity;
- моніторинг частоти кадрів під час виконання ігрових сценаріїв;
- спостереження за поведінкою додатку під час інтенсивної взаємодії з об'єктами;
- тривалі ігрові сесії на автономному пристрої.

Тестування проводилося на підготовлених рівнях із найбільшою кількістю активних фізичних об'єктів та інтерактивних елементів.

Результати вимірювання продуктивності

У процесі тестування було встановлено, що VR-додаток демонструє стабільну продуктивність у межах цільових показників платформи. (Детально описано в таблиці 3.4).

Таблиця 3.4

Основні показники продуктивності VR-гри

Параметр	Значення
Середня частота кадрів	83 FPS
Мінімальна частота кадрів	62–68 FPS
Час кадру	Стабільний
Затримка введення	Не відчувається
Аварійні просідання FPS	Не зафіксовано

Незначні короткочасні коливання частоти кадрів спостерігалися під час одночасної взаємодії з декількома фізичними об'єктами, проте вони не мали критичного впливу на загальне сприйняття гри.

Вплив фізичної симуляції на продуктивність

Оскільки ігровий процес значною мірою базується на фізичній взаємодії, особливу увагу було приділено навантаженню фізичного рушія Unity. У процесі тестування встановлено, що:

- використання стандартних параметрів маси та гравітації не створює надмірного навантаження;
- обмежена кількість одночасно активних Rigidbody сприяє стабільній роботі фізики;
- відсутність складних колізійних форм позитивно впливає на продуктивність.

Загалом фізична система гри є збалансованою та відповідає можливостям автономної VR-платформи.

Оптимізаційні рішення, застосовані у проєкті

Для досягнення стабільної продуктивності в процесі розробки були застосовані такі підходи:

- використання оптимізованих 3D-моделей із помірною кількістю полігонів;
- мінімізація кількості одночасно активних фізичних об'єктів;
- відсутність складних постпроцесингових ефектів;
- прості анімації відкривання дверей без використання ресурсоемних систем;
- орієнтація на фізичну взаємодію замість складних скриптових обчислень.

Проведений аналіз продуктивності показав, що VR-гра **Little Strongman** відповідає технічним вимогам платформи Meta Quest 2 та забезпечує стабільний і комфортний ігровий процес. Реалізовані інтерактивні механіки та фізична симуляція не призводять до критичних просідань продуктивності, що свідчить про коректно обрані архітектурні та оптимізаційні рішення.

Висновки до розділу 3

У третьому розділі магістерської роботи було розглянуто процес реалізації та тестування інтерактивних механік VR-гри **Little Strongman**, розробленої для автономного VR-шолому **Meta Quest 2** із використанням рушія Unity. Основну увагу зосереджено на практичній реалізації керування користувачем, фізичної взаємодії з об'єктами та перевірці працездатності й продуктивності VR-додатку.

У межах розділу реалізовано систему керування користувачем у VR-середовищі, що поєднує стандартні компоненти **XR Interaction Toolkit** із власними програмними модулями. Забезпечено коректне позиціонування гравця у віртуальному просторі, реалізовано переміщення, орієнтацію та механіку стрибка, адаптовану до особливостей VR-керування. Використання комбінованого підходу дозволило досягти інтуїтивного та стабільного керування без порушення відчуття занурення.

Реалізація фізичної взаємодії з об'єктами ґрунтувалася на принципах прямої маніпуляції, що є ключовими для VR-ігор. Розроблений механізм захоплення та переміщення предметів забезпечує стабільну фіксацію об'єктів у руках гравця, коректне відновлення фізичних властивостей після відпускання та можливість використання предметів як інструментів для подолання ігрових перешкод. Це дозволило реалізувати ігровий процес, заснований на фізичній логіці та просторовому мисленні.

У межах реалізації інтерактивних механік було розроблено систему фізичних кнопок та логіку відкривання дверей, яка базується на аналізі реального переміщення об'єктів у просторі. Такий підхід забезпечує природну взаємодію з ігровим середовищем та виключає використання умовних або абстрактних способів активації ігрових подій, що позитивно впливає на рівень занурення користувача.

Проведене тестування VR-гри на платформі Meta Quest 2 підтвердило працездатність і стабільність усіх ключових ігрових механік. У процесі експлуатації не було виявлено критичних помилок, аварійних завершень або

збоїв у роботі системи керування та взаємодії з об'єктами. Додатково проведений аналіз продуктивності засвідчив відповідність розробленого VR-додатку апаратним обмеженням автономного VR-шолому, збереження стабільної частоти кадрів та відсутність суттєвих просідань продуктивності.

Отже, результати третього розділу підтверджують ефективність обраних архітектурних та програмних рішень для реалізації інтерактивного VR-середовища. Розроблена система керування, фізичної взаємодії та інтерактивних механік забезпечує цілісний ігровий процес та може бути використана як практичний приклад моделювання інтерактивного середовища у VR-іграх на платформі Meta Quest 2.

ВИСНОВКИ

У межах магістерської роботи було розв'язано актуальну науково-практичну задачу, що полягає у моделюванні інтерактивного середовища у VR-іграх із застосуванням VR-шолому Meta Quest 2. Актуальність дослідження зумовлена стрімким розвитком технологій віртуальної реальності та зростаючою потребою у створенні інтерактивних VR-додатків, здатних забезпечити високий рівень занурення, природну взаємодію користувача з віртуальним простором та стабільну роботу на автономних VR-платформах.

У першому розділі роботи було досліджено теоретичні основи віртуальної реальності та VR-ігор як різновиду інтерактивних програмних систем. Розглянуто ключові поняття та характеристики VR, визначено основні сфери застосування VR-ігор, а також проаналізовано роль фізичної взаємодії та інтерактивності у формуванні відчуття присутності користувача у віртуальному середовищі. Окрему увагу приділено апаратно-програмному забезпеченню для розробки VR-додатків, зокрема архітектурі та технічним можливостям VR-шолому Meta Quest 2. Узагальнення теоретичних положень дозволило сформулювати концептуальну основу для подальшого проєктування інтерактивного VR-середовища.

Другий розділ було присвячено проєктуванню та реалізації VR-гри з інтерактивним середовищем. У межах розділу сформульовано функціональні та нефункціональні вимоги до VR-гри, визначено основні цілі ігрового процесу та концепцію проєкту. Було спроектовано архітектуру VR-гри у рушії Unity з урахуванням обмежень автономної VR-платформи Meta Quest 2. Запропоновано структуру програмного проєкту, визначено ключові підсистеми гри та розроблено архітектуру взаємодії об'єктів. Особливу увагу приділено оптимізаційним рішенням, що забезпечують стабільну продуктивність і коректну фізичну симуляцію. У процесі моделювання ігрового середовища реалізовано віртуальні приміщення, ігрові сцени та

фізичні об'єкти з налаштованими параметрами взаємодії, що формують цілісне інтерактивне VR-середовище.

У третьому розділі здійснено реалізацію та тестування інтерактивних механік VR-гри. Реалізовано систему керування користувачем у VR-середовищі з використанням інструментів XR Interaction Toolkit та власних програмних модулів, що забезпечують переміщення, орієнтацію та стрибок гравця. Розроблено механізми фізичної взаємодії з об'єктами, включаючи захоплення, переміщення та використання предметів у ігровому процесі. Також реалізовано інтерактивні механіки фізичних кнопок і логіку відкриття дверей, засновані на реальному фізичному впливі у віртуальному просторі. Проведене тестування VR-гри на платформі Meta Quest 2 підтвердило коректність реалізації ігрових механік, стабільність роботи додатку та відповідність його продуктивності апаратним вимогам пристрою.

У результаті дослідження було спроектовано, реалізовано та протестовано повноцінну VR-гру з інтерактивним середовищем, яка демонструє практичні можливості моделювання фізичної взаємодії у віртуальній реальності. Розроблений VR-додаток забезпечує інтуїтивну взаємодію користувача з об'єктами, стабільну роботу на автономному VR-шоломі та високий рівень занурення у віртуальний простір.

Практична цінність отриманих результатів полягає в тому, що запропоновані архітектурні та програмні рішення можуть бути використані при розробці інших VR-ігор і VR-додатків для автономних платформ. Матеріали роботи та реалізовані механіки можуть бути застосовані у навчальному процесі під час вивчення технологій віртуальної реальності, розробки ігор та інтерактивних середовищ.

Таким чином, поставлена мета магістерської роботи досягнута, а всі визначені завдання успішно виконані. Отримані результати підтверджують доцільність використання рушія Unity та VR-шолому Meta Quest 2 для моделювання інтерактивного середовища у VR-іграх та відкривають перспективи для подальшого розвитку подібних програмних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Slater M., Sanchez-Vives M. V. Enhancing Our Lives with Immersive Virtual Reality // *Frontiers in Robotics and AI*. — 2016. — Vol. 3. URL: <https://doi.org/10.3389/frobt.2016.00074> (дата звернення: 09.05.2025).
2. Slater M., Usoh M., Steed A. Taking steps: The influence of a walking technique on presence in virtual reality // *ACM Transactions on Computer-Human Interaction*. — 1995. — Vol. 2, No. 3. — P. 201–219. URL: <https://doi.org/10.1145/210079.210084> (дата звернення: 09.05.2025).
3. Steuer J. Defining virtual reality: Dimensions determining telepresence // *Journal of Communication*. — 1992. — Vol. 42, No. 4. — P. 73–93. URL: <https://doi.org/10.1111/j.1460-2466.1992.tb00812.x> (дата звернення: 09.05.2024).
4. Milgram P., Kishino F. A taxonomy of mixed reality visual displays // *IEICE Transactions on Information and Systems*. — 1994. — Vol. E77-D, No. 12. — P. 1321–1329. URL: <https://ieeexplore.ieee.org/document/797914> (дата звернення: 10.05.2024).
5. Jerald J. *The VR Book: Human-Centered Design for Virtual Reality*. — New York : ACM Books, 2015. — 638 p. URL: <https://dl.acm.org/doi/book/10.1145/2792790> (дата звернення: 10.05.2024).
6. Sherman W. R., Craig A. B. *Understanding Virtual Reality: Interface, Application, and Design*. — Boston : Morgan Kaufmann, 2018. — 944 p. (дата звернення: 10.05.2024).
7. Burdea G., Coiffet P. *Virtual Reality Technology*. — Hoboken : Wiley-Interscience, 2003. — 464 p. (дата звернення: 10.05.2024).
8. Bowman D. A., McMahan R. P. Virtual reality: How much immersion is enough? // *Computer*. — 2007. — Vol. 40, No. 7. — P. 36–43. URL: <https://doi.org/10.1109/MC.2007.257> (дата звернення: 12.05.2024).
9. Anthes C., Garcia-Hernandez R. J., Wiedemann M., Kranzlmüller D. State of the art of virtual reality technology // *IEEE Aerospace Conference*. —

2016. URL: <https://doi.org/10.1109/AERO.2016.7500674> (дата звернення: 12.05.2024).

10. Meta Platforms, Inc. Meta Quest 2 Product Documentation. URL: <https://www.meta.com/quest/quest-2/> (дата звернення: 12.05.2024).

11. Meta Platforms, Inc. Meta Quest Developer Documentation. URL: <https://developer.oculus.com/documentation/> (дата звернення: 12.05.2024).

12. Unity Technologies. Unity User Manual 2022 LTS. URL: <https://docs.unity3d.com/Manual/index.html> (дата звернення: 15.05.2024).

13. Unity Technologies. XR Interaction Toolkit Documentation. URL: <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@latest> (дата звернення: 15.05.2024).

14. Unity Technologies. Best Practices for VR Development. URL: <https://learn.unity.com/tutorial/vr-best-practices> (дата звернення: 12.05.2024).

15. Khronos Group. OpenXR Specification. URL: <https://www.khronos.org/openxr/> (дата звернення: 14.05.2024).

16. Meehan M., Insko B., Whitton M., Brooks F. P. Physiological measures of presence in stressful virtual environments // *ACM Transactions on Graphics*. — 2002. — Vol. 21, No. 3. — P. 645–652. URL: <https://doi.org/10.1145/566654.566630> (дата звернення: 14.05.2024).

17. LaValle S. M. Virtual Reality. — Cambridge : Cambridge University Press, 2017. — 414 p.

18. Rizzo A., Koenig S. Is clinical virtual reality ready for primetime? // *Neuropsychology*. — 2017. — Vol. 31, No. 8. — P. 877–899. URL: <https://doi.org/10.1037/neu0000405> (дата звернення: 14.05.2024).

19. Radianti J., Majchrzak T. A., Fromm J., Wohlgenannt I. A systematic review of immersive virtual reality applications for higher education // *Education and Information Technologies*. — 2020. — Vol. 25. — P. 2527–2545. URL: <https://doi.org/10.1007/s10639-019-10018-2> (дата звернення: 14.05.2024).

20. Slater M. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments // *Philosophical Transactions of the Royal*

Society B. — 2009. — Vol. 364. — P. 3549–3557. URL:

<https://doi.org/10.1098/rstb.2009.0138> (дата звернення: 14.05.2024).

21. Unity Technologies. Physics Engine Overview. URL:

<https://docs.unity3d.com/Manual/PhysicsOverview.html> (дата звернення: 14.05.2024).

22. NVIDIA. VR Performance Guidelines. URL:

<https://developer.nvidia.com/vrworks> (дата звернення: 15.05.2024).

23. Oculus VR. Best Practices Guide. URL:

<https://developer.oculus.com/resources/best-practices/> (дата звернення: 15.05.2024).

24. ISO/IEC 25010:2011. Systems and software quality models. URL:

<https://www.iso.org/standard/35733.html> (дата звернення: 15.05.2024).

25. Brooks F. P. What's real about virtual reality? // IEEE Computer Graphics and Applications. — 1999. — Vol. 19, No. 6. — P. 16–27.

26. Unity Technologies. Input System Package Documentation. URL:

<https://docs.unity3d.com/Packages/com.unity.inputsystem@latest> (дата звернення: 20.05.2024).

27. Oculus VR. Hand Tracking Overview. URL:

<https://developer.oculus.com/documentation/unity/unity-handtracking/> (дата звернення: 20.05.2024).

28. Slater M., Wilbur S. A framework for immersive virtual environments (FIVE) // Presence. — 1997. — Vol. 6, No. 6. — P. 603–616.

29. Bowman D. A. et al. 3D User Interfaces: Theory and Practice. — Boston : Addison-Wesley, 2005.

30. Unity Technologies. Mobile VR Optimization Guide. URL:

<https://docs.unity3d.com/Manual/MobileOptimization.html> (дата звернення: 20.05.2024).

31. Oculus VR. Performance Headroom and Profiling. URL:

<https://developer.oculus.com/documentation/unity/unity-perf/> (дата звернення: 20.05.2024).

32. Makransky G., Petersen G. B., Immersive virtual reality and learning: A meta-analysis // *Educational Psychology Review*. — 2019. — Vol. 31. — P. 1-23. URL: <https://doi.org/10.1007/s10648-018-9452-6> (дата звернення: 20.05.2024).
33. Google Developers. VR Design Principles. URL: <https://developers.google.com/vr/design> (дата звернення: 12.05.2024).
34. Unity Technologies. Character Controller Component. URL: <https://docs.unity3d.com/Manual/class-CharacterController.html> (дата звернення: 12.05.2024).
35. Meta Platforms, Inc. Guardian System Overview. URL: <https://www.meta.com/help/quest/articles/in-vr-experiences/oculus-features/guardian/> (дата звернення: 12.05.2024).
36. Mine M. R. Virtual environment interaction techniques // *ACM SIGGRAPH Computer Graphics*. — 1995. — Vol. 29, No. 2. — P. 29–36. URL: <https://doi.org/10.1145/218380.218395> (дата звернення: 20.05.2024).
37. XR Interaction Design Guidelines. URL: <https://developer.oculus.com/design/> (дата звернення: 21.05.2024).
38. Unity Technologies. XR Origin Component. URL: <https://docs.unity3d.com/Packages/com.unity.xr.core-utils@latest> (дата звернення: 21.05.2024).
39. Zhang X., Wang Y. Design and evaluation of interactive virtual environments // *IEEE Computer Graphics and Applications*. — 2018. — Vol. 38, No. 6. — P. 20–30. URL: <https://doi.org/10.1109/MCG.2018.064181> (дата звернення: 21.05.2024).
40. Slater M. Immersion and the illusion of presence in virtual reality // *British Journal of Psychology*. — 2018. — Vol. 109. — P. 431–433.
41. Oculus VR. Spatial Audio SDK Documentation. URL: <https://developer.oculus.com/documentation/unity/audio-spatializer/> (дата звернення: 21.05.2024).

42. Unity Technologies. Audio in VR. URL:
<https://learn.unity.com/tutorial/audio-for-vr> (дата звернення: 21.05.2024).
43. LaViola J. J. et al. 3D User Interfaces. — New York : ACM Press, 2017.
44. Meta Platforms, Inc. Meta XR SDK for Unity. URL:
<https://developer.oculus.com/downloads/> (дата звернення: 21.05.2024).
45. Unity Technologies. Scriptable Render Pipeline Overview. URL:
<https://docs.unity3d.com/Manual/SRP.html> (дата звернення: 21.05.2024).
46. Oculus VR. Unity XR Plugin Framework. URL:
<https://developer.oculus.com/documentation/unity/unity-ovr-plugin/> (дата звернення: 21.05.2024).
47. Carmigniani J., Furht B. Augmented reality: An overview // *Handbook of Augmented Reality*. — Springer, 2011. — P. 3–46. URL:
https://doi.org/10.1007/978-1-4614-0064-6_1 (дата звернення: 21.05.2024).
48. Slater M., Spanlang B., Sanchez-Vives M. First person experience of body transfer in virtual reality // *PLoS ONE*. — 2010. — Vol. 5, No. 5. URL:
<https://doi.org/10.1371/journal.pone.0010564> (дата звернення: 24.05.2024).
49. Jerald J., LaViola J. J. Immersive virtual reality interaction design // *Human–Computer Interaction*. — 2014. — Vol. 29, No. 5–6. — P. 1–50. URL:
<https://doi.org/10.1080/07370024.2014.945394> (дата звернення: 24.05.2024).
50. Oculus VR. Interaction SDK Overview. URL:
<https://developer.oculus.com/documentation/unity/unity-isdk/> (дата звернення: 24.05.2024).
51. Unity Technologies. Physics Optimization Techniques. URL:
<https://learn.unity.com/tutorial/physics-best-practices> (дата звернення: 24.05.2024).
52. Meta Platforms, Inc. Quest Performance Best Practices. URL:
<https://developer.oculus.com/resources/performance/> (дата звернення: 24.05.2024).

53. Bowman D. A., Hodges L. F. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments // ACM Symposium on Interactive 3D Graphics. — 1997.
54. Slater M. Measuring presence: A response to the Witmer and Singer presence questionnaire // Presence. — 1999. — Vol. 8, No. 5. — P. 560–565.
55. Unity Technologies. VR Development Pathway. URL: <https://learn.unity.com/pathway/vr-development> (дата звернення: 24.05.2024).
56. NVIDIA PhysX System Software URL: <https://www.nvidia.com/en-us/drivers/physx/physx-9-19-0218-driver/> (дата звернення: 24.05.2024).
57. Havok Physics URL: <https://www.havok.com/havok-physics/> (дата звернення: 24.05.2024).
58. Wwise, the most advanced, feature-rich interactive audio solution. URL: <https://www.audiokinetic.com/en/wwise/overview/> (дата звернення: 24.05.2024).
59. Qualcomm Technologies, Inc. Snapdragon XR2 Platform for Virtual Reality and Mixed Reality [Електронний ресурс]. URL: <https://www.qualcomm.com/products/application/xr/snapdragon-xr2> (дата звернення: 24.05.2024).

ДОДАТКИ

Додаток А. Вихідний код додатку

```
using UnityEngine;
using UnityEngine.Events;

public class PhysicalButton : MonoBehaviour
{
    public UnityEvent OnButtonPressed;

    public float pressThreshold = 0.015f;

    private Vector3 startPos;
    private bool isPressed = false;

    void Start()
    {
        startPos = transform.localPosition;
    }

    void Update()
    {
        float displacement = startPos.z - transform.localPosition.z;

        if (!isPressed && displacement > pressThreshold)
        {
            isPressed = true;
            OnButtonPressed?.Invoke();
        }

        if (isPressed && displacement < pressThreshold * 0.5f)
        {
            isPressed = false;
        }
    }
}
```

```

}

using UnityEngine;
using UnityEngine.XR;

public class VRGrab : MonoBehaviour
{
    public XRNode controllerNode;
    public string pickupTag = "Pickup";
    public Transform grabPoint;

    private InputDevice device;
    private GameObject objectInHand;
    private Collider objectInTrigger;
    private Vector3 localPosOffset;
    private Quaternion localRotOffset;

    void Start()
    {
        device = InputDevices.GetDeviceAtXRNode(controllerNode);
    }

    void Update()
    {
        if (device.TryGetFeatureValue(CommonUsages.gripButton, out bool gripPressed))
        {
            if (gripPressed && objectInHand == null && objectInTrigger != null)
            {
                GrabObject(objectInTrigger.gameObject);
            }
            else if (!gripPressed && objectInHand != null)
            {
                ReleaseObject();
            }
        }
        if (objectInHand != null)

```

```

    {
        UpdateHeldObjectPose();
    }
}

private void GrabObject(GameObject obj)
{
    objectInHand = obj;
    Rigidbody rb = obj.GetComponent<Rigidbody>();

    if (rb != null)
    {
        rb.isKinematic = true;
        rb.useGravity = false;
    }
    Transform reference = grabPoint != null ? grabPoint : transform;

    localPosOffset = reference.InverseTransformPoint(obj.transform.position);
    localRotOffset = Quaternion.Inverse(reference.rotation) * obj.transform.rotation;
    if (obj.TryGetComponent<ShelfItem>(out ShelfItem shelfItem))
        shelfItem.OnPickedUp();
}

private void ReleaseObject()
{
    Rigidbody rb = objectInHand.GetComponent<Rigidbody>();

    if (rb != null)
    {
        rb.isKinematic = false;
        rb.useGravity = true;
    }

    objectInHand = null;
}

```

```

private void UpdateHeldObjectPose()
{
    Transform reference = grabPoint != null ? grabPoint : transform;

    objectInHand.transform.position = reference.TransformPoint(localPosOffset);
    objectInHand.transform.rotation = reference.rotation * localRotOffset;
}

private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag(pickupTag))
        objectInTrigger = other;
}

private void OnTriggerExit(Collider other)
{
    if (objectInTrigger == other)
        objectInTrigger = null;
}
}

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

public class Jump : MonoBehaviour
{
    [SerializeField] private InputActionReference jumpButton;
    [SerializeField] private float jumpHeight = 2.0f;
    [SerializeField] private float gravityValue = -9.81f;

    private CharacterController _characterController;
    private Vector3 _playerVelocity;

    private void Awake() => _characterController = GetComponent<CharacterController>();

```

```

private void OnEnable() => jumpButton.action.performed += Jumping;

private void OnDisable() => jumpButton.action.performed -= Jumping;

private void Jumping(InputAction.CallbackContext obj)
{
    if (!_characterController.isGrounded) return;
    _playerVelocity.y += Mathf.Sqrt(jumpHeight * -3.0f * gravityValue);
}

private void Update()
{
    if (_characterController.isGrounded && _playerVelocity.y < 0)
    {
        _playerVelocity.y = 0f;
    }

    _playerVelocity.y += gravityValue * Time.deltaTime;
    _characterController.Move(_playerVelocity * Time.deltaTime);
}
}

using Unity.XR.CoreUtils;
using UnityEngine.Assertions;
using UnityEngine.XR.Interaction.Toolkit;

namespace UnityEngine.XR.Interaction.Toolkit.Samples.StarterAssets
{
    public class DynamicMoveProvider : ActionBasedContinuousMoveProvider
    {
        private float originalMoveSpeed = 2.0f;
        private float modifiedMoveSpeed = 35.0f;
        private bool areControllersMoving = false;
        private Vector3 lastLeftControllerPosition;
        private Vector3 lastRightControllerPosition;
    }
}

```



```

public AudioSource footstepAudioSource;

public enum MovementDirection
{
    HeadRelative,
    HandRelative,
}

[Space, Header("Movement Direction")]
[SerializeField]
[Tooltip("Directs the XR Origin's movement when using the head-relative mode. If not set, will automatically find and use the XR Origin Camera.")]
Transform m_HeadTransform;

public Transform headTransform
{
    get => m_HeadTransform;
    set => m_HeadTransform = value;
}

[SerializeField]
[Tooltip("Directs the XR Origin's movement when using the hand-relative mode with the left hand.")]
Transform m_LeftControllerTransform;

public Transform leftControllerTransform
{
    get => m_LeftControllerTransform;
    set => m_LeftControllerTransform = value;
}

[SerializeField]
[Tooltip("Directs the XR Origin's movement when using the hand-relative mode with the right hand.")]
Transform m_RightControllerTransform;

public Transform rightControllerTransform
{

```

```

    get => m_RightControllerTransform;
    set => m_RightControllerTransform = value;
}

```

[SerializeField]

[Tooltip("Whether to use the specified head transform or left controller transform to direct the XR Origin's movement for the left hand.")]

```

MovementDirection m_LeftHandMovementDirection;

```

```

public MovementDirection leftHandMovementDirection
{
    get => m_LeftHandMovementDirection;
    set => m_LeftHandMovementDirection = value;
}

```

[SerializeField]

[Tooltip("Whether to use the specified head transform or right controller transform to direct the XR Origin's movement for the right hand.")]

```

MovementDirection m_RightHandMovementDirection;

```

```

public MovementDirection rightHandMovementDirection
{
    get => m_RightHandMovementDirection;
    set => m_RightHandMovementDirection = value;
}

```

[SerializeField]

[Tooltip("Deadzone value for Y-axis movement of the controllers.")]

```

float m_Deadzone = 0.05f;

```

```

public float deadzone

```

```

{
    get => m_Deadzone;
    set => m_Deadzone = value;
}

```

```

Transform m_CombinedTransform;

Pose m_LeftMovementPose = Pose.identity;
Pose m_RightMovementPose = Pose.identity;

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

    footstepAudioSource = GetComponent<AudioSource>();
    originalMoveSpeed = moveSpeed;

    m_CombinedTransform = new GameObject("[Dynamic Move Provider] Combined Forward Source").transform;

    m_CombinedTransform.SetParent(transform, false);
    m_CombinedTransform.localPosition = Vector3.zero;
    m_CombinedTransform.localRotation = Quaternion.identity;
    forwardSource = m_CombinedTransform;

    lastLeftControllerPosition = leftControllerTransform.position;
    lastRightControllerPosition = rightControllerTransform.position;
}

private bool isObstacleDetected = false;

private void OnTriggerEnter(Collider other)
{
    // Перевірка, чи об'єкт має тег "obstacle"
    if (other.CompareTag("obstacle"))
    {
        isObstacleDetected = true;
        moveSpeed = 0.0f;
    }
}

protected override Vector3 ComputeDesiredMove(Vector2 input)
{
    if (input == Vector2.zero)
    {

```

```

    areControllersMoving = false;
    return Vector3.zero;
}
if (input != Vector2.zero && footstepAudioSource != null && !footstepAudioSource.isPlaying)
{
    footstepAudioSource.Play();
}
if (input == Vector2.zero && footstepAudioSource != null && footstepAudioSource.isPlaying)
{
    footstepAudioSource.Stop();
}
var currentLeftControllerPosition = leftControllerTransform.position;
var currentRightControllerPosition = rightControllerTransform.position;
if (!areControllersMoving &&
    Mathf.Abs(currentLeftControllerPosition.y - lastLeftControllerPosition.y) >= m_Deadzone &&
    Mathf.Abs(currentRightControllerPosition.y - lastRightControllerPosition.y) >= m_Deadzone &&
    Mathf.Sign(currentLeftControllerPosition.y - lastLeftControllerPosition.y) !=
    Mathf.Sign(currentRightControllerPosition.y - lastRightControllerPosition.y))
{
    areControllersMoving = true;
}
else if (areControllersMoving)
{
    areControllersMoving = false;
}
lastLeftControllerPosition = currentLeftControllerPosition;
lastRightControllerPosition = currentRightControllerPosition;
if (!areControllersMoving)
{
    moveSpeed = Mathf.Lerp(moveSpeed, originalMoveSpeed, Time.deltaTime);
}
else
{
    moveSpeed = Mathf.Lerp(moveSpeed, modifiedMoveSpeed, Time.deltaTime);
}
if (m_HeadTransform == null)

```

```

{
    var xrOrigin = system.xrOrigin;
    if (xrOrigin != null)
    {
        var xrCamera = xrOrigin.Camera;
        if (xrCamera != null)
            m_HeadTransform = xrCamera.transform;
    }
}

switch (m_LeftHandMovementDirection)
{
    case MovementDirection.HeadRelative:
        if (m_HeadTransform != null)
            m_LeftMovementPose = m_HeadTransform.GetWorldPose();
        break;

    case MovementDirection.HandRelative:
        if (m_LeftControllerTransform != null)
            m_LeftMovementPose = m_LeftControllerTransform.GetWorldPose();
        break;

    default:
        Assert.IsTrue(false,                                     $"Unhandled
{nameof(MovementDirection)}={m_LeftHandMovementDirection}");
        break;
}

switch (m_RightHandMovementDirection)
{
    case MovementDirection.HeadRelative:
        if (m_HeadTransform != null)
            m_RightMovementPose = m_HeadTransform.GetWorldPose();
        break;

    case MovementDirection.HandRelative:
        if (m_RightControllerTransform != null)
            m_RightMovementPose = m_RightControllerTransform.GetWorldPose();

```

```

        break;

        default:
            Assert.IsTrue(false,                                $"Unhandled
{nameof(MovementDirection)}={m_RightHandMovementDirection}");
            break;
    }

    var move = base.ComputeDesiredMove(input);
    if (footstepAudioSource != null)
    {
        footstepAudioSource.pitch = Mathf.Lerp(1f, 15f, moveSpeed / modifiedMoveSpeed);
    }

    var leftHandValue = leftHandMoveAction.action?.ReadValue<Vector2>() ?? Vector2.zero;
    var rightHandValue = rightHandMoveAction.action?.ReadValue<Vector2>() ?? Vector2.zero;
    var totalSqrMagnitude = leftHandValue.sqrMagnitude + rightHandValue.sqrMagnitude;
    var leftHandBlend = 0.5f;
    if (totalSqrMagnitude > Mathf.Epsilon)
        leftHandBlend = leftHandValue.sqrMagnitude / totalSqrMagnitude;

    var combinedPosition = Vector3.Lerp(m_RightMovementPose.position, m_LeftMovementPose.position,
leftHandBlend);

    var combinedRotation = Quaternion.Slerp(m_RightMovementPose.rotation, m_LeftMovementPose.rotation,
leftHandBlend);

    m_CombinedTransform.SetPositionAndRotation(combinedPosition, combinedRotation);

    return base.ComputeDesiredMove(input);
}
}
}

using UnityEngine;

public class DoorController : MonoBehaviour
{
    public float openAngle = -40f;
    public float speed = 2f;

```

```

private Quaternion closedRot;
private Quaternion openRot;
private bool isOpening = false;

void Start()
{
    closedRot = transform.rotation;
    openRot = Quaternion.Euler(transform.eulerAngles + new Vector3(0, openAngle, 0));
}

public void OpenDoor()
{
    Debug.Log("BUTTON PRESSED → OpenDoor()");
    isOpening = true;
}

void Update()
{
    if (isOpening)
    {
        transform.rotation = Quaternion.Lerp(transform.rotation, openRot, Time.deltaTime * speed);
    }
}

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

public class CameraReset : MonoBehaviour
{
    public Transform head;
    public Transform origin;
    public Transform target;

```

```

public InputActionProperty recenterButton;

public void Recenter()
{
    Vector3 offset = head.position - origin.position;
    offset.y = 0;
    origin.position = target.position - offset;

    Vector3 targetForward = target.forward;
    targetForward.y = 0;
    Vector3 cameraForward = head.forward;
    cameraForward.y = 0;

    float angle = Vector3.SignedAngle(cameraForward, targetForward, Vector3.up);
    origin.RotateAround(head.position, Vector3.up, angle);
}

void Update()
{
    if(recenterButton.action.WasPressedThisFrame())
    {
        Recenter();
    }
}

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

public class PlayerController : MonoBehaviour
{
    private CharacterController controller;
    private CapsuleCollider col;

```



```

//private Score score;
private Vector3 dir;
[SerializeField] private float jumpForce;
[SerializeField] public int coins;
[SerializeField] private GameObject losePanel;
[SerializeField] private GameObject SscoreText;
[SerializeField] public Text coinsText;
//[SerializeField] private Score scoreScript;

void Start()
{
    controller = GetComponent<CharacterController>();
    col = GetComponent<CapsuleCollider>();
    //score = SscoreText.GetComponent<Score>();
    //score.scoreMultiplier = 1;
    Time.timeScale = 1;
    coins = PlayerPrefs.GetInt("coins");
    coinsText.text = "coins: " + coins.ToString();
}

private void Update()

{

}

private void OnControllerColliderHit(ControllerColliderHit hit)
{
    if (hit.gameObject.tag == "obstacle")
    {
        //losePanel.SetActive(true);
        //int lastRunScore = int.Parse(scoreScript.scoreText.text.ToString());
        //PlayerPrefs.SetInt("lastRunScore", lastRunScore);
        Time.timeScale = 0;
    }
}

```

```

public void OnTriggerEnter(Collider other)
{
    if (other.gameObject.tag == "Coin")
    {
        coins++;
        PlayerPrefs.SetInt("coins", coins);
        coinsText.text = "coins: " + coins.ToString();
        Destroy(other.gameObject);
    }

    if (other.gameObject.tag == "BonusStar")
    {
        StartCoroutine(StarBonus());
        Destroy(other.gameObject);
    }
}

```

```

private IEnumerator StarBonus()
{
    //score.scoreMultiplier = 2;

    yield return new WaitForSeconds(5);

    //score.scoreMultiplier = 1;
}
}

```

Цієї сторінки