Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: *"Розроблення анімації персонажів для VR-ігор із використанням Blender та Unity"*

Виконала: студентка 4 курсу групи КН-41
спеціальності 122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Сапса Ю. П.
(прізвище та ініціали)
Керівник Сінкевич О. В.
(прізвище та ініціали)
Керівник Думанський О. І.
(прізвище та ініціали)
Рецензент Сторожук О. Л.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 "Комп'ютерні науки"

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

Борецька І.Б.

"10" червня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Сапса Юлія Петрівна

(прізвище, ім'я, по батькові)

1. Тема роботи: *"Інформаційна технологія розроблення анімації персонажів для VR-ігор із використанням Blender та Unity"*

керівник роботи: *Сінкевич О. В., старший викладач кафедри комп'ютерних наук, PhD, Думанський О. І., доцент кафедри комп'ютерних наук, к.ф.-м.н.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "15" листопада 2024 року №С-882

2. Термін подання студентом роботи 10 червня 2025 року.

3. Вихідні дані до роботи: мова програмування C#; середовище для 3D-моделювання Blender; рушій для створення комп'ютерних ігор Unity.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

1) Стан проблемної області; 2) Інформаційне та математичне забезпечення; 3) Програмне та технічне забезпечення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): *статистичні діаграми для аналізу VR-індустрії; UML-діаграма проєкту; блок-схема алгоритму взаємодії гравця зі зброєю; демонстрація моделювання 3D-персонажа; інтерфейс розробленого проєкту;*

6. Дата видачі завдання 18 листопада 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Вивчення необхідних технологій та підготовка ресурсів, розширень та асетів	18.11.2024-10.01.2025	Виконано
2	Розробка 3D-моделі персонажа у Blender	15.01.2025-21.02.2025	Виконано
3	Імпорт створених моделей, створення ігрового рівня	23.02.2025-10.03.2025	Виконано
4	Програмування алгоритмів та механізмів пересування, стрільби	15.03.2025-20.04.2025	Виконано
5	Тестування проєкту	21.04.2025-25.04.2025	Виконано
6	Оформлення пояснювальної записки	27.04.2025-13.05.2025	Виконано

Студент Сапса Ю.П.

Керівник роботи Сінкевич О.В.

Керівник роботи Думанський О.І.

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 78 сторінок, з яких 40 сторінок основного тексту, що включає 29 рисунків, 2 таблиці, 13 джерел та 3 додатків, які розміщено на 28 сторінках.

У роботі розглянуто процес створення прототипу VR-гри з інтерактивними персонажами з використанням сучасних інструментів 3D-моделювання Blender та ігрового рушія Unity.

Значну увагу приділено побудові інформаційної та концептуальної моделей взаємодії користувача з віртуальним середовищем, створенню анімованого 3D персонажа, та інтеграції об’єктів у середовище розробки Unity. Реалізовано базові механіки персонажа – рух, захоплення об’єктів, стрільба та реакція віртуальних NPC на дії з боку гравця.

Важливим компонентом роботи стало забезпечення взаємодії з контролерами віртуальної реальності та базової поведінки персонажів.

У результаті виконаної роботи створено функціональний VR-прототип гри, що може використовуватись як навчальна чи демонстраційна платформа для ознайомлення з основами VR-розробки.

Ключові слова: віртуальна реальність, ригінг, анімація, XR Interaction Toolkit, C#.

ABSTRACT

The bachelor's thesis consists of 74 pages, including 40 pages of main text, which contains 29 figures, 2 tables, 13 references, and 3 appendices presented on 28 pages. The work explores the process of developing a VR game prototype with interactive characters using modern 3D modeling tools such as Blender and the Unity game engine. Particular attention is paid to building both informational and conceptual models of user interaction within a virtual environment, designing an animated 3D character, and integrating assets into the Unity development environment. Core character mechanics were implemented, including movement, object grabbing, shooting, and non-player characters (NPCs) reacting to player actions.

An important aspect of the project was enabling interaction with virtual reality controllers and implementing the basic behavioral logic of the characters. As a result, a

functional VR game prototype was developed, which may serve as an educational or demonstrational platform for learning the fundamentals of VR development. Keywords: virtual reality, Unity, Blender, rigging, animation, XR Interaction Toolkit, C#.

ТЕХНІЧНЕ ЗАВДАННЯ

Мета дипломної роботи полягає у розробці функціонального прототипу VR-гри (міні-шутера) з інтерактивними персонажами, що дозволить продемонструвати комплексне використання інструментів 3D-моделювання, анімації та сучасного рушія розробки ігор Unity, для створення віртуального середовища з базовою взаємодією користувача.

Основні завдання для реалізації функціональності проєкту:

• Створення 3D-моделі персонажа за допомогою Blender.

• Застосувати методи скелетної анімації для оживлення персонажів і створення базових рухів.

• Імпорт моделей та анімацій у Unity з налаштуванням матеріалів і середовища.

• Інтегрувати підтримку VR (за допомогою XR Interaction Toolkit або OpenXR), включаючи:

- систему управління рухом (пересування, обертання),
- реалізацію взаємодії з об’єктами (вибір, підняття, активація), ○ запуск анімацій персонажів у відповідь на дії користувача. • Реалізація сценаріїв взаємодії гравця з персонажами або об’єктами гри. Очікувані результати:

• Прототип VR-гри з інтерактивними персонажами.

• Інтеграція та тестування гри у середовищі Unity.

• Демонстрація практичних навичок роботи з Blender, Unity, VR інструментарієм та основами геймдизайну.

Використані технології та інструменти:

• Blender – для створення моделі ворожого персонажа та його анімації. • Unity – багатоплатформний рушій для створення ігор.

• XR Interaction Toolkit – для інтеграції VR-функцій.
• C# – мова програмування для реалізації ігрової логіки.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ 1. VR (Virtual Reality) – віртуальна реальність. Технологія, що створює ілюзію присутності користувача в комп’ютерно змодельованому тривимірному середовищі шляхом використання спеціальних пристроїв (шоломів, контролерів, датчиків руху тощо).

2. AR (Augmented Reality) – доповнена реальність. Технологія, що поєднує реальний світ із віртуальними об'єктами, які відображаються у реальному часі через камеру пристрою.

3. Ригінг (Rigging) – процес створення скелету для 3D-моделі з метою анімації, який включає побудову кісток (armature), встановлення зв’язків між ними та прив’язку до геометрії моделі.

4. Інверсна кінематика (IK, Inverse Kinematics) – метод анімації, при якому розраховується положення суглобів у скелеті моделі для досягнення заданої позиції кінцівки (наприклад, розташування руки на об’єкті).

5. OpenXR – відкритий стандарт для розробки додатків доповненої та віртуальної реальності, який забезпечує кросплатформену підтримку VR/AR пристроїв через єдиний API.

6. ARCore – платформа Google для створення додатків доповненої реальності на пристроях з Android, яка забезпечує визначення простору, відстеження рухів і розміщення віртуальних об’єктів у фізичному середовищі.

7. ARKit – платформа Apple для розробки AR-додатків на пристроях з iOS, яка надає засоби для відстеження руху, розпізнавання площин та інтеграції 3D об’єктів у реальний світ.

8. Інтерактори – елементи в системі XR (Extended Reality), які дозволяють користувачу взаємодіяти з віртуальними об'єктами, зокрема за допомогою контролерів або рук (наприклад, натискання, захоплення, наведення).

9. Інтерактиви – елементи віртуального середовища, з якими може відбуватись взаємодія; це можуть бути об'єкти, персонажі, інтерфейсні компоненти, які реагують на дії користувача.

10. Локомоція – спосіб переміщення користувача у віртуальному середовищі, що включає фізичну ходу, переміщення джойстиком, телепортацію, пересування сцени тощо.

11. LINQ (Language Integrated Query) – інтегрована у мову програмування C# технологія запитів до колекцій даних (масивів, списків, баз даних), що забезпечує зручний синтаксис для пошуку, сортування та фільтрації.

ВСТУП

Актуальність теми дослідження. У сучасному світі віртуальна реальність (VR) стрімко розвивається та стає важливою складовою індустрії розваг, освіти, медицини та інших сфер. Особливу популярність VR здобула у галузі комп'ютерних ігор, де дозволяє створювати повноцінний ефект присутності, забезпечуючи новий рівень занурення та взаємодії з цифровим середовищем. VR-ігри відкривають для користувачів унікальний досвід, який неможливо отримати у звичайних 2D- або навіть 3D-іграх, надаючи змогу не просто спостерігати за ігровим світом, а безпосередньо бути його частиною.

З розвитком VR-технологій зростає і попит на фахівців, здатних створювати якісний VR-контент. У цьому контексті важливе значення мають сучасні інструменти розробки, зокрема Unity – один із найпопулярніших рушіїв для створення VR додатків, що підтримує широкі можливості для інтеграції VR-обладнання та роботи з анімаціями. Не менш важливим є і Blender – безкоштовний, потужний інструмент для створення 3D-моделей та анімацій, який широко використовується як індивідуальними розробниками, так і великими студіями.

Розробка VR-ігор із використанням Blender та Unity дає змогу не лише опанувати базові принципи моделювання, анімації та програмування, а й на практиці реалізувати повноцінний інтерактивний проєкт. Саме тому створення VR-гри з інтерактивними персонажами є актуальним завданням, яке поєднує в собі творчість, технічні навички та розуміння сучасних вимог до ігрової індустрії.

Об'єкт дослідження – процес розробки VR-додатку з інтерактивним ігровим середовищем.

Предмет дослідження – технології моделювання, анімації та програмування, які використовуються при створенні VR-ігор на основі Blender і Unity. **Мета роботи –** створення VR-мінішутера для демонстрації основних принципів розробки VR-ігор, включаючи створення 3D-моделей, їх анімацію, імпорт у VR середовище, а також реалізацію базової взаємодії з гравцем.

Завдання – прототип VR-гри з інтерактивними персонажами, використовуючи

сучасні інструменти 3D-моделювання та рушій розробки ігор Unity; створення 3D моделі ворожого персонажу у Blender, додавання анімацій та їх інтеграція у Unity.

Практичне значення – практичне значення дипломної роботи полягає в розробці навчального VR-прототипу, що демонструє повний цикл створення інтерактивного додатку: від моделювання та анімації до реалізації користувацької взаємодії у віртуальному середовищі. Він може слугувати базою для подальшого вдосконалення та створення більш складної VR-гри з розширеним функціоналом.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз стану індустрії VR-ігор.

Ігрова індустрія, попри її вагому популярність у сучасному світі, має глибокі історичні засади, які сягають ще перших ігрових симуляцій 1950-х років. З плином часу вона трансформувалася з експериментальних проєктів у вагомий сегмент культурної індустрії, тісно пов'язаний із кіноіндустрією, телебаченням та індустрією розваг загалом. Сучасний стан ринку характеризується великою різноманітністю жанрів та ігрових платформ, а також стрімким зростанням цифрової дистрибуції та мобільного геймінгу.

Поступово змінюється і саме призначення ігор – окрім розважальної функції, вони все частіше використовуються як засіб навчання, тренування й професійної підготовки. У цьому контексті особливе місце займають освітні та серйозні ігри, які сприяють розвитку пізнавальних навичок, критичного мислення та інтерактивного досвіду. Віртуальна реальність як інструмент подібних рішень відкриває нові горизонти для застосування ігор у навчальному процесі, тренажерах і симуляційних системах.

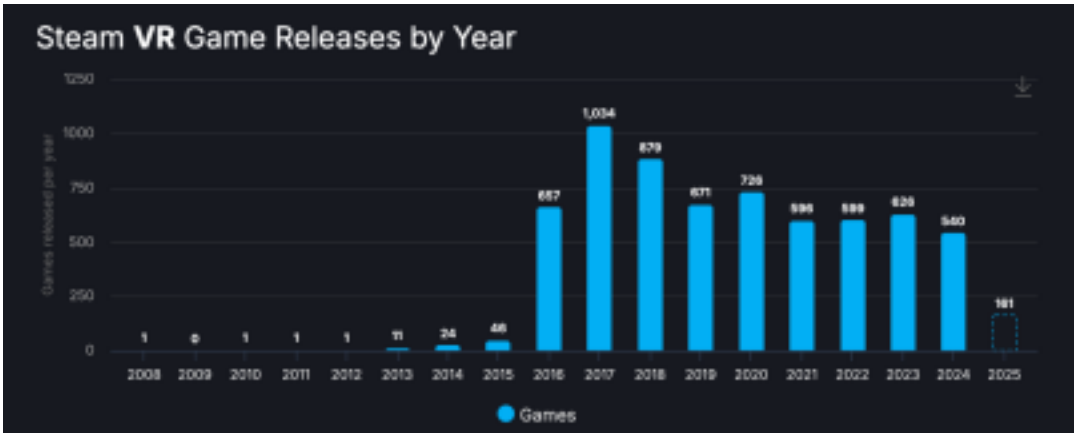

Рис.1.1. Випуск VR-ігор на платформі Steam з 2008-2025 рр. [1]

Історичні дані, що зображені на графіку демонструють значний прогрес реалізації та випуску VR-ігор на платформі Steam. Зокрема, у 2017 році офіційно було випущено 1 034 гри, що пов'язане зі значним обсягом інвестицій у сферу VR технологій. Для прикладу, компанія Unity Engine залучила близько 400 мільйонів доларів прагнучи вдосконалити свої платформи розробки VR та AR [2, переклад].

Така фінансова підтримка сприяла створенню більш досконалих інструментів розробки, що дозволило розробникам ефективніше створювати якісний VR-контент. Незважаючи на весь "technological boom", який стався протягом 2016-2019 рр., обсяг

випуску VR-ігор зазнав вагомого спаду. У індустрії бракує визначних VR-ігор (які ще можна назвати “killer app”), які могли б стимулювати масове прийняття технологій та залучати широку аудиторію. Також варто зазначити про вартість VR обладнання, яке не лише є дорогим, але й потребує потужне апаратне забезпечення.

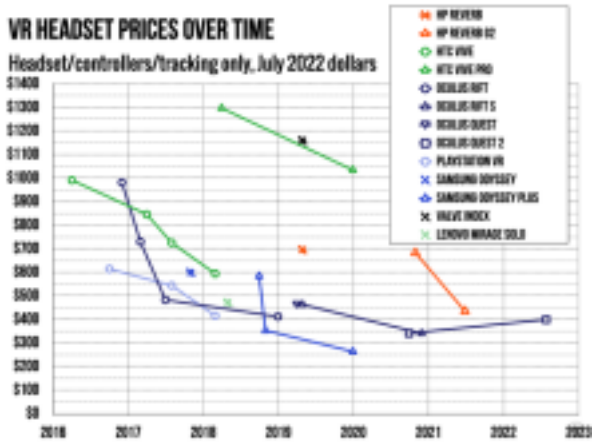

Рис.1.2. Вартість VR обладнання 2016-2023 рр. [3]

Протягом цих років ринок поділився умовно на:

• Преміум-сегмент (HTC Vive Pro, Valve Index, HoloLens) – початкові ціни $1000+.

• Масовий сегмент (Oculus Quest, Rift, PSVR) – стабільне зниження до ~$300–$500.

Виходячи з ринкових цін, можна зробити висновок, що більшість середньостатистичних користувачів не можуть дозволити собі придбати техніку за такі кошти, і досить часто, окрім самої гарнітури, користувачі часто повинні інвестувати в потужний комп'ютер або ігрову консоль, що додає витрати до загальної вартості володіння VR-системою.

Говорячи про сучасний стан речей, попри недоліки VR-систем у їх вартості та не надто актуальних ігор, ця сфера все одно користується чималим попитом серед шанувальників комп’ютерних ігор.

Розмір ринку віртуальної реальності у сфері ігор зазнав стрімкого зростання в останні роки. Очікується, що він збільшиться з 19,24 мільярда доларів США у 2024

році до 24,51 мільярда доларів у 2025 році, демонструючи середньорічний темп зростання (CAGR) на рівні 27,4% [4]. Такий стрімкий розвиток пов’язаний зі зростанням інтересу користувачів до імерсивних технологій, активним розширенням

провідних ігрових платформ, покращенням якості контенту та ігрових продуктів, а також удосконаленням апаратного забезпечення.

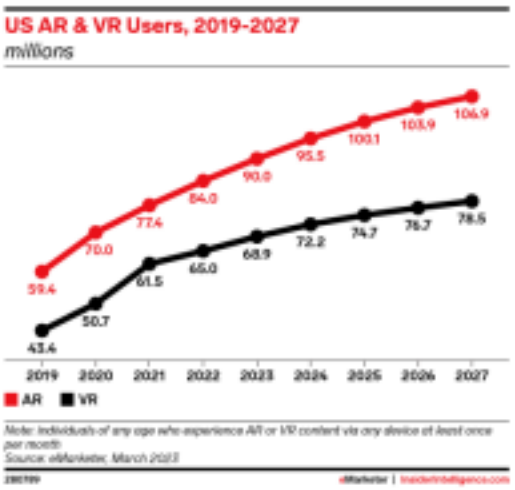

Рис.1.3. Прогнозування активних користувачів AR&VR технологій у США, 2019-2027 рр. (у млн. осіб) [5]

У подальшій перспективі ринок віртуальної реальності в ігровій індустрії має всі передумови для суттєвого зростання – прогнозується, що його обсяг сягне 73,24 мільярда доларів США до 2029 року, зі зростанням на середньорічному рівні 31,5% [4]. Зі збільшенням частки ринку, прогнозується і збільшення активних користувачів. Зокрема, у 2027 році плановий показник активних користувачів сягатиме 106,9 млн.осіб для AR- та 78,5 млн.осіб для VR-технологій відповідно. Прогнозоване зростання ринку пов’язується з активним розвитком хмарних ігрових платформ, зростанням популярності соціальних та багатокористувацьких форматів, а також із технічним удосконаленням контролерів і пристроїв введення. Серед основних тенденцій можна виділити VR-ігри, орієнтовані на сюжетну складову, фітнес-додатки у віртуальній реальності, активну участь користувачів у створенні контенту та модифікацій, впровадження технологій тактильного зворотного зв’язку, а також розвиток стримінгових платформ і сервісів за підпискою для VR-продуктів.

1.2. Огляд аналогів VR-шутерів.

Серед найбільш популярних шутерів-аналогів варто відзначити такі ігри як “Hot Dogs, Horseshoes & Hand Grenades” (H3VR) та “HARD BULLET”. “Hot Dogs, Horseshoes & Hand Grenades” – це симулятор стрільби у VR середовищі, що

базується на високій деталізації та фізично точною взаємодією зі зброєю. Гра пропонує понад десяток різних режимів для практики стрільби, експериментів із великою кількістю видів вогнепальної зброї, вибухівки та мішеней.

Рис.1.4. Геймплей гри "Hot Dogs, Horseshoes & Hand Grenades" [Офіційний сайт гри у Steam]

Ключовою особливістю є максимально реалістична механіка перезаряджання, де кожна одиниця зброї потребує індивідуального підходу. Гравцю потрібно вручну вставляти магазини, натискати затвори, витягувати патрони тощо – усе це сприяє глибокому зануренню, але може бути складним для новачків, які вперше користуються VR технікою.

"HARD BULLET" – це динамічний VR-шутер з високим рівнем жорстокості та насильства, орієнтований на досвідчених гравців, які шукають екстремальні відчуття у віртуальному середовищі. Гра не має сюжету, однак дозволяє гравцю вільно експериментувати з фізикою тіла, об'єктами оточення та великою кількістю зброї.

Рис.1.4. Геймплей гри "HARD BULLET " [Офіційний сайт гри у Steam] 15

Особливістю цієї гри є її надмірна жорстокість та брутальність, оскільки гравець може кидати ворогів, штовхати їх на шипи, використовувати довколишні предмети як імпровізовану зброю, що супроводжується великою кількістю візуальних ефектів пошкоджень. Через складність геймплею та надмірну жорстокість, гра не надто підходить для новачків, а тим паче для молодшої аудиторії.

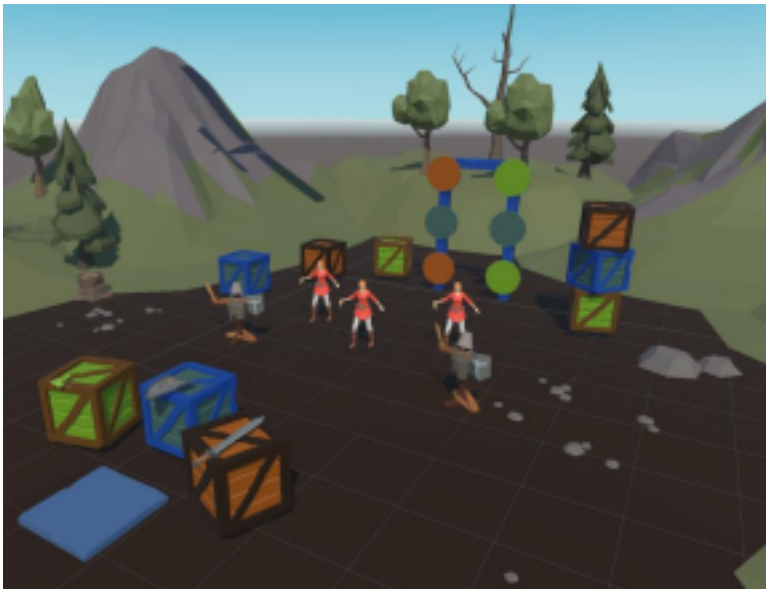

Рис.1.5. Інтерфейс розробленого проєкту "Low Poly Strike"

На відміну від оглянутих аналогів, проєкт, що був розроблений у межах дипломної роботи, має більш ознайомчий характер та призначений для початківців у VR (зокрема дітей/підлітків). Основна мета гри – дати змогу користувачу поступово звикнути до базових елементів управління, таких як рух, захоплювання елементів, взаємодія з об'єктами середовища та механіка стрільби. Гра не містить сцен надмірної жорстокості й фокусується на безпечному навчальному досвіді. Таким чином, запропонований прототип може слугувати підґрунтям для подальшого вивчення VR шутерів і розвитку навичок взаємодії у віртуальному середовищі.

1.3. Реалізація функціоналу програмного продукту.

Для реалізації програмного продукту необхідно виконати низку етапів, кожен з яких має конкретні технічні та функціональні завдання. Основні напрямки охоплюють 3D моделювання ворожих персонажів, анімації, розробку ігрової логіки, налаштування VR-взаємодії користувача з середовищем, тестування проєкту.

Основні етапи розробки проєкту:

1. Розробка концепції VR-гри:

a. Визначення цільової аудиторії => діти та початківці;

b. Вибір стилістики гри => low poly (низько полігональний), просте середовище без деталізованої графіки;

c. Розробка вигляду ворожого персонажа => низько полігональний, без виражених ознак агресії, у стилі рицаря;

2. Створення 3D ворожого персонажа у Blender:

a. Моделювання персонажа => створення моделі, скульптинг, деталізація; додавання матеріалів;

b. Додавання скелету персонажа та додавання анімації готовності до бою;

c. Експортування моделі у .fbx форматі у Unity;

3. Побудова VR-середовища:

a. Імпорт ассетів та моделей;

b. Розміщення об’єктів та створення базового рівня;

c. Додавання colliders, фізики для об’єктів та встановлення матеріалів;

4. Реалізація механік взаємодії з VR:

a. Підключення XR-плагінів (XR Interaction Toolkit, OpenXR);

b. Налаштування систем переміщення, захоплення об’єктів, стрільби;

5. Програмування логіки гри мовою C#:

a. Реалізація алгоритму захоплення об’єктів;

b. Реалізація логіки стрільби;

c. Створення алгоритму для спрацювання анімацій;

6. Тестування та оптимізація прототипу:

a. Перевірка роботи програми за допомогою XR Device Simulator;

b. Оптимізація алгоритмів та аналіз управління;

1.4. Висновки до розділу 1.

У даному розділі було проаналізовано сучасний стан галузі VR-ігор, що базується на історичних аспектах, статистичних даних та прогнозуванню майбутніх результатів розвитку цієї сфери. Базуючись на цьому аналізі, було обґрунтовано доцільність створення VR-проєкту та описано етапи його реалізації.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Виділення ключових компонентів ігрової сцени.

У цьому підрозділі описуються основні компоненти віртуального середовища, з якими взаємодіє користувач у межах реалізованого VR-додатку. Ці компоненти відіграють важливу роль у побудові логіки гри, а також є основою для подальшого концептуального та інформаційного моделювання.

Таблиця 2.1. Основні об’єкти проєкту

Об’єкт	Дії
Гравець	Діє у VR середовищі через контролери та гарнітуру
Ворожі персонажі	Реагують на дії з боку гравця (стрільба, удар мечем)
Об’єкти сцени	Мішені, ящики, манекени, що реагують на дії гравця (зіткнення, оновлення сцени при дії тригерів)
Навколишнє середовище	Простір, у межах якого реалізується ігрова логіка

Основні дії користувача реалізуються через фізичну активність у межах ігрової сцени та взаємодію з об’єктами за допомогою VR-контролерів:

• Фізичне переміщення гравця у межах ігрового простору – здійснюється за допомогою контролерів. Гравець може пересуватися, повертатися та обходити перешкоди.

• Захоплення об’єктів – реалізується за допомогою кнопок на контролерах. Гравець має можливість підіймати зброю/щит, переміщати їх та міняти положення в руках.

• Реалізація стрільби – механізм стрільби реалізується через натискання тригерів на VR-контролерах, що дозволяє гравцеві здійснювати постріли у віртуальному середовищі.

2.2. Розробка 3D моделі ворожого персонажа у Blender.

На етапі візуального наповнення ігрового середовища, важливою складовою є створення якісної тривимірної моделі персонажа, з якою буде взаємодіяти гравець. У межах даного проєкту було обрано середовище моделювання Blender. У цьому підрозділі будуть продемонстровані основні кроки при розробці персонажа.

На початковому етапі моделювання було виконано побудову базової сітки (mesh) для майбутнього одягу персонажа. Як основа використовувалась готова модель людського тіла, до якої планувалося додати елементи одягу вручну.

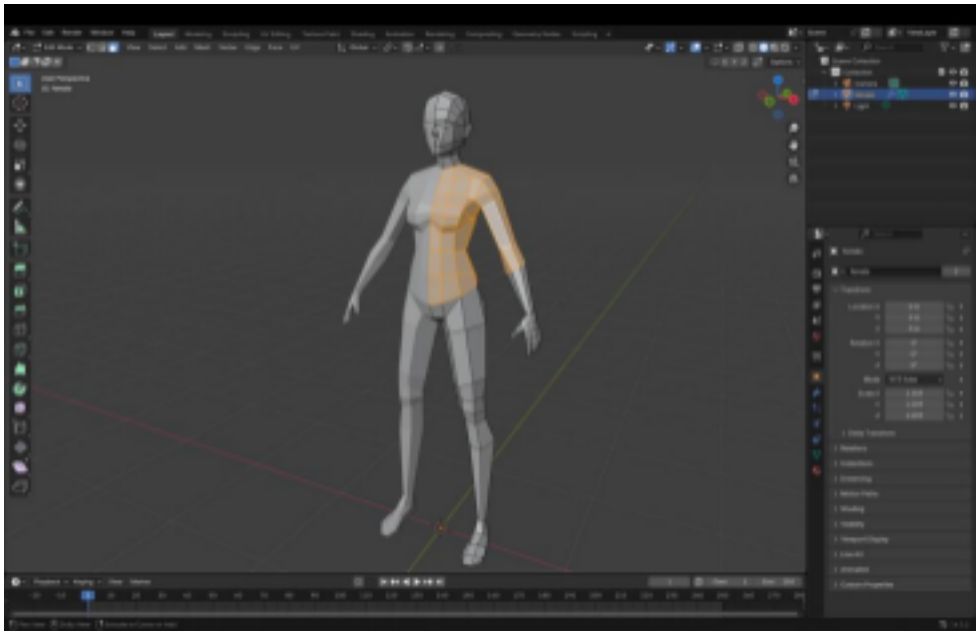

Рис.2.1. Створення сітки для моделювання одягу персонажа [виконано автором]

Етапи виконання:

• У режимі редагування (Edit Mode) було виділено полігони в області тулуба та рук, які слугуватимуть формою для майбутньої броні.

• Виділену геометрію скопійовано та відокремлено в новий об’єкт (P → Selection), що дозволило працювати з елементом як із незалежною частиною. Цей підхід дає змогу зберегти точне прилягання одягу до тіла моделі, не створюючи його з нуля.

Після формування базової сітки для одягу було здійснено редагування та закриття геометричних проміжків між тілом персонажа та майбутнім одягом. Це важливо як для забезпечення візуальної цілісності моделі, так і для уникнення візуальних артефактів під час анімації.

Наступні зображення наочно демонструють логіку побудови:

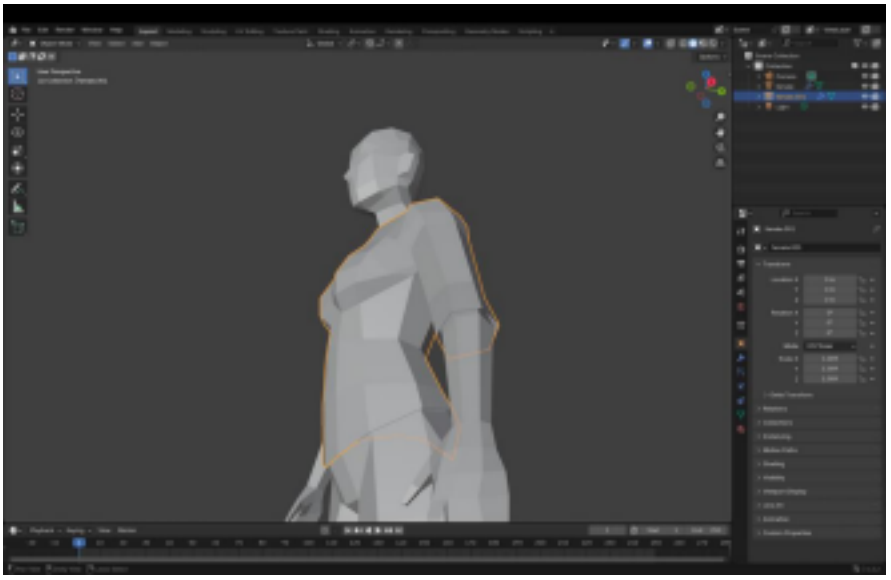

Рис.2.2. Виділення частини тіла персонажа для створення одягу [виконано автором]

Для з’єднання відкритих частин сітки використовувалась команда Extrude (E) у режимі редагування. Вона дозволила створювати нові полігони шляхом витягування країв або поверхонь, що забезпечує плавне злиття з іншими частинами моделі.

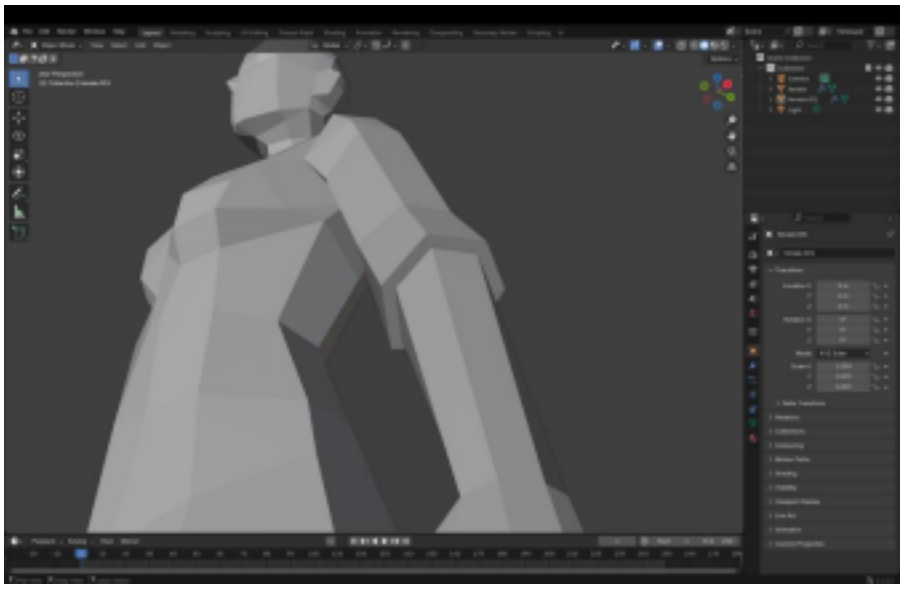

Рис.2.3. Закриття геометричних проміжків на моделі за допомогою команди Extrude [виконано автором]

Такий метод оптимізує роботу зі скіннінгом та ригінгом, оскільки топологія зберігає правильний розподіл полігонів та дозволяє ефективно контролювати деформацію в подальших етапах анімації.

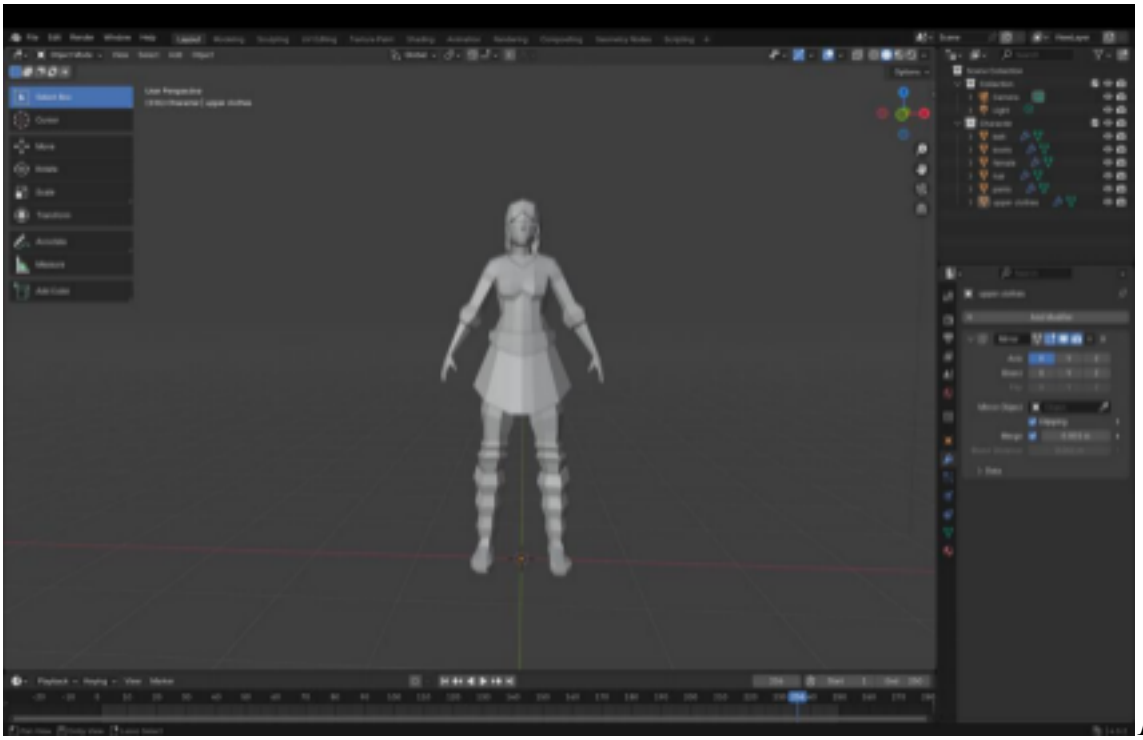

Рис.2.4. Базове моделювання персонажа у Blender з використанням модифікатора Mirror [виконано автором]

Для оптимізації процесу моделювання симетричного персонажа у Blender було використано модифікатор Mirror, що дозволяє автоматично дублювати геометрію об’єкта відносно однієї або кількох координатних осей.

На рисунку 2.4 зображено базову стадію моделювання ворожого персонажа з активованим модифікатором Mirror для осей X, Y та Z, що дозволяє досягти абсолютної симетрії під час формування форм одягу, кінцівок та голови. У правій панелі властивостей видно, що модифікатор застосовується до об'єкта upper clothes, що формує верхню частину тіла персонажа. Для коректної роботи було також увімкнено параметр Clipping, який запобігає накладенню вершин по осі симетрії, забезпечуючи акуратний злив центральної геометрії.

Після завершення базового моделювання наступним етапом є створення скелету персонажа (ригінгу), що дозволяє анімувати його рухи у середовищі Blender або під час експорту до Unity.

Рис.2.5. Додавання гуманоїдного скелету Human Meta-Rig до моделі персонажа [виконано автором]

На даному етапі до моделі персонажа в середовищі Blender було додано арматуру (скелет) із застосуванням стандартного шаблону “Human (Meta-Rig)”, що містить набір основних кісток для гуманоїдної моделі. Для цього використовувалась команда:

Add → Armature → Human (Meta-Rig).

Після додавання скелету було виконано його масштабування (команда S) для приведення до пропорцій персонажа, а також позиціонування в просторі за допомогою інструментів G (переміщення) та R (обертання), щоб арматура відповідала розташуванню частин тіла моделі.

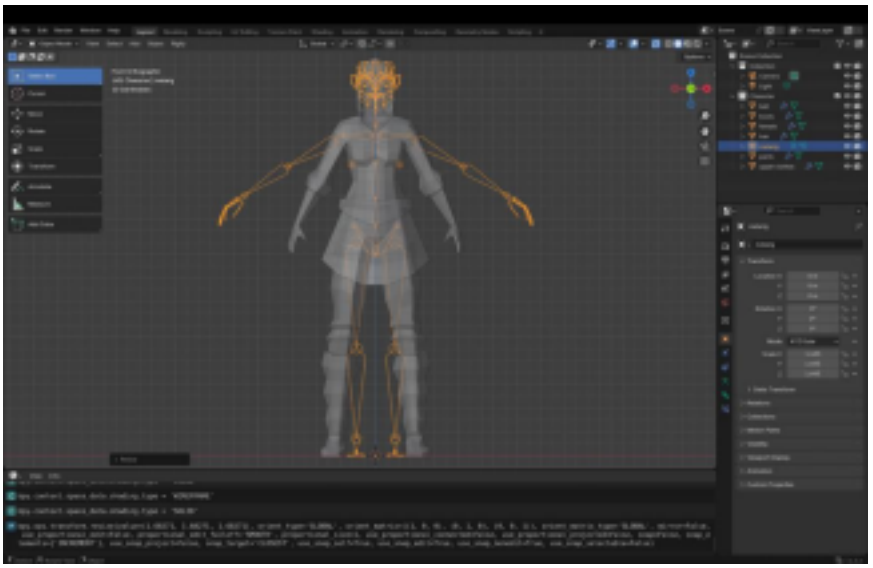

Рис.2.6. Масштабування та позиціонування кісток скелету відповідно до частин тіла персонажа [виконано автором]

Для забезпечення симетрії при створенні скелету використовувалась функція

Armature → *Symmetrize*, що дозволяє автоматично віддзеркалити кістки з однієї половини тіла на іншу. Це значно пришвидшує процес, виключаючи потребу у ручному дублюванні та розміщенні елементів.

Рис.2.7. Скелет персонажа у Blender [виконано автором]

Перш ніж розпочинати створення кісток, необхідно переконатися, що модель є технічно готовою до ригінгу:

1. Перевірка нормалей: нормалі визначають напрямок поверхні й критичні для коректної анімації. У разі їх неправильної орієнтації, об’єкти можуть деформуватись під час руху. За допомогою команди Shift+N відбувається автоматичне перерахування нормалей.

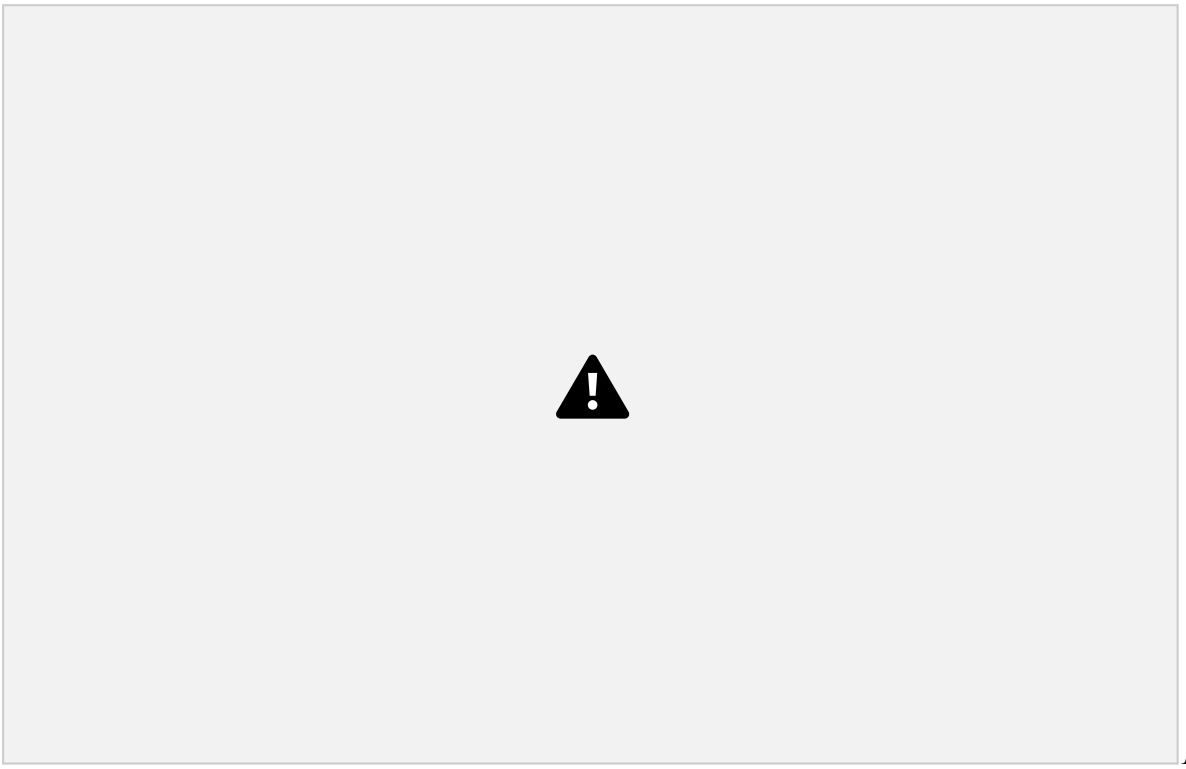

Рис. 2.8. Перерахунок нормалей моделі у Blender [виконано автором] 2. Виявлення подвійної геометрії (Double Geometry): це визначає наявність накладених полігонів може спричинити артефакти під час анімації. За допомогою команди M – Merge by Distance здійснюється об'єднання вершин, що накладаються, та очищається сітка.

Далі після створення кісток їх потрібно розташувати всередині моделі. Після створення скелету об’єкт (модель персонажа) об’єднується з арматурою через батьківський зв’язок за допомогою команди *Ctrl + P => With Automatic Weights*. Це створює зв’язок між вершинами моделі та відповідними кістками, що дозволяє кісткам впливати на деформацію тіла під час анімації.

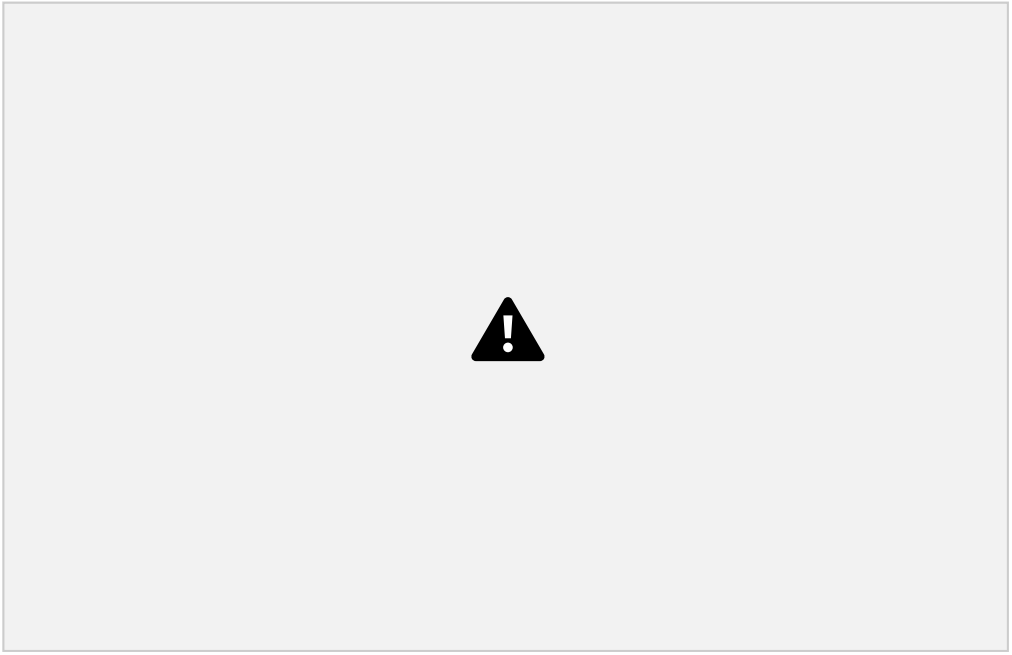

Рис. 2.9. Персонаж з активованим ригом у режимі Pose Mode у Blender [виконано автором]

Після додавання скелету, додається ригінг, який полягає у налаштуванні контролерів кісток, їх ієрархії та обмежень (constraints), що забезпечують зручне керування позами та рухами персонажа. Ригінг дозволяє не лише обертати чи переміщувати окремі кістки вручну, а й застосовувати до них складніші трансформації – наприклад, автоматичне згинання суглобів, обмеження кута повороту або зв’язування руху рук і пальців з допомогою інверсної кінематики (IK).

Рис.2.10. Рендер готової моделі персонажа [виконано автором] У режимі Pose Mode здійснюється перевірка правильності прив’язки та реакції моделі на маніпуляції зі скелетом.

Рис.2.11. Вікно анімації у Blender [виконано автором]

Завдяки повноцінному ригу створюється основа для подальшої анімації – як у Blender, так і після експорту до Unity.

2.3. Побудова інформаційної системи.

Інформаційна модель – модель об'єкта, представлена у вигляді інформації, що зображує істотні для даного розгляду параметри та змінні величини об'єкта, зв'язки між ними, входи і виходи об'єкта і дозволяє шляхом подачі на модель вхідних величин моделювати можливі стани об'єкта. [6]

Вона описує ключові об’єкти, події та взаємозв’язки, що виникають під час ігрової сесії. В основі моделі – причинно-наслідкові дії користувача та реакції системи на них.

Подана нижче блок-схема ілюструє типовий сценарій дії користувача у VR-грі “Low Poly Strike”. Вона відображає послідовність етапів, починаючи з моменту запуску гри, завершуючи взаємодією з ворожим об’єктом, що супроводжується анімацією “смерті” персонажа.

Рис.2.12. Блок-схема алгоритму взаємодії гравця зі зброєю [виконано автором]

Пояснення головних етапів алгоритму:

3. Натискання тригера – натискає на тригер контролера, що імітує дію спускового гачка. Це вхідна подія, яка ініціює подальшу програмну обробку.

4.

Виклик метода FireBullet() – метод FireBullet() активується, створюючи об'єкт кулі в заданій позиції (spawnPoint). Цей процес також задає швидкість кулі та її напрям.

Рис.2.13. Демонстрація вистрілу з пістолета. [виконано автором]

5. Створення кулі (Bullet) – у сцені з’являється новий об’єкт з компонентом

Rigidbody і Collider, що імітує фізичну модель кулі. Об’єкт рухається вперед відповідно до вектора напрямку зброї.

Рис.2.14. Вигляд мішені до пострілу [виконано автором]

6. Зіткнення з ворожим об’єктом – куля влучає у мішень – об’єкт TargetDummy. У результаті колізії викликається метод OnCollisionEnter(), що обробляє подальшу логіку.

Рис.2.15. Демонстрація вистрілу по мішені [виконано автором] 28

Рис.2.16. Анімація “смерті” мішені після пострілу [виконано автором] 7.

Анімація смерті – у відповідь на зіткнення активується анімаційний тригер Death, що змінює стан NPC: вмикається відповідна анімація, після чого об’єкт деактивується.

2.4. Побудова концептуальної моделі.

Для візуалізації архітектури VR-гри побудовано UML-діаграму класів, яка відображає логічні взаємозв’язки між основними компонентами системи. У діаграмі представлені чотири основні класи: HandsAnimation, Fire, TargetDummy та EnemyCharacterTrigger, що взаємодіють відповідно до сценаріїв ігрового процесу.

Рис.2.17. Інформаційна модель проєкту [виконано автором]

Клас HandsAnimation відповідає за відображення рухів руки користувача у віртуальному середовищі. Він зчитує значення стискання тригера та ручки (grip) з VR-контролерів за допомогою системи InputActionReference і передає їх в аніматор

руки:

```
float gripValue = gripReference.action.ReadValue<float>();
 handAnimator.SetFloat("Grip", gripValue);
```

Цей клас опосередковано впливає на логіку стрільби, оскільки дії з контролером можуть використовуватись для ініціації події пострілу в класі Fire. Клас Fire реалізує механізм стрільби. Його метод FireBullet() створює екземпляр кулі (bullet) у заданій позиції (spawnPoint) і надає їй початкову швидкість:

```
public void FireBullet()
{
 GameObject spawnedBullet = Instantiate(bullet, spawnPoint.position, spawnPoint.rotation);
spawnedBullet.GetComponent<Rigidbody>().velocity = spawnPoint.forward * bulletSpeed;
Destroy(spawnedBullet, 5f);
}
```

Куля, яка створюється, має Collider та Rigidbody, завдяки чому вона може зіштовхуватись з об’єктами, зокрема – з TargetDummy. У UML-діаграмі вказано зв’язок “1..1 до 1..*” між Fire та TargetDummy, що відображає можливість ураження кількох цілей.

Клас TargetDummy відповідає за поведінку мішеней, які гравець може уразити. Метод OnCollisionEnter() активує анімацію “смерті” або зникнення при влученні кулі:

```
private void OnCollisionEnter(Collision other)
{
 if(other.gameObject.CompareTag("Weapon") || other.gameObject.CompareTag("Bullet")) {
 dummyAnimator.SetTrigger("Death");
 GetComponent<BoxCollider>().enabled = false;
}
}
```

Крім цього, клас має метод ActivateDummy(),

```
public void ActivateDummy()
{
 dummyAnimator.SetTrigger("Activate");
}
```

що запускає анімацію активації цілі. Цей метод використовується тригером у класі EnemyCharacterTrigger.

Клас EnemyCharacterTrigger реалізує механізм активації цілей при вході гравця у визначену зону (тригер).


```
private void OnTriggerEnter(Collider other)
{
 if(other.gameObject.CompareTag("Player")){
 foreach(GameObject enemies in targetEnemy){
 enemies.GetComponent<TargetDummy>().ActivateDummy();
```

```
}
}
}
```

Він зберігає масив об’єктів (targetEnemy), до яких при вході гравця викликає метод ActivateDummy(). У UML-діаграмі зв’язок “1..* до 1..1” між EnemyCharacterTrigger та TargetDummy вказує на можливість активації одразу кількох NPC одним тригером.

2.5. Висновки до розділу 2.

Розділ 2 охоплює усі ключові аспекти підготовки для створення віртуального 3D персонажа до інтеграції в ігрове VR-середовище. Також на основі аналізу об’єктів системи було здійснено побудову інформаційної моделі, яка відображає логіку взаємодії гравця з віртуальним середовищем, а також UML-діаграму класів, що формалізує структуру застосованих скриптів і об’єктно-орієнтовану архітектуру проєкту.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Огляд та обґрунтування вибору інструментарію.

У процесі розробки VR-гри було використано наступні програмні засоби:

1. **Blender** – це повністю інтегрований комплект створення 3D, який пропонує

широкий діапазон основних засобів, включаючи моделювання – modeling, рендеринг – rendering, анімація та оснащення – animation & rigging, редагування відео – video editing, ефекти відео – VFX, компонування – compositing, текстурування – texturing та багато типів симуляцій – simulations [7]. Завдяки потужному набору інструментів та активній спільноті користувачів, його було обрано для створення та риггінгу 3D-моделей персонажа.

Як було попередньо розглянуто, риггінг є важливим елементом при розробці 3D моделей, адже забезпечує гнучкість та хюманізацію моделей для подальшого позиціонування або анімування.

Риггінг (rigging) – це процес створення скелетної структури (арматури) для 3D моделі, що дозволяє анімувати її, змінюючи положення та обертання кісток. *Основні елементи риггінгу:* [8]

• Арматури (armatures): дають змогу створювати скелет для об'єкта, що дозволяє деформувати сітку (mesh) і створювати гнучкі суглоби – зазвичай використовуються для анімації персонажів.

• Обмеження (constraints): дозволяють контролювати рухи, роблячи їх логічними та реалістичними. Наприклад, можна обмежити обертання суглоба лише в певному діапазоні.

• Модифікатори об’єкта (object modifiers): для складної деформації сітки можна застосовувати спеціальні модифікатори, які допомагають точно налаштувати поведінку об’єкта при русі.

• Ключові форми (shape keys): дають змогу створювати альтернативні форми сітки (наприклад, міміку обличчя) та плавно перемикатися між ними. • Драйвери (drivers): автоматизують керування параметрами. Наприклад, один контролер може змінювати одразу кілька властивостей, або параметри можуть змінюватися залежно від інших змін.

Процес риггінгу в Blender включає:

• Створення арматури: додавання кісток (bones) всередину 3D-моделі для визначення її рухомих частин.

• Позиціонування кісток: розміщення кісток відповідно до анатомії моделі,

наприклад, для персонажа – вздовж хребта, рук та ніг.

• Зв'язування кісток з моделлю (скіннінг): призначення вершин моделі до відповідних кісток, щоб вони рухалися разом.

• Налаштування обмежень (constraints): визначення правил руху кісток для реалістичної анімації, наприклад, обмеження кутів обертання суглобів. • Створення контролерів: додавання спеціальних об'єктів для спрощення керування складними рухами моделі.

Рис.3.1. Демонстрація риггінгу моделі у Blender для застосування у VR [виконано автором]

Рис.3.2. Демонстрація додавання осей до моделі у Blender для побудови позиціонування персонажа [виконано автором]

Функції риггінгу в Blender:

• Інверсна кінематика (IK): дозволяє керувати ланцюжками кісток, задаючи положення кінцевої кістки, а проміжні кістки автоматично обчислюють свої позиції. • Обмеження (constraints): надають можливість встановлювати правила для руху кісток, такі як обмеження обертання, переміщення або масштабування. • Контролери (custom shapes): створення користувацьких форм для кісток, що спрощує процес анімації, надаючи зручні інтерфейси для керування. • Автоматичне зважування (automatic weights): функція, що автоматично розподіляє вплив кісток на вершини моделі, спрощуючи процес скіннінгу. Отже, риггінг у Blender є ключовим етапом у процесі створення анімації, що дозволяє надавати 3D-моделям реалістичні рухи та взаємодії. Завдяки потужним інструментам, Blender надає гнучкість та ефективність у створенні складних анімаційних систем. Опанування цих інструментів відкриває широкі можливості для реалізації творчих задумів та забезпечує високу якість анімації.

2. **Unity** – це популярний багатоплатформовий рушій для розробки ігор, який підтримує створення VR-додатків. Його використано для інтеграції 3D-моделей, налаштування сцен та реалізації ігрової логіки. Unity забезпечує гнучкість та широкий спектр інструментів для VR-розробки. Для роботи з VR у Unity потрібно встановити додаткові плагіни такі як XR Plugin Management та XR Interaction Toolkit.

Рис.3.3. Вікно Package Manager у Unity для встановлення XR Plugin Management та XR Interaction Toolkit [виконано автором]

Говорячи про альтернативні ігрові рушії, варто згадати про Unreal Engine. Unreal Engine – це потужний рушій для створення 3D-графіки та інтерактивного контенту, розроблений компанією Epic Games у 1998 році та з часом став одним із провідних інструментів для розробки відеоігор, віртуальної реальності, а також застосовується в кіно, телебаченні, архітектурі та автомобільній промисловості.

У таблиці знизу наведено порівняльну характеристику обох рушіїв для розробки VR-проєктів.

Таблиця 3.1. Порівняльна характеристика Unity та Unreal Engine при розробці VR-проєктів

Фактор	**Unity**	**Unreal Engine**

Поріг входу	Низький – ідеально підходить для початківців	Високий – вимагає певних знань
Скриптування (логіка програми)	Мова C# (простіша для вивчення)	C++ у поєднанні з Blueprints – візуальне програмування (складніше для вивчення)
Цільові платформи	Кросплатформенна підтримка, включно з WebXR і Android (Meta Quest)	Кросплатформенна, але складніша оптимізація під мобільні VR
Мережеві VR-досвіди	Популярна у VR Chat, Population One, хороша інтеграція Photon/Netcode	Обмежені мультиплеєрні рішення, менша спільнота у VR
Маркетплейс та розширення	Великий Asset Store, багато безкоштовних інструментів	Вбудовані інструменти якісні, але менш різноманітні в Marketplace

Налаштування VR (XR Toolkit)	Простий доступ через Unity XR Interaction Toolkit	Більш складне підключення через OpenXR або SteamVR

Отже, для розробки дипломного проєкту було обрано ігровий рушій Unity, який є більш адаптованим до невеликих та нескладних проєктів. Він є більш доцільним, оскільки:

• Дозволяє швидке протипування об’єктів сцени.

• Краще оптимізований для невеликих сцен.

• Має простіший інструментарій для інтеграції VR (XR Toolkit). • Надає широкий доступ до багатьох інструментів, що полегшує розробку.

Зокрема, до таких інструментів належать XR Plugin Management та XR Interaction Toolkit.

XR Plugin Management – це пакет Unity, призначений для спрощення процесу керування XR-плагінами у проєкті. Він забезпечує єдиний інтерфейс для налаштування та ініціалізації різних XR-провайдерів, таких як OpenXR, ARCore, ARKit та інших. [1]

Основні функції:

• Ініціалізація та керування плагінами: автоматизує процес завантаження та налаштування XR-плагінів під час запуску застосунку.

• Уніфіковані налаштування: надає централізоване місце в Project Settings для керування XR-плагінами для різних платформ.

• Підтримка кількох платформ: дозволяє налаштовувати та використовувати різні XR-провайдери для різних цільових платформ у межах одного проєкту.

Переваги використання:

• Спрощення процесу розробки: зменшує складність інтеграції та налаштування XR-плагінів, дозволяючи розробникам зосередитися на створенні контенту.

• Гнучкість: можливість легко переключатися між різними XR провайдерами та платформами без значних змін у коді.

• Стабільність та сумісність: забезпечує узгодженість між версіями плагінів та Unity, зменшуючи ризик конфліктів та помилок.

XR Plugin Management працює на рівні налаштувань проєкту, забезпечуючи основу для роботи інших XR-компонентів, таких як XR Interaction Toolkit. Він гарантує, що необхідні плагіни правильно ініціалізовані та готові до використання в застосунку.

Пакет XR Interaction Toolkit – це високорівнева система взаємодії, що базується

на компонентах і призначена для створення VR- та AR-додатків. Вона забезпечує фреймворк, який дозволяє реалізовувати взаємодію з 3D-об'єктами та елементами інтерфейсу користувача за допомогою подій введення в Unity. Основу системи

становлять базові компоненти Interactor і Interactable, які об’єднуються за допомогою Interaction Manager. Крім цього, пакет включає елементи для реалізації переміщення користувача та візуального супроводу взаємодії. [10]

XR Interaction Toolkit містить набір компонентів, які підтримують такі завдання взаємодії: [10]

• Кросплатформенне введення з XR-контролерів: Meta Quest (Oculus), OpenXR, Windows Mixed Reality та інші.

• Базова взаємодія з об'єктами: наведення, вибір і захоплення. • Тактильний зворотний зв’язок через XR-контролери.

• Візуальний зворотний зв’язок (відтінки, лінії), що вказують на можливу або активну взаємодію.

• Базова взаємодія з UI-елементами (canvas) за допомогою XR-контролерів. • Утиліта для взаємодії з XR Origin – VR-камерою, яка керує стаціонарним або room-scale VR-досвідом.

Основні функції:

• Інтерактори та інтерактиви: моделі взаємодії, де інтерактори (наприклад, контролери) ініціюють взаємодію, а інтерактиви (об'єкти сцени) відповідають на неї. • Підтримка 3D та UI-взаємодії: дозволяє користувачам взаємодіяти як з тривимірними об'єктами, так і з елементами користувацького інтерфейсу. • Локомоція: містить компоненти для реалізації переміщення користувача в просторі, включаючи телепортацію та плавний рух.

• Гнучка система подій: надає події для різних станів взаємодії (наприклад, наведення, вибір), що дозволяє легко налаштовувати поведінку об'єктів.

Рис.3.4. Демонстрація інтерфейсу XR Interaction Toolkit при додаванні VR об'єктів у Unity [виконано автором]

Отже, XR Interaction Toolkit використовує інфраструктуру, налаштовану XR Plugin Management, для доступу до функціональності XR-пристроїв. Він працює поверх базових XR-систем, надаючи високорівневі абстракції для взаємодії та спрощуючи процес створення інтерактивних елементів.

Не менш важливим елементом у Unity є система анімацій. Unity має розвинену та складну систему анімації, відому також під назвою Mecanim. Вона є потужним інструментом для розробників ігор, який дозволяє створювати динамічні, реалістичні анімації для персонажів, об'єктів та середовищ у грі. Система анімації Unity дає змогу вдихнути життя в ігрових персонажів та об'єкти, анімуючи їхні рухи, дії та візуальні ефекти. [11]

Її широкі можливості охоплюють як базове створення та керування анімацією, так і складніші процеси – такі як змішування анімацій (blending), маскування (masking) та інверсна кінематика (inverse kinematics). [11]

Ключові компоненти системи анімацій Unity:

• Animation Clips (анімаційні кліпи) – це файли, які задають зміну властивостей об’єкта у часі — наприклад, положення, обертання чи масштаб. Анімаційні кліпи можна імпортувати з зовнішніх програм (Blender, Maya) або створити безпосередньо в Unity через вікно Animation. [11]

• Animator Controller (контролер анімацій) – це актив, який організовує кліпи у вигляді машини станів, визначаючи, як анімації переходять одна в одну. Переходи можуть запускатися через параметри – наприклад, вхід користувача або події в грі. Контролер має візуальний інтерфейс для управління анімаціями. [11]

Рис.3.5. Демонстрація вікна Animation Controller [виконано автором] • Blend Trees (дерева змішування) – дозволяють плавно змішувати декілька анімацій на основі параметрів (наприклад, швидкість – між ходьбою і бігом). • Animation Layers (анімаційні шари) – дають змогу одночасно відтворювати кілька анімацій на різних частинах тіла (наприклад, персонаж біжить і стріляє). • Inverse Kinematics (інверсна кінематика) – забезпечує реалістичне положення кінцівок шляхом автоматичного розрахунку суглобів для досягнення заданої точки (наприклад, постановка ноги на поверхню).

3. **Visual Studio Code (VS Code)** – це легкий, але водночас потужний редактор коду, який широко використовується для написання C#-скриптів у середовищі Unity. Завдяки підтримці численних розширень, таких як C# for Visual Studio Code (Omnisharp), IntelliCode та Debugger for Unity, VS Code забезпечує автодоповнення коду, зручне налагодження (debugging) та підсвітку синтаксису. Його

тісна інтеграція з Unity дозволяє швидко переходити від редагування скриптів до тестування проєкту. Для роботи та інтеграції з Unity, потрібно встановити розширення Unity.

Рис.3.6. Вікно розширення Unity у VS Code [виконано автором]

4. **Мова програмування C#** – це об'єктно-орієнтована мова програмування, створена компанією Microsoft для розробки застосунків на платформі .NET. Вона надає розробникам доступ до широкого набору бібліотек і інструментів .NET, що

дозволяє створення різноманітних застосунків, включаючи десктопні, веб, мобільні та ігрові програми. [12]

Коротко кажучи, C# – це сучасна, гнучка та ефективна мова програмування, яка стала незамінним інструментом у сфері розробки програмного забезпечення на платформі .NET.

Переваги C#:

• Об'єктно-орієнтоване програмування: C# із самого початку був створений за принципами об'єктно-орієнтованого програмування (ООП). Цей підхід передбачає організацію даних у вигляді об'єктів, що дозволяє легше розділяти програму на частини, які зручно розробляти та підтримувати. [13]

• Проста крива навчання: це мова високого рівня з легкою кривою навчання, що робить її доступною для програмістів з будь-яким рівнем підготовки. [12]

• Кросплатформеність: завдяки .NET Core та .NET 5/6, C# дозволяє створювати застосунки, що працюють на різних операційних системах, таких як Windows, macOS та Linux. До них входять:

- Консольні застосунки
- Настільні застосунки (Windows Forms, WPF)
- Служби Windows
- Вебсервіси та вебзастосунки (ASP.NET Core, Blazor)
- Нативні десктопні та мобільні застосунки (.NET MAUI)
- Застосунки на базі ШІ (ML, Microsoft.Extensions.AI)
- Розподілені та хмарні застосунки (Azure)
- Базовані на БД застосунки (Entity Framework)
- Ігри (Unity, Godot)
- IoT-застосунки (.NET NanoFramework, Wildernesslabs)
- Багаторазово використовувані бібліотеки

• Широке застосування: C# використовується для розробки різних типів застосунків, включаючи ігри з використанням рушія Unity, вебсервіси та мобільні додатки.

• Продуктивність: C# забезпечує високу продуктивність завдяки потужним

інструментам розробки, таким як IntelliSense (автодоповнення коду) та розширене debugging. Це значно полегшує написання та тестування коду, зменшує кількість помилок і прискорює розробку. Крім того, сучасні середовища розробки, як-от Visual Studio, надають інтеграцію з системами контролю версій, шаблонами проектів і візуальними дизайнерами інтерфейсів, що ще більше підвищує ефективність програміста. [12]

Функції C#:

• Делегати та події: механізми, що дозволяють реалізовувати патерни проєктування, такі як спостерігач, та створювати гнучкі системи обробки подій. • LINQ (Language Integrated Query): інтегрований у мову засіб для запитів до колекцій, баз даних та XML, що спрощує обробку даних.

• Асинхронне програмування: підтримка асинхронних методів через ключові слова async та await дозволяє створювати високопродуктивні застосунки. • Розширювальні методи: можливість додавати нові методи до існуючих типів без необхідності змінювати їхній вихідний код.

Отже, C# – це потужна, сучасна та багатофункціональна мова програмування, яка забезпечує гнучкість, високу продуктивність і широку підтримку інструментів та платформ. Завдяки своїм перевагам, таким як об'єктно-орієнтованість, простота у вивченні, кросплатформеність і потужні можливості розробки.

3.2. Опис апаратних засобів.

Для розробки проєкту було використано Apple MacBook Pro M4. Технічні характеристики:

1. Процесор (CPU): 14-ядерний чип Apple M4 Pro, що складається з 10 продуктивних та 4 енергоефективних ядер, забезпечуючи високу продуктивність для вимогливих завдань.

2. Оперативна пам'ять (RAM): 24 ГБ уніфікованої пам'яті.

3. Графічний процесор (GPU): 20-ядерний GPU, інтегрований у чип M4 Pro.

4. Накопичувач: SSD 512 GB.

Ця конфігурація забезпечила комфортну роботу з Blender та Unity, а також

ефективне тестування VR-додатку.

3.3. Вимоги до програмного забезпечення.

• Blender: версія 2.8 або новіша;

• Unity: версія 2021.3 LTS або новіша;

• Visual Studio Code: остання стабільна версія;

• XR Plugin Management та XR Interaction Toolkit: для підтримки VR функціональності в Unity.

3.4. Опис розробленого програмного забезпечення.

3.4.1. Основний функціонал гри.

Розроблене програмне забезпечення є прототипом VR-мінішутера, створеним з метою демонстрації базових принципів побудови інтерактивного VR-середовища. Додаток реалізує ключові механіки, характерні для VR-ігор, що робить його цінним

як для практичного ознайомлення з інструментами розробки, так і як базу для подальших ігрових або навчальних VR-проєктів.

Зокрема, основний функціонал включає:

• Переміщення гравця по карті: реалізовано плавне пересування гравця за допомогою контролерів VR-шолома та протестовано за допомогою XR Device Simulator.

• Взаємодія з об'єктами: реалізована можливість підбору предметів, використання зброї, активації ігрових механізмів (наприклад, кнопок), що дозволяє гравцю взаємодіяти з віртуальним світом природним способом.

• Механіка стрільби: реалізовано механіку стрільби та ударів з різних видів зброї з відповідною анімацією та ліквідацією ворогів.

• Вороги: створено простий штучний інтелект для ворогів, які реагують на дії гравця та при ліквідації можуть відроджуватись.

3.4.2. Завдання, що реалізує гра.

• Навчання основам VR-взаємодії: гра є навчальним прикладом, у якому продемонстровано основні механіки, типові для VR-додатків – такі як пересування, захоплення об’єктів, натискання кнопок, стрільба тощо. Для людей, що раніше не були дотичні до VR-ігор це може стати новим, простим, та цікавим досвідом, що

допоможе краще розібратись, як працює взаємодія з XR-системами в Unity.

• Дослідження середовища: рівень створено таким чином, щоб користувач міг вільно пересуватись, ознайомлюючись з ігровим простором. Додано прості елементи для імітації тиру та навколишнього середовища.

Підсумовуючи, формалізація функціональних та технічних вимог на початковому етапі проєктування VR-додатку є важливою передумовою його успішної реалізації відповідно до поставлених цілей. Завдяки цьому забезпечується структуроване розуміння процесу створення інтерактивного віртуального середовища, що включає етапи моделювання, анімації, програмної інтеграції та налаштування системи взаємодії. Результатом є програмний продукт, який може бути використаний не лише з навчальною метою, але й як основа для розширення функціоналу чи впровадження у більш складні VR-проєкти.

3.5. Висновки до розділу 3.

У розділі 3 було обґрунтовано вибір інструментів програмної та технічної реалізації VR-проєкту, що базується на використанні сучасних технологій моделювання, розробки ігрових сцен та програмування. Під час реалізації проєкту було застосовано комплексні засоби розробки такі як Blender, Unity та мова програмування C#.

Для забезпечення функціональної підтримки інструментів віртуальної реальності та спрощення реалізації механік взаємодії в ігровому середовищі, було використано плагіни XR Plugin Management та XR Interaction Toolkit.

Середовищем розробки було обрано Visual Studio Code як гнучке та інтегроване середовище для налагодження та редагування коду.

Використані в дипломному проєкті технічне та програмне забезпечення відповідає цілям та завданням розробки проєкту та забезпечило створення прототипу VR-гри з базовими взаємодіями.

ВИСНОВКИ

У ході розробки дипломного проєкту було розроблено прототип VR-гри з інтерактивними персонажами, що реалізує основні принципи взаємодії користувача у віртуальному середовищі.

У ході реалізації було виконано всі ключові етапи життєвого циклу створення VR-програми – від побудови логіки взаємодії, моделювання та ригінгу персонажа до інтеграції анімацій, реалізації стрільби, реакцій NPC на події та тестування сцени в середовищі віртуальної реальності.

У рамках проєкту були виконані наступні ключові завдання:

• Створено 3D-модель персонажа та виконано риггінг і анімацію за допомогою Blender.

• Налаштовано VR-середовище у Unity та реалізовано базовий функціонал гри, включаючи переміщення, стрільбу, взаємодію з об’єктами та ворогами. • Інтегровано технології XR Plugin Management та XR Interaction Toolkit для забезпечення підтримки VR-контролерів та симуляції взаємодії. • Розроблено структуру сцени, імпортовано ассети, налаштовано механіки геймплею та протестовано логіку гри за допомогою XR Device Simulator. • Налагоджено середовище розробки на базі Unity та Visual Studio Code, здійснено написання скриптів мовою C#.

• Проведено тестування проєкту та оптимізацію роботи прототипу. Результатом виконаної роботи став навчальний VR-прототип, що може бути використаний як основа для подальшої розробки повноцінної VR-гри або для демонстрації принципів побудови взаємодії у віртуальному середовищі. Розроблений проєкт має практичну цінність і може бути застосований у навчальному процесі, а також для створення власних ігрових чи симуляційних VR-додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Steam VR Game Releases by Year. *SteamDB*. [Електронний ресурс] – Режим доступу: https://steamdb.info/stats/releases/?tagid=21978

2. Dan Primack. AR/VR gaming engine Unity raises $400 million. *Axios*. [Електронний ресурс] – Режим доступу: https://www.axios.com/2017/12/15/arvr

gaming-engine-unity-raises-400-million-1513302547

3. Kyle Orland. Despite $100 price increase, Meta Quest 2 still offers historically cheap VR. *ArsTechnica, 2022.* [Електронний ресурс] – Режим доступу: https://arstechnica.com/gaming/2022/07/despite-100-price-increase-meta-quest-2-still offers-historically-cheap-vr/

4. Virtual Reality In Gaming Market Forecast and Insights: Exploring Growth Trends, Market Size, and Emerging Opportunities - Latest Global Market Insights. *Latest Global Market Insights - By The Business Research Company.* [Електронний ресурс] – Режим доступу: https://blog.tbrc.info/2025/04/virtual-reality-in-gaming-market-outlook 3/

5. Wurmser Y. US Virtual Reality Forecast 2023. *EMARKETER.* [Електронний ресурс] – Режим доступу: https://www.emarketer.com/content/us-virtual-reality-forecast 2023

6. Інформаційна модель. Вікіпедія. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C

7. Вступ - Introduction – Blender Manual. Blender Documentation. *blender.org.* [Електронний ресурс] - Доступний за адресою: https://docs.blender.org/manual/uk/2.82/getting_started/about/introduction.html

8. Introduction – Blender Manual. Blender Documentation. *blender.org.* [Електронний ресурс] - Доступний за адресою: https://docs.blender.org/manual/es/3.4/animation/introduction.html#rigging 9. About the XR Plug-in Management package | XR Plugin Management | 4.5.1. Unity - Manual: Unity 6 User Manual. *Unity3d.com* [Електронний ресурс] - Доступний за адресою: https://docs.unity3d.com/Packages/com.unity.xr.management@4.5/manual/index.html 10. XR Interaction Toolkit | XR Interaction Toolkit | 3.0.7. Unity - Manual: Unity 6 User Manual. *Unity3d.com* [Електронний ресурс] - Доступний за адресою: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html

11. Introduction to Unity's Animation System. *LogicSimplified.com.* [Електронний ресурс] - Доступний за адресою: https://logicsimplified.com/newgames/introduction-to-unitys-animation-system/ 12.

Introduction to C#: definition and utilities. *Mytaskpanel.* [Електронний ресурс] - Доступний за адресою: https://www.mytaskpanel.com/introduction-to-csharp/ 13. What Is C#: Definitions, Strengths & Usages. *Designveloper.* [Електронний ресурс] - Доступний за адресою: https://www.designveloper.com/blog/what-is-c-sharp/ 14. McMahan R., Bowman D., Cobb S. Virtual Reality Games: Extending Unity Learn Games to VR. *Cornell University,* 2024.

15. Blain J. M. The Complete Guide to Blender Graphics. A K Peters//*CRC Press*, 2019

16. Gabajová G., Zolotová I. Designing Virtual Workplace Using Unity 3D Game Engine *// Acta Electrotechnica et Informatica*. – 2021. – Vol. 21, № 2. – C. 12–18. 17. Murray J. W. C# Game Programming Cookbook for Unity 3D. *Taylor & Francis Group*, 2021. 298 с.

18. Lövy, J. C# for Game Development / J. Lövy. – New York : *Apress*, 2021. – 350 с.

19. Kriger, M. Rigging Characters for Blender / M. Kriger. – New York : *Apress*, 2021. – 321 с.

ДОДАТКИ

Додаток А

Процес створення ворожого персонажа у Blender

Створення файлу enemy_character.blend

Option+G (Alt+G) для центрування персонажа відносно 3D-курсору.

Створення сітки для додавання одягу персонажа.

Робота з “сіткою” для одягу (верхній одяг).

За допомогою команд Shift+D копіюємо виділений слой. Та за допомогою команди P відокремлюємо область виділення та робимо його як окремий слой.

Надаємо об’єму за допомогою масштабування Option+S+Shift.

51

Створення рукавів:

Закриття рукавів за допомогою F+I (insert)

52

Далі зміна положення об’єкта за допомогою команди G+Z. Створення додаткових границь командою Command+B (Ctrl+B).

Надаємо об’єму за допомогою команди S (Scale).

Загальний вигляд персонажа після змін.

54

Закриваємо проміжки між тілом та одягом персонажа. За допомогою команди E (Extrude).

55

Створення паска (ремінь):

Обираємо зону, з якої буде створено ремінь за допомогою команди Option+Shift+LMB. Далі Shift+D для копіювання виділення та комадна P для відокремлення елементу від основної фігури.

Далі за допомогою команди Option+E (Extrude) обираємо “Extrude Among Normals”.

56

Готовий ремінь.
Створення нижнього одягу (штани) + взуття:

Виділяємо необхідну поверхню.

57

Далі відокремлюємо штани від основної моделі за допомогою команди P.

Та за допомогою масштабування S (Scale), переміщення об’єктів G, створюємо штани довільної форми.

Створюємо взуття за таким ж принципом.

58

Shift+D, P відокремлюємо слой від основного об’єкта.
Масштабуємо за допомогою команди S (Scale) та закриваємо порожні місця командою F. Далі за допомогою команди Command+R створюємо границі на взутті для подальшого модифікування взуття персонажа.

59

Створення волосся:

Обираємо зону для подальшого моделювання волосся. Shift+D, P для відокремлення волосся від основного об’єкта.

60

Далі за допомогою команд E (Extrude), S (Scale), формуємо волосся персонажа.
Option+E -> “Extrude faces along normals”
Додаємо об’єм та проробляємо форму для волосся.

61

62

Додавання скелету персонажа:

Перед тим як додавати скелет до персонажа, потрібно перевірити нормалі, аби знати чи об’єкт підходить для цього.

Команда Shift+N для розрахунку нормалей.

Оскільки всі нормалі виходять на зовні, значить все гаразд.

Далі перевірка на double geometry -> Команда M

Все впорядковано.

Додаємо людський скелет:

64

За допомогою масштабування S (Scale) підводимо скелет до плечей персонажа.
Здійснюємо ре позиціонування рук персонажа згідно скелету. За допомогою команди R (Rotate).

65

Доданий скелет припасовуємо до тіла персонажа та симетрично дублюємо ліву сторону моделі. За допомогою команди Armature -> Symmetrize.

66

Додавання осей для побудування пози персонажа.

67

Створення меча.

68

Додаємоматеріалитавстановлюємопозуперсонажа
Рендер моделі.

Додаток Б

Проміжні етапи створення VR-проєкту

Менеджер файлів проєкту в Unity.
Імпорт рук:

Створення анімації стискання рук та захоплення об’єктів

було створено окремий Animator Controller, у якому анімації керуються через змінні типу Float. У вікні Animator було створено два параметри: • Grip (Float) –

відповідає за стискання кисті,

• Trigger (Float) – відповідає за натискання вказівного пальця.

70

Створення скрипта:

У вікні анімації створюємо дві змінні Float, які зберігатимуть значення для Grip та Trigger.

Для плавної зміни між різними положеннями руки (від відкритої до повністю стиснутої) було використано Blend Tree.

71

Елемент 1 – базова анімація руки у спокійному стані (рука

відкрита).

Елемент 2 – часткове стискання (анімація щипців): великий палець торкається вказівного.

Елемент 3 – повне захоплення (стискання кулака).

Blend Tree дозволяє плавно інтерполювати між цими станами залежно від значення Grip і Trigger, які змінюються в діапазоні від 0 до 1.

Готовий варіант анімацій.

Результат роботи:

Реалізація переміщення гравця у VR-сцені (XR Origin) Додавання компонента Locomotion System.
Налаштування безперервного переміщення (Continuous Move Provider).

Обрано лівий контролер для пересування.

Налаштування
обертання (Continuous Turn Provider або Snap Turn Provider)
74

Додавання колайдерів до XR Origin.

Щоб забезпечити фізичну взаємодію з навколишнім середовищем, до об’єкта XR Origin необхідно додати компонент Character Controller або Capsule Collider, який визначає зіткнення між тілом гравця та іншими об’єктами сцени. Створене середовище для гри.

75

Додаток В

Програмний код

HandsAnimation.cs

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

public class HandsAnimation : MonoBehaviour
{
 // Reading values from controllers
 [SerializeField] private InputActionReference gripReference;
 [SerializeField] private InputActionReference triggerReference;
 [SerializeField] private Animator handAnimator;

 // Update is called once per frame
 void Update()
 {
 float gripValue = gripReference.action.ReadValue<float>();
 handAnimator.SetFloat("Grip", gripValue);

 float triggerValue = triggerReference.action.ReadValue<float>();
 handAnimator.SetFloat("Trigger", triggerValue);
 }
}
```

Fire.cs

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

public class Fire : MonoBehaviour
{
 [SerializeField] private GameObject bullet;
 [SerializeField] private Transform spawnPoint;

 [SerializeField] private float bulletSpeed = 10f;


 public void FireBullet()
 {
 GameObject spawnedBullet = Instantiate(bullet, spawnPoint.position, spawnPoint.rotation);
spawnedBullet.GetComponent<Rigidbody>().velocity = spawnPoint.forward * bulletSpeed;
Destroy(spawnedBullet, 5f);
 }
}
```

TargetDummy.cs

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



```
public class TargetDummy : MonoBehaviour
{
 [SerializeField] private Animator dummyAnimator;
 private void OnCollisionEnter(Collision other)
 {
 if(other.gameObject.CompareTag("Weapon") || other.gameObject.CompareTag("Bullet")) {
```

```
dummyAnimator.SetTrigger("Death");
GetComponent<BoxCollider>().enabled = false;
}
}

public void ActivateDummy()
{
dummyAnimator.SetTrigger("Activate");
}
        }
```

EnemyCharacterTrigger.cs

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

public class EnemyCharacterTrigger : MonoBehaviour
{
[SerializeField] private GameObject[] targetEnemy;

private void OnTriggerEnter(Collider other)
{
if(other.gameObject.CompareTag("Player")){
foreach(GameObject enemies in targetEnemy){
enemies.GetComponent<TargetDummy>().ActivateDummy();
}
}
}
}
```

FixedXRGrabInteractable.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

public class FixedXRGrabInteractable : XRGrabInteractable
{
[SerializeField] private Transform leftHandAttachTransform;

[SerializeField] private Transform rightHandAttachTransform;

protected override void OnSelectEntered(SelectEnterEventArgs args)
{
if(args.interactableObject.transform.CompareTag("LeftHand")){
attachTransform = leftHandAttachTransform;
}
if(args.interactableObject.transform.CompareTag("RightHand")){

attachTransform = rightHandAttachTransform; }

base.OnSelectEntered(args);
}
}
```