

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО

В.о. завідувача кафедри

_____ Олександр КОВАЛЬ

« ____ » _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення**

**на тему: «Програмний модуль адаптації складності реабілітаційних вправ на
основі прогресу пацієнта»**

Виконала:

студентка IV курсу, групи ТВ-23

Корж Інна В'ячеславівна

(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник:

Старший викладач, Дацюк О.А.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Рецензент:

_____ (посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.

Студентка _____

(підпис)

Київ – 2026

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Олександр КОВАЛЬ

«__» _____ 2026р.

ЗАВДАННЯ

на дипломну роботу студенту

_____ Корж Інні В'ячеславівні

(прізвище, ім'я, по батькові)

1. Тема роботи: Програмний модуль адаптації складності реабілітаційних вправ на
основі прогресу пацієнта

керівник роботи Дацюк О.А., старший викладач

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

затверджені наказом по університету від «__» _____ 2026р. № _____

2. Строк подання студентом роботи «__» _____ 2026р.

3. Вихідні дані до роботи: VR-застосунок, мова програмування C#, ігровий рушій
Unity, платформа .NET, ORM Entity Framework Core, СКБД PostgreSQL,
середовище розробки Visual Studio,

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно
розробити): проаналізувати предметну область, визначити вимоги системи,
провести дослідження аналогічних програм, спроектувати архітектуру, описати
засоби розробки VR застосунка та серверної частини, розробити програмний
продукт, провести тестування і налагодження.

5. Перелік ілюстративного матеріалу: схема бази даних, схеми алгоритму адаптації
складності вправ, інтерфейс користувача, зображення аналогів системи

6. Дата видачі завдання «31» жовтня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	31.10.2025	Виконано
2	Дослідження предметної області	01.11.2025 – 31.11.2025	Виконано
3	Дослідження існуючих рішень	01.12.2025 – 31.12.2025	Виконано
4	Постановка вимог до проектування системи	01.01.2026 – 31.01.2026	Виконано
5	Розробка програмного продукту	01.02.2026 – 03.05.2026	Виконано
6	Тестування	04.05.2026 – 10.05.2026	Виконано
7	Захист програмного продукту	11.05.2026 – 15.05.2026	Виконано
8	Оформлення дипломної роботи	18.05.2026 – 31.05.2026	Виконано
9	Передзахист	01.06.2026 – 05.06.2026	Виконано
10	Захист	15.06.2026 – 19.06.2026	Виконано

Студент

(підпис)

Інна КОРЖ

(ім'я, прізвище)

Керівник роботи

(підпис)

Оксана ДАЦЮК

(ім'я, прізвище)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 56 сторінок, 22 рисунків, 1 таблицю, 2 додатки та 13 посилань.

Метою роботи є розробка програмного модуля, який забезпечує відстеження рухів пацієнта за допомогою VR-контролерів та додаткових сенсорів, а також автоматичну адаптацію складності реабілітаційних вправ відповідно до індивідуального прогресу користувача. Розроблене програмне забезпечення повинно містити алгоритм адаптивного підбору вправ, що дозволяє персоналізувати процес реабілітації та забезпечує поступове, безпечне й ефективне збільшення фізичного навантаження залежно від стану пацієнта.

Практичне значення одержаних результатів полягає у застосуванні розроблених алгоритмів адаптації складності вправ та відстеження рухів користувача для створення програмного продукту, який може використовуватися в реабілітаційних цілях.

У межах роботи створено програмний застосунок, що забезпечує можливість взаємодії з користувачами у VR-середовищі та може бути адаптований для використання з реальними пацієнтами в процесі відновлювальної терапії. Розроблене рішення сприяє персоналізації реабілітаційного процесу, підвищенню його ефективності та об'єктивності оцінювання прогресу користувача.

Ключові слова: VR-застосунок, Unity, реабілітація, C#, анімація, PostgreSQL, алгоритми адаптації і відстеження рухів

ABSTRACT

Structure and scope of the thesis. The work contains 56 pages, 22 figures, 1 table, 2 appendices and 13 references.

The purpose of the work is to develop a software module that tracks the patient's movements using VR controllers and additional sensors, as well as automatically adapts the difficulty of rehabilitation exercises according to the user's individual progress. The developed software should include an algorithm for adaptive exercise selection, which allows for the personalization of the rehabilitation process and ensures a gradual, safe, and effective increase in physical load depending on the patient's condition.

The practical value of the results lies in the application of the developed algorithms for adapting exercise difficulty and tracking user movements to create a software product that can be used for rehabilitation purposes.

As part of this work, a software application was created that enables interaction with users in a VR environment and can be adapted for use with real patients during rehabilitation therapy. The developed solution facilitates the personalization of the rehabilitation process, increases its effectiveness, and ensures objective assessment of the user's progress.

Keywords: VR application, Unity, rehabilitation, C#, animation, PostgreSQL, motion adaptation and tracking algorithms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	8
ВСТУП.....	9
1 ПОСТАНОВКА ЗАВДАННЯ АДАПТАЦІЇ СКЛАДНОСТІ РЕАБІЛІТАЦІЙНИХ ВПРАВ НА ОСНОВІ ПРОГРЕСУ ПАЦІЄНТА.....	11
1.1 Функціональні вимоги системи	12
1.2 Нефункціональні вимоги системи	13
Висновки до розділу 1	14
2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ.....	15
2.1 Аналіз існуючих рішень.....	15
2.2 Адаптація складності вправ.....	22
Висновки до розділу 2	25
3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	26
3.1 Середовище розробки Unity	26
3.2 Система керування базами даних PostgreSQL.....	27
3.3 Мова C#.....	28
Висновки до розділу 3	29
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ.....	30
4.1 Архітектура додатку.....	30
4.2 Архітектура бази даних.....	31
4.3 Реалізація серверної частини.....	34
4.4 Анімація.....	35
4.5 Створення UI елементів	39
4.6 Алгоритми	42
4.7 Тестування.....	47
4.8 Впровадження	48
Висновки до розділу 4	49
5 РОБОТА КОРИСТУВАЧА ІЗ ПРОГРАМНОЮ СИСТЕМОЮ	50
5.1 Віртуальні середовища системи.....	50

5.2 Користувачський інтерфейс системи	53
Висновки до розділу 5	55
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТОК А Лістинг розробленої системи	59
ДОДАТОК Б Презентація	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	База даних
СКБД	Система керування базами даних
ACID	Atomicity, Consistency, Isolation, Durability (Атомарність, Узгодженість, Ізольованість, Довговічність)
AR	Augmented Reality (Додаткова реальність)
DDA	Dynamic Difficulty Adjustment (Динамічна адаптація складності)
EMG	Electromyography (Електроміографія)
HTTP	HyperText Transfer Protocol (Протокол передачі гіпертексту)
LAN	Local Area Network (Локальна комп'ютерна мережа)
ML	Machine Learning (Машинне навчання)
NoSQL	Not Only Structured Query Language (Не лише (структурована мова запитів)
OOP	Object-Oriented Programming (Об'єктно-орієнтоване програмування)
ORM	Object-Relational Mapping (Об'єктно-реляційне відображення)
PC	Personal Computer (Персональний комп'ютер)
REST API	Representational State Transfer Application Programming Interface (Програмний інтерфейс передачі стану представлення)
SQL	Structured Query Language (Структурована мова запитів)
UI	User Interface (Користувацький інтерфейс)
UX	User Experience (Користувацький досвід)
VR	Virtual Reality (Віртуальна реальність)
XR	Extended Reality (Розширена реальність)

ВСТУП

У ході виконання роботи було здійснено детальний аналіз предметної області. Стрімкий розвиток технологій сприяє трансформації різних сфер життя людини та вдосконаленню існуючих рішень з метою підвищення їхньої ефективності. Особливо помітні ці зміни у сфері медицини, де інноваційні технології відкривають нові можливості для діагностики, лікування та реабілітації пацієнтів. Поява технологій віртуальної реальності (VR) змінила не лише формат дозвілля, але й стала основою для впровадження сучасних реабілітаційних методик. VR все частіше використовується в різних медичних галузях, включаючи психотерапію, неврологічну та фізіологічну реабілітацію. Крім того, VR застосовується в освітніх цілях для моделювання хірургічних процедур в ортопедії та неврології. Останні дослідження показали, що системи на основі VR можуть бути оптимальним елементом підтримки для підвищення мотивації пацієнтів та їх залучення до призначених реабілітаційних програм.

В умовах воєнного конфлікту, коли зростає кількість військових та цивільних осіб, які потребують медичної допомоги, особливої актуальності набувають дистанційні та гнучкі формати реабілітації. Завдяки своїй природі VR-технології забезпечують можливість продовження реабілітаційного процесу поза межами медичних закладів. Це сприяє оптимізації використання медичних ресурсів, підвищенню пропускну здатності системи охорони здоров'я, навантаження на медичні заклади та розвитку горизонтальної взаємодії між фахівцями й пацієнтами.

Однією з ключових переваг є можливість продовження реабілітаційного процесу поза межами медичних закладів. Це сприяє оптимізації використання медичних ресурсів, підвищенню пропускну здатності системи охорони здоров'я, навантаження на медичні заклади та розвитку горизонтальної взаємодії між фахівцями й пацієнтами.

Іншою важливою перевагою є автономність використання VR-технологій, що мінімізує залежність від зовнішніх умов. Пацієнт може продовжувати

реабілітаційний процес самостійно, без постійної присутності медичного персоналу, у будь-якому приміщенні, обладнаному необхідними технічними засобами. Застосування VR дозволяє уникнути витрат часу на дорогу до реабілітаційного центру, очікування в черзі чи проходження адміністративних процедур. Крім того, пацієнт отримує можливість самостійно планувати графік занять відповідно до власного режиму та самопочуття.

Додатковою перевагою є можливість динамічного налаштування параметрів складності, інтенсивності та типу вправ у режимі реального часу. VR застосунки дозволяють змінювати формат і спрямованість вправ залежно від індивідуальних потреб, стану та цілей пацієнта. Реабілітаційний процес може бути реалізований як у формі ігрових сценаріїв із підвищеним рівнем залучення, так і в більш структурованому, клінічно орієнтованому форматі.

1 ПОСТАНОВКА ЗАВДАННЯ АДАПТАЦІЇ СКЛАДНОСТІ РЕАБІЛІТАЦІЙНИХ ВПРАВ НА ОСНОВІ ПРОГРЕСУ ПАЦІЄНТА

Метою роботи є створення програмного модуля, який забезпечує відстеження рухів пацієнта за допомогою VR-контролерів та додаткових сенсорів та автоматично адаптує складність реабілітаційних вправ відповідно до індивідуального прогресу. Програмне забезпечення повинно мати алгоритм адаптивного підбору вправ, який дозволяє персоналізувати реабілітацію та забезпечує безпечне та ефективне підвищення фізичних навантажень для пацієнта.

Основні завдання, які необхідно вирішити для досягнення поставленої мети, полягають у наступному:

- 1) зібрати інформацію про аналогічні додатки та проаналізувати їх переваги та недоліки;
- 2) розробити алгоритм відстеження рухів користувача з використанням VR-контролерів та додаткових сенсорів;
- 3) розробити алгоритм аналізу прогресу пацієнта на основі попередніх результатів і метрик;
- 4) створити програмний модуль адаптації складності вправ відповідно до попереднього аналізу прогресу;
- 5) реалізувати механізм автоматичного коригування параметрів вправ (інтенсивність, кількість повторень, складність рухів).

Вхідні дані для застосунку:

- 1) дані з контролерів і сенсорів (швидкість і амплітуда рухів, координати положення);
- 2) обраний режим (кількість необхідних повторень, час виконання, сет необхідних вправ);
- 3) історія прогресу пацієнта.

Крім того в додатку також будуть присутні проміжні дані, такі як оцінка якості виконання вправ. Якщо користувач неправильно виконує вправу, то система може її не зарахувати і попросити переробити чи змінити техніку. Від проміжних даних будуть безпосередньо залежати вихідні дані.

Вихідними даними буде слугувати: рівень складності вправ який буде застосований наступного разу – це може бути як простіший так і складніший рівень; оновлені параметри вправ і можливо навіть рекомендації для пацієнта.

1.1 Функціональні вимоги системи

Розроблювана система є VR-додатком для реабілітації нижніх кінцівок, тому її функціональність орієнтована на виконання реабілітаційних вправ, аналіз рухів користувача та надання зворотного зв'язку в режимі реального часу.

Пацієнт повинен мати можливість обирати та запускати реабілітаційні вправи у VR-середовищі. Для реалізації цієї функціональності необхідно передбачити різні віртуальні середовища (environments), а також інтерфейс взаємодії користувача, включаючи модальні вікна для вибору та підтвердження дій.

У системі має бути реалізований алгоритм визначення правильності виконання вправ на основі аналізу кутів суглобів та фаз руху користувача. Основними показниками оцінювання є точність, плавність та завершеність виконання рухів, які формують основу механізму аналізу якості вправ.

Крім того, на серверній частині (backend) необхідно реалізувати механізм формування інтегральної оцінки виконання вправи. До цієї оцінки враховуються показники емоційного стану пацієнта, результати поточної сесії, а також історія попередніх тренувань. Це дозволяє забезпечити більш адаптивний підхід до оцінювання прогресу користувача.

Додатково система повинна містити безпекові обмеження, які гарантують, що при досягненні критичних значень хоча б одного з параметрів оцінювання

автоматичне підвищення складності не відбувається. Це забезпечує стабільність роботи алгоритму адаптації та запобігає надмірному навантаженню на пацієнта.

Після прийняття рішення системою повинна здійснюватися адаптація складності вправ залежно від результатів користувача. У разі підвищення або зниження рівня складності відповідним чином змінюються параметри допустимих діапазонів кутів згинання та ротації суглобів, а також значення порогових величин (threshold), що визначають критерії зарахування рухів.

Також у системі необхідно реалізувати механізм візуального зворотного зв'язку під час виконання вправ. Це дозволяє користувачу в режимі реального часу отримувати інформацію про коректність виконання рухів, зокрема розуміти, чи був рух зарахований, виконаний правильно або потребує корекції.

Після завершення тренувальної сесії результати виконання вправ повинні автоматично зберігатися у базі даних. Надалі ці дані використовуються для аналізу прогресу користувача та формування наступних рішень щодо адаптації складності, що базуються на історії попередніх тренувань пацієнта.

1.2 Нефункціональні вимоги системи

Для розроблюваного VR-додатка нефункціональні вимоги є особливо важливими, оскільки система повинна забезпечувати стабільну роботу у режимі реального часу, комфортну взаємодію користувача з VR-середовищем та можливість подальшого масштабування.

Система повинна забезпечувати стабільну роботу у VR-середовищі без помітних затримок та просідання FPS, оскільки низька продуктивність може негативно впливати на комфорт користувача та викликати VR-дискомфорт. Особливо важливо забезпечити плавність відображення рухів і своєчасне оновлення даних під час виконання вправ у режимі реального часу

Однією з ключових нефункціональних вимог системи є швидкодія, оскільки за задумом додаток повинен оперативно формувати рішення щодо адаптації

складності вправ. Алгоритми аналізу рухів та обробки результатів мають працювати з мінімальною затримкою, забезпечуючи швидкий зворотний зв'язок і комфортну взаємодію користувача із системою у режимі реального часу.

Крім того, система повинна бути масштабованою, що забезпечить можливість подальшого розширення функціональності без необхідності суттєвої зміни архітектури додатка. Зокрема, має бути передбачена можливість легкого додавання нових вправ, ігрових середовищ та додаткових модулів системи.

Інтерфейс користувача повинен відповідати особливостям VR-середовища та бути максимально простим і мінімалістичним. Усі елементи інтерфейсу, зокрема кнопки, текстові написи та слайдери, повинні мати достатній розмір для зручної взаємодії за допомогою VR-контролерів. Такий підхід дозволяє покращити зручність використання системи та зменшити навантаження на користувача під час виконання вправ.

Висновки до розділу 1

У першому розділі було визначено мету роботи, обґрунтованою актуальністю теми. На основі мети було описано необхідні задачі для досягнення кінцевої мети, у вигляді робочого застосунку. Описані вхідні та вихідні дані допоможуть а подальшому процесі розробки, оцінці вимог і результатів, які будуть отримані від її використання.

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Предметна область даної роботи охоплює процеси фізичної реабілітації пацієнтів із порушеннями функцій кінцівок із використанням сучасних цифрових технологій. У розділі проаналізовано існуючі підходи до реабілітації, визначено їхні переваги та недоліки, а також обґрунтовано доцільність застосування віртуальної реальності як інструменту підвищення ефективності відновлювальних процесів.

2.1 Аналіз існуючих рішень

Лорен Барон, Вутеа Чхеанг та інші розробили віртуальну терапевтичну гру для реабілітації верхніх кінцівок з використанням інтелектуальних носимих датчиків. Вони використовували Unity (версія 2019.1.0f2) для ігрового движка. Користувачу необхідно за допомогою ігрового пензля, який точно повторює рухи рук пацієнта, з'єднати всі ігрові точки, які він бачить перед собою на екрані. Ці точки розташовані у вигляді простих форм, наприклад, такі як риба, ці моделі в залежності від рівня гри мають різні параметри, такі як орієнтація (вертикальна, горизонтальна), так і конфігурація (плоска, вигнута). На рисунку 2.1 зображено один з доступних рівнів.

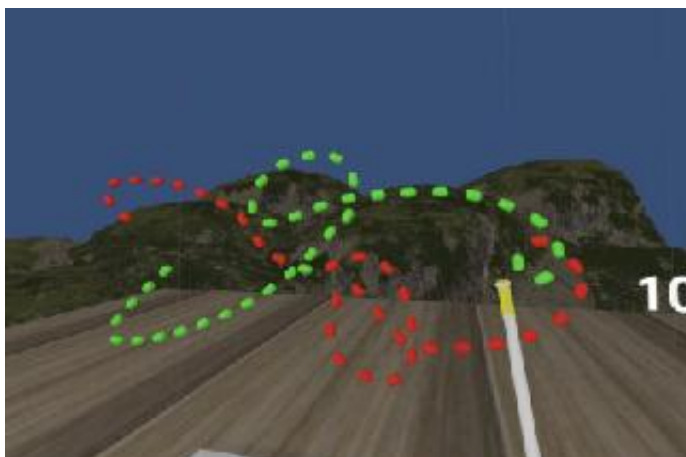


Рисунок 2.1 – Інтерфейс застосунку віртуальної терапевтичної гри для реабілітації верхніх кінцівок [1]

Серед основних переваг додатку, які виділяють його на фоні інших аналогів, можна виділити наступне:

- 1) можливість змінювати позиції пунктирної моделі. Один з контролерів надавав можливість користувачам підлаштовувати висоту і позицію фігури під себе;
- 2) наявність позитивного відклику, при успішному проходженні рівня. Закінчення завдання супроводжувалось візуальними анімаціями феєрверків із звуковими ефектами.

Основними недоліками додатку є:

- 1) дуже малий сет форм фігур. Через те, що рівні відрізняються між собою лише орієнтацією і конфігурацією фігури, то мотивація користувача і бажання продовжувати гру з кожним новим рівнем буде поступово знижуватися;
- 2) відсутність адаптивності. Результати проходження минулих вправ не відіграють ніякого значення в генерації наступних завдань, оскільки рівні є однаковими для всіх пацієнтів;
- 3) датчик-манжета був єдиного розміру, що спричинило поганий збір даних. Це не лише суттєво зменшує зручність використання, а й перешкоджає роботі системи.

У роботі Фан Чжанга, Ланцзин Сюй та інших було запропоновано прототип VR-системи реабілітації нижніх кінцівок із мультисенсорним зворотним зв'язком, реалізований на базі Unity (2019.4.17f1c1). Пристрій використовує Arduino для зчитування даних з датчика дальності та надсилання їх через послідовний порт до UnityEngine. UnityEngine потім надсилає дані до Arduino, щоб визначити, чи потрібно вібрувати та чи потрібно подавати електричну стимуляцію.

Застосунок має три сценарії тренування: рівноваги, ходьби і підйому сходами. Коли користувач проходить тренування від сценарію сфокусованого на рівновазі, то інтерфейс виглядає так наче користувач стоїть на дошці з серфінгу. Цими вправами пацієнт покращує координацію гомілковостопного, колінного і кульшового суглоба. При тренуванні в режимі ходьби, пацієнти зазвичай починають з вправ на кроки з використанням балансира чи інших допоміжних пристроїв. Після досягнення хороших результатів пацієнти можуть виконувати завдання самостійно. Найскладнішим режимом є ходьба сходами. Віртуальна гра, розроблена для цього етапу, називається «Полювання за гірськими скарбами», яка в першу чергу зосереджена на зміцненні здібностей до згинання та розгинання кульшового та колінного суглобів. Вхідні дані базуються на розпізнаванні висоти стопи, тоді як вихідні дані включають вібрацію стопи, низькочастотну електричну стимуляцію, аудіопідказки та інтерактивні підказки на екрані, коли положення правильне. Інтерфейс даного рівня можна бачити на рисунку 2.2.

Крім того всі вищеперечислені сценарії також поділені на рівні складності. Тобто навіть при проходженні одного й того ж сценарію, користувач має можливість обрати вищу складність і потренуватися з нею, роблячи заняття більш цікавими і результативними.



Рисунок 2.2 – Інтерфейс застосунку для реабілітації нижніх кінцівок [2]

Серед основних переваг додатку, які виділяють його на фоні інших аналогів, можна виділити наступне:

- 1) наявність різних сценаріїв реабілітації. Цей додаток здатен охопити різний спектр пацієнтів з різною серйозністю травм:
- 2) наявність різних рівнів складності:
- 3) наявність вібростимуляції і візуальних сигналів при проходженні тренувань позитивно сприяє прогресу пацієнта не лише на рівні психосоматики, а й на рівні сенсомоторної координації та формування правильних рухових патернів.

Основними недоліками додатку є:

- 1) програма була розроблена для тренування лише однієї ноги, тому даний застосунок не підійде пацієнтам з травмами обох ніг;
- 2) малий діапазон рівнів складності. Програмний продукт мав лише два рівня складності сценаріїв, не маючи адаптивних рівней під конкретного пацієнта.

У роботі Уміле Джузеппе Лонго, Аріанни Карневале та інших було запропоновано прототип програми для реабілітації плеча, реалізований на базі Unity (v.2020.17).

Користувач у даному додатку мав використовувати тростину, тримаючи її обома руками, для опорно-рухового апарату плеча. Додаток мав 4 рівня складності, кожен рівень мав свій діапазон необхідних рухових вправ, таких як: підйоми у

сагітальній площині, підйоми у фронтальній площині та інші. Кожен новий рівень вимагає від користувача підняти руки на все більший градус (від 60° на першому рівні і від приблизно 140° до максимального можливого для пацієнта значення амплітуди руху на четвертому рівні).

Крім того в залежності від виду тренувань зовнішній вигляд рівня також змінюється, для тренування на підняття рук вперед-вгору користувачу необхідно було тягнутися за яблуком і потім класти його в кошик, для тренування на підняття рук в бік користувачу необхідно було торкатися фруктів або овочів, а для тренувань пов'язаних з зовнішньою ротацією плеча пацієнт мав «вдаряти» метелика при зігнутому лікті на 90° . Інтерфейс сценарію з тренуванням сагітальної площини можна бачити на рисунку 2.3.

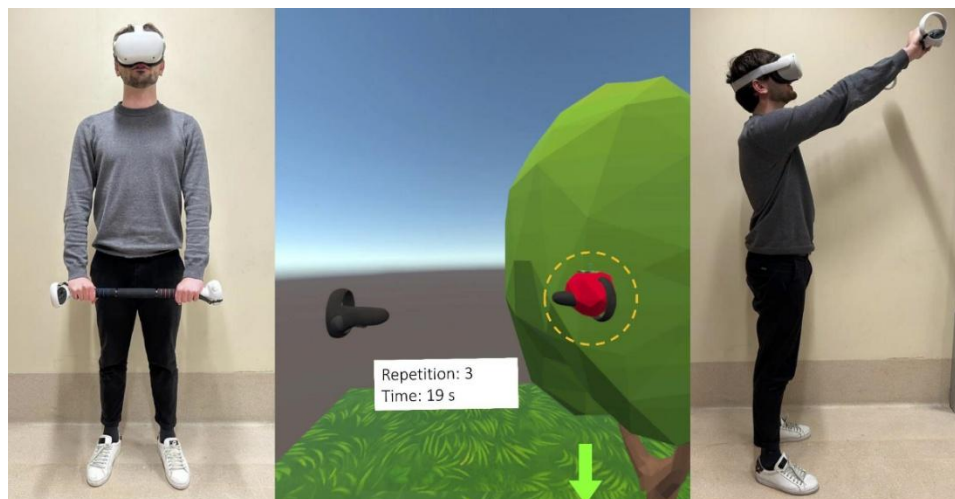


Рисунок 2.3 – Інтерфейс застосунку для реабілітації плеча [3]

Серед основних переваг додатку, які виділяють його на фоні інших аналогів, можна виділити наступне:

- 1) наявність різних сценаріїв і спектру вправ для реабілітації плечових суглобів;
- 2) наявність рівнів складності тренувальних вправ. Це відповідає стандартній практиці фізичної реабілітації, де навантаження збільшується по мірі відновлення пацієнта;

3) рівні побудовані відповідно до реабілітаційних протоколів відновлення плечового суглоба.

Як і в роботі Фан Чжанга, Ланцзин Сюй, у даному додатку головним недоліком є те, що складність рівнів тренувань фіксована і не адаптується під пацієнта. Крім того, іншим недоліком є простота та одноманітність дизайну, гра здається «пустою» і малопривабливою, про що свідчать проведені опитування в самій роботі. Також в опитуванні до даної роботи результати показують, що близько 36% опитуваних вважають, що пацієнти не зможуть самі проходити тренування без допомоги терапевта.

Інший аналог - додаток MIDAS (Multi-sensorial Immersive Dynamic Autonomous System) складається з 3 підсистем, кожна з яких задіює один або кілька органів чуття. Перша підсистема - це екзоскелет руки. Друга підсистема - це підсистема VR, що складається з VR-гарнітури (Oculus Quest 2), VR-гри та VR-контролера. Остання - це нюхова підсистема, що складається з маски для обличчя з нюховим пристроєм та друкованої плати, ця система інтегрована в MIDAS оскільки інші дослідження показують позитивний психофізіологічний вплив запахів на людей, зменшуючи стрес і посилюючи ефект занурення в VR-ігри.

Гра складається з п'яти етапів з різними середовищами. Кожен етап гри триває 3 хвилини. Механізми гри розроблені для обох типів пацієнтів, які перенесли інсульт, тобто пацієнтів, які можуть згинати пальці та пацієнтів, які перенесли інсульт, але пальці рук не згинаються. Весь функціонал гри полягає в тому, що користувачу необхідно чавити лимони (рисунок 2.4), кожен новий рівень вимагає від користувача більше зусиль. Крім того в додаток включено різні рівні звукового супроводу: фонові музика, звуки вичавлювання лимона, щебетання птахів і музика яка сигналізує успішне проходження рівня.



Рисунок 2.4 – Інтерфейс застосунку MIDAS для реабілітації верхніх кінцівок [4]

Серед основних переваг додатку, які виділяють його на фоні інших аналогів, можна виділити наступне:

1) використання екзоскелета для руки. Цей екзоскелет допомагає пацієнтам стискати і розтискати пальці рук, тим самим надаючи можливість виконувати вправи пацієнтам навіть при дуже обмеженій рухливості руки;

2) використання біосигналів. MIDAS аналізує електроміографічні сигнали передпліччя, щоб визначити намір пацієнта виконати рух. Це дозволяє системі активувати допоміжний механізм у потрібний момент;

3) наявність рівнів складності тренувальних вправ.

Серед недоліків можна виділити:

1) відсутність адаптивності. Гра має лише чотири рівня складності, які є однаковими для всіх пацієнтів;

2) одноманітність завдань. Єдиний функціонал передбачений розробниками – це чавіння лимона. Рівні можна розрізняти між собою лише за зовнішніми декораціями;

3) складність апаратної системи. Як зазначалось раніше, система складається з трьох підсистем, в які входять: VR-гарнітура, екзоскелет, EMG-сенсори. Тим самим роблячи продукт технічно складним і дорогим для масового виробництва.

1.2 Адаптація складності вправ

Традиційно розробники ігор реалізують систему проходження за допомогою фіксованих рівнів складності, які визначаються заздалегідь і не змінюються під час взаємодії користувача з системою - «легкий», «середній» або «складний», але такий підхід не враховує весь спектр потреб гравців. Проведений аналіз показав, що ключовим недоліком розглянутих аналогів є відсутність адаптивності тренувальних вправ.

Говорячи про адаптивність в додатках, люди часто фокусуються лише на індивідуальних здібностях гравця в моменті оцінки і на його оцінці результатів проходження, проте важливим фактором, яким часто нехтують - є емоційний стан користувача. Гравці можуть мати однакову оцінку індивідуальних здібностей і результатів проходження завдань при тому відчувати зовсім різні емоції, хтось відчуватиме задоволення від досвіду проходження, а в когось навпаки переважатиме стрес і знервованість. Це фактори про котрі не варто забувати.

Однією з найвпливовіших концепцій, що лежать в основі DDA(Dynamic Difficulty Adjustment), є теорія потоку психолога Міхая Чікцентміхайі [5]. Потік описує психічний стан глибокого занурення, що характеризується інтенсивною концентрацією, відчуттям контролю та внутрішньою насолодою від самої діяльності. Для насолоди процесом необхідно зберігати баланс між можливостями гравця, складністю і насолодою, якщо гравець не почувався занадто нервово чи нудьгує, то даний баланс порушено. Цей баланс ілюструється моделлю на рисунку 2.5, яка наочно показує, як виникає оптимальний досвід, коли складність завдання відповідає індивідуальним здібностям.

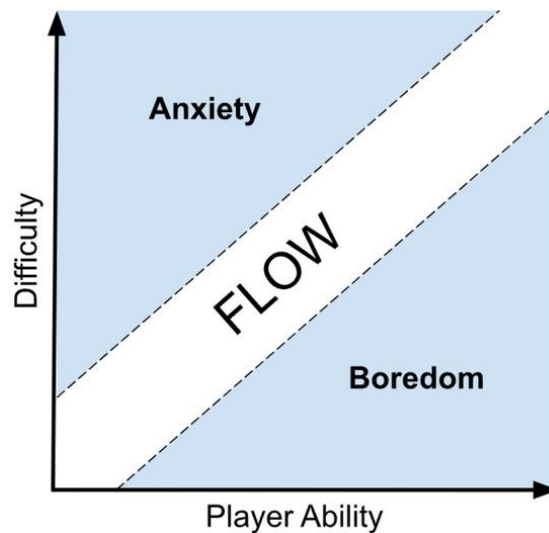


Рисунок 2.5 – Модель балансу між складністю і можливостями користувача [6]

Динамічна адаптація складності (DDA) у сучасних дослідженнях та практичних застосуваннях, як правило, ділиться на три основні категорії:

- 1) системи, що базуються на результатах – враховують такі метрики як: точність, час реакції, співвідношення перемог і поразок, а також залишок здоров'я;
- 2) системи, що базуються на емоціях - враховують такі метрики як: стрес, нудьгу або когнітивне перевантаження. Ці метрики збираються за допомогою моніторів частоти серцевих скорочень, сенсорів шкірної провідності та пристроїв для вимірювання мозкової активності, зокрема електроенцефалографічної (EEG) гарнітури;
- 3) гібридні – комбінують обидва підходи систем.

Однак при всіх доступних перевагах, які пропонує гібридна система варто пам'ятати, що продукт при її імплементації стає технічно складним для реалізації і дорогим для масового виробництва, що показав на своєму прикладі розглянутий раніше додаток MIDAS. Для економії ресурсів можна прийняти рішення про додавання коротких опитувань для користувача про його емоційний стан після кожного проходження рівня, тим самим включаючи отримані дані інших метрик при організації наступного рівня замість використання сенсорів і іншої спеціальної гарнітури. І крім самооцінки настрою в систему можна додати аналіз залученості пацієнта, тобто враховувати: тривалість сесії, кількість добровільних повторів,

перерву між сеансами гри. Ці дані можуть посприяти більш об'єктивній оцінці мотивації, втоми та нудьги.

На основі цього аналізу можна перейти до вивчення можливих варіантів реалізації адаптивного алгоритму підбору складності тренувань. У таблиці 2.1 наведено порівняння деяких алгоритмів.

Таблиця 2.1 – Порівняння алгоритмів адаптації

	Rule-based алгоритм	Heuristic алгоритм	Reinforcement Learning
Простота реалізації	Легка	Помірна	Склана
Гнучкість	Низька. Можливі зміни лише при модифікації умов	Середня через використання загальних, гнучких вказівок, а не жорстких, заздалегідь визначених правил	Висока, адаптується до кожного користувача
Швидкість прийняття рішень	Висока	Висока	Низька, потребує часу для навчання
Вимоги до даних	Невеликі	Невеликі	Потрібні великі масиви даних для навчання
Можливість навчання	Відсутня	Пристуня	Пристуня
Точність	Середня, залежить від прописаних правил	Середня, залежить від прописаних інструкцій	Висока

Висновки до розділу 2

Проаналізувавши доступні аналоги, можна прийти до рішення, що для побудови ефективного програмного продукту доцільно поєднати ключові переваги і при цьому недопустити недоліки даних систем.

Водночас аналіз показав, що одним із найбільш суттєвих недоліків більшості існуючих систем є обмежена адаптивність реабілітаційного процесу. У багатьох випадках програми використовують заздалегідь визначені сценарії та рівні складності, які є однаковими для всіх пацієнтів. Такий підхід не враховує індивідуальні особливості стану пацієнтів та рівень їхнього функціонального відновлення, що може знижувати ефективність реабілітації, особливо для пацієнтів із більш складними травмами.

Для подолання цієї нерівності необхідно сфокусуватись на розробленні адаптивного алгоритма підбору вправ, який дозволяє персоналізувати реабілітацію.

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі наведено опис основних засобів розробки програмного забезпечення, що використовувалися під час створення VR-додатка для реабілітації нижніх кінцівок. Розглянуто ключові технології, мови програмування та інструменти, які забезпечили реалізацію клієнтської та серверної частин системи, а також створення інтерактивного віртуального середовища. Особливу увагу приділено обґрунтуванню вибору кожного з інструментів з урахуванням вимог до продуктивності, масштабованості та зручності розробки.

3.1 Середовище розробки Unity

Для розробки клієнтської частини VR-додатка було використано Unity - кросплатформний ігровий рушій, який широко застосовується для створення інтерактивних 2D та 3D застосунків, зокрема відеоігор, симуляцій і VR/AR-рішень [7].

Unity надає розробникам повний набір інструментів для створення тривимірних сцен, фізичних симуляцій, анімацій, користувацьких інтерфейсів та інтерактивної логіки. Рушій підтримує роботу з різними платформами, включаючи ПК, мобільні пристрої та VR/AR-пристрої, що робить його універсальним рішенням для мультиплатформної розробки.

Однією з ключових переваг Unity є інтеграція з мовою програмування C#, що дозволяє реалізовувати складну ігрову логіку, обробку подій та взаємодію з об'єктами сцени. Також рушій підтримує велику кількість готових інструментів, таких як фізичний рушій, система анімацій, Shader Graph, UI Toolkit та підтримка VR-пристроїв через OpenXR [8].

Для Editor було обрано версію 6000.4.0f1 оскільки на момент написання диплому вона була рекомендованою. Про це важливо пам'ятати, оскільки через несумісність версій проект може взагалі не відкритись (рисунок 3.1).

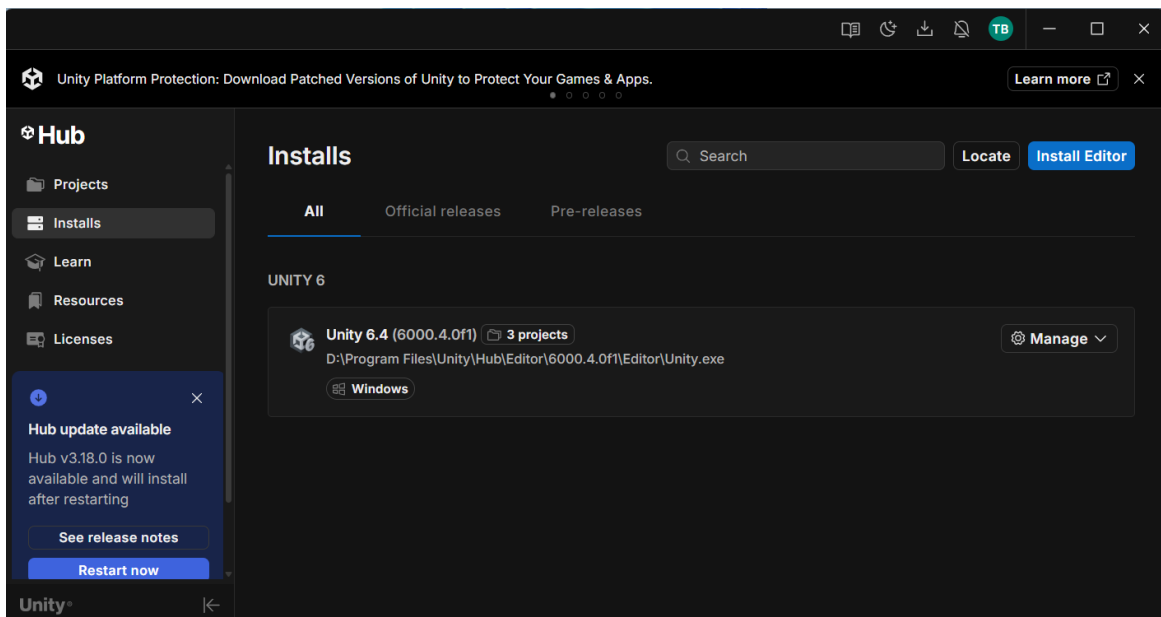


Рисунок 3.1 – Вікно Unity Hub

Крім того при першому імпорті нових асетів у проект може виникнути проблема з шейдерами, щоб виправити це є два способи:

- 1) перейти в Inspector, потім знайти секцію Material і в випадаючому списку обрати Universal Render Pipeline (Complex Lit/Lit);
- 2) перейти в налаштування проекту і перевірити, чи встановлений Render Pipeline Asset.

3.2 Система керування базами даних PostgreSQL

У якості системи керування базами даних було обрано PostgreSQL. Це потужна об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, яка підтримує та розширює можливості мови SQL, а також надає широкий набір вбудованих функцій для забезпечення надійного зберігання, обробки та масштабування даних навіть у складних інформаційних системах. Система підтримує роботу на всіх основних операційних системах та з 2001 року відповідає стандарту ACID, що гарантує надійність транзакцій, цілісність даних та стійкість до збоїв. Крім того, PostgreSQL має розширювану архітектуру та

підтримує велику кількість додаткових модулів, зокрема PostGIS для роботи з геопросторовими даними [7].

До основних переваг PostgreSQL можна віднести: високу надійність та відповідність принципам ACID, підтримку складних SQL-запитів та аналітичних операцій, високу продуктивність при роботі з великими обсягами даних, кросплатформеність та стабільну роботу на різних ОС.

NoSQL-рішення не були обрані для даного проєкту, оскільки структура даних у системі є чітко визначеною та має складні взаємозв'язки між сутностями (користувачі, вправи, сесії, результати та історія прогресу). У таких умовах реляційна модель є більш доцільною, оскільки забезпечує цілісність даних за рахунок зовнішніх ключів, підтримує складні зв'язки між таблицями та дозволяє ефективно виконувати аналітичні SQL-запити, необхідні для оцінювання та адаптації складності вправ.

3.3 Мова C#

Для реалізації серверної та клієнтської логіки застосунку було обрано мову програмування C#. Даний вибір обумовлений тим, що основною платформою розробки є Unity, у якій саме C# є основною та офіційно підтримуваною мовою програмування для створення ігрової логіки, взаємодії з об'єктами сцени та реалізації скриптів.

C# є сучасною об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft [8]. Вона поєднує потужність низькорівневого контролю з високорівневими можливостями абстракції, що робить її зручною для розробки складних програмних систем. Мова підтримує такі концепції, як об'єктно-орієнтоване програмування (ООП), асинхронне програмування, роботу з делегатами та подіями, а також має велику стандартну бібліотеку для роботи з даними, мережею та файловою системою.

Використання C# є особливо доцільним у розробці VR-застосунків, оскільки він забезпечує високу продуктивність та ефективну роботу з об'єктами сцени у реальному часі. У поєднанні з Unity мова дозволяє зручно реалізовувати взаємодію з VR-контролерами, обробку подій введення та фізичних взаємодій у 3D-просторі. Крім того, підтримка компонентно-орієнтованої архітектури Unity та інтеграція з його API дозволяє швидко створювати інтерактивні VR-системи з мінімальними затримками, що є критично важливим для забезпечення комфортного користувацького досвіду та уникнення VR-дискомфорту [9].

Висновки до розділу 3

У даному розділі було розглянуто основні засоби розробки програмного забезпечення, що використовувалися під час створення VR-додатка для реабілітації нижніх кінцівок. Описано ключові технології, мови програмування та інструменти, зокрема Unity та C#, а також обґрунтовано їх вибір з урахуванням вимог проєкту. Використані технології забезпечили ефективну реалізацію функціоналу системи, її продуктивність та можливість подальшого розвитку

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Реалізація програмного забезпечення є ключовим етапом розробки системи, що забезпечує перехід від концептуальної моделі до функціонального продукту. У даному розділі описано процес програмної реалізації VR-додатка, його архітектурні особливості, основні модулі та технологічні рішення, які були використані для забезпечення його роботи.

4.1 Архітектура додатку

Архітектура розробленого додатка побудована за принципом 3-tier architecture (трирівневої архітектури) [7]. Даний підхід передбачає поділ системи на три основні рівні: рівень представлення, рівень логіки та рівень даних (зображено на рисунку 4.1).

Рівень представлення (Presentation Layer) відповідає за взаємодію кінцевого користувача із системою. У межах даного проєкту цю роль виконує застосунок, розроблений у Unity, де користувач взаємодіє з VR-середовищем, об'єктами сцени та інтерфейсом додатка.

Рівень логіки (Business Logic Layer) є центральною частиною архітектури та відповідає за обробку даних, виконання обчислень, реалізацію бізнес-логіки й взаємодію між користувацьким інтерфейсом та базою даних. У розробленій системі цей рівень забезпечує надсилання запитів до сервера, обробку результатів виконання вправ, аналіз рухів користувача та повернення відповідних даних до рівня представлення.

Рівень даних (Data Layer) відповідає за зберігання, отримання та керування даними додатка. Для реалізації даного рівня було використано систему керування базами даних PostgreSQL.

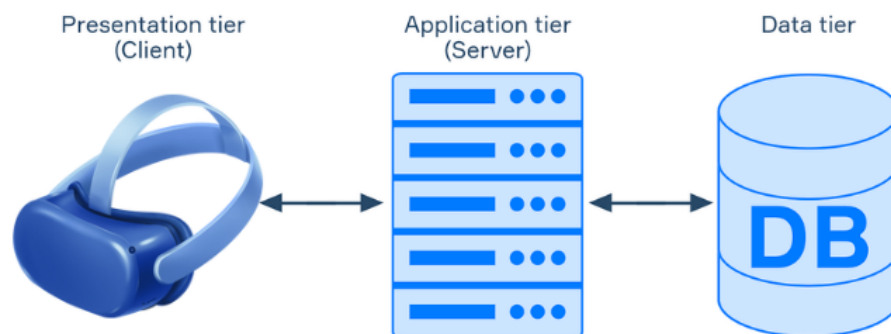


Рисунок 4.1 – Трирівнева архітектура

Вибір трирівневої архітектури був зумовлений її гнучкістю, масштабованістю та чітким розділенням відповідальностей між компонентами системи. Серед основних переваг даної архітектури можна виділити наступне:

- 1) розділення функціональності між окремими рівнями системи;
- 2) спрощення підтримки та модифікації коду;
- 3) можливість незалежного оновлення окремих компонентів;
- 4) підвищення масштабованості системи;
- 5) покращення безпеки завдяки відокремленню бази даних від клієнтської частини;
- 6) спрощення тестування окремих модулів системи.

4.2 Архітектура бази даних

Під час розробки було створено базу даних (зображено на рисунку 4.2), оскільки вона необхідна для зберігання даних отриманих під час занять пацієнтів і подальшого їх використання для алгоритму адаптації складності вправ. Крім того, дану базу можна використовувати для майбутніх статистичних досліджень, побудови моделі машинного навчання і адміністрування даними пацієнтів у корпоративних лікарських додатках.

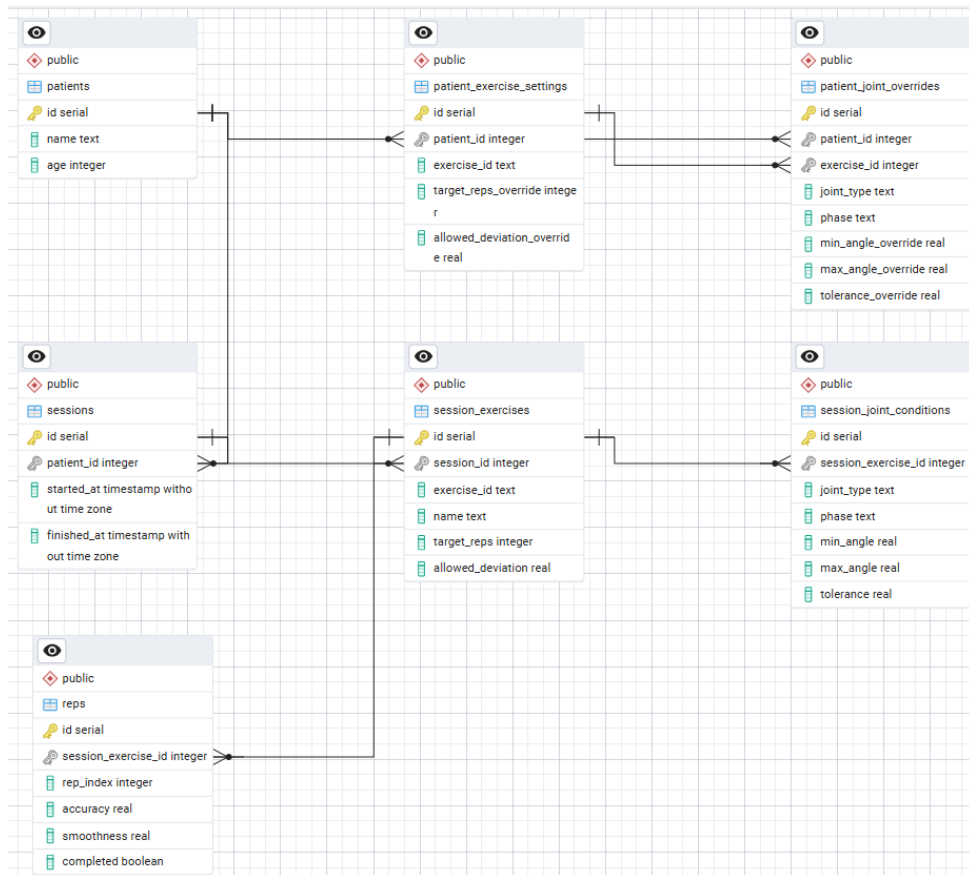


Рисунок 4.2 – Діаграма структури БД

База побудована за першими нормальними формами. Таблиці створювались виходячи з думки, що необхідно мати записи для поточних налаштувань індивідуальних вправ для пацієнтів (*patient_exercise_settings*, *patient_joint_overrides*) і записи з інформацією про налаштування вправ, які були виконані пацієнтом під час минулих сесій (*sessions*, *session_exercise*, *session_joint_conditions*, *reps*). Залежності між моделями:

1) налаштування поточної вправи:

- а. кожна вправа має пацієнта, до якого належить;
- б. кожна вправа має еталонний зразок виконання;

2) кожен суглоб, з інформацією про необхідні параметри виконання, належить до якоїсь вправи;

3) кожна сесія виконаних вправ належить пацієнту;

4) вправа окремої сесії належить конкретній сесії, яку користувач виконав в минулому;

5) кожен суглоб, з інформацією про необхідні параметри виконання, належить до якоїсь вправи;

6) кожен повтор посилається на конкретну вправу, яка була виконана під час однієї з минулих сесій.

Кожна вправа містить фази виконання, такі як: start, peak і finish. Start відповідає за необхідну стартову позицію, peak – пікова точка, при якій досягається максимальний кут згибання або ж ротації, finish – кінцева позиція виконання, дана фаза може бути відсутня у разі, якщо після пікової точки необхідно повернутися назад у стартову для зарахування повтору.

Оскільки додаток розроблявся для реабілітації нижніх кінцівок, то типи суглобів, які розпізнає алгоритм наступні: колінний, кульшовий, гомілковостопний і суглоби пальців стопи. Для складних вправ передбачена можливість того, що при виконанні однієї і тієї ж прави, в різних фазах виконання можуть бути задіяні різні суглоби і при цьому алгоритм справно працюватиме.

Мінімальний і максимальний кут відповідають за ідеальний (базовий) діапазон правильної пози. Tolerance - допустиме розширення діапазону, тобто значення наскільки можна помилитися. Без tolerance система буде надто жорстка по відношенню до користувача. Tolerance не варто плутати з deviation, tolerance відповідає за локальну точність для конкретного суглоба і фази, а deviation – глобальне допустиме значення різниці виконання і необхідного руху.

Таблиця reps зберігає інформацію про середні показники точності, плавності виконання вправи та іншу інформацію під час однієї з сесій, отримані за допомогою обрахунку середніх значень повторів даної вправи. Для майбутніх покращень алгоритму оцінки правильності виконання завдань було передбачено поле rep_index, за допомогою якого можна відстежувати параметри оцінки кожного повтору прави під час конкретної сесії.

Важливо зауважити, що система будувалась за сценарієм домашньої реабілітації, тобто кожен VR шолом і інша гарнітура строго прив'язана до кожного пацієнта, у відношенні один до одного, проте за потреби, систему легко масштабувати до відношення один VR шолом для кількох пацієнтів.

Поділ таблиць на поточні і минулі налаштування систем вправ зумовлений функціональним призначенням даних, яке надає можливість швидкого пошуку і сортування записів, при цьому зберігаючи простоту написання запитів. При необхідності, для пришвидшення пошуку необхідної інформації можна проіндексувати наступні колонки:

- 1) patient_id в таблиці sessions;
- 2) session_id в таблиці session_exercises;
- 3) session_exercise_id в таблиці reps.

4.3 Реалізація серверної частини

Об'єктно-реляційне відображення (ORM) - це процес абстрагування зв'язку між сутностями мови програмування (об'єктами) та відповідними елементами бази даних [8]. Було використано Entity Framework Core це дозволяє розробникам взаємодіяти з базами даних за допомогою об'єктів .NET (C#) замість написання сирих SQL-запитів.

Крім того, ORM-підхід дозволив використовувати механізм міграцій (migrations), завдяки якому структура бази даних могла автоматично оновлюватися відповідно до змін у моделях додатка. Це значно спростило процес модифікації бази даних під час розробки системи.

У межах розробки серверної частини додатка було використано підхід Code First, який передбачає початкове створення моделей даних у кодї з подальшим автоматичним формуванням структури бази даних. Даний підхід забезпечив гнучкість під час розробки системи та спростив внесення змін до структури бази даних за допомогою механізму міграцій (migrations).

Серверна частина системи реалізована у вигляді REST API, що забезпечує обробку даних у реальному часі та виконання алгоритму адаптації складності вправ. Основною точкою входу є HTTP-ендпоінт /adaptation, який отримує поточні дані сесії пацієнта, включаючи результати виконання вправ, суб'єктивні показники (біль, втоми) та ідентифікатори пацієнта і вправи. Після отримання запиту сервер виконує вибірку історичних даних пацієнта з бази даних PostgreSQL. Зокрема, аналізуються попередні повтори вправ (reps), які агрегуються для обчислення: середньої точності виконання, середньої плавності рухів, рівня завершеності вправ. Ці метрики формують базовий профіль прогресу пацієнта.

На основі поточних та історичних даних викликається модуль адаптації (AdaptationEngine), який реалізує логіку Dynamic Difficulty Adjustment. Результатом роботи алгоритму є: узагальнений бал (finalScore), рішення щодо зміни складності (increase / decrease / maintain). Після прийняття рішення сервер модифікує індивідуальні налаштування пацієнта: кількість повторень вправ, допустиме відхилення руху, діапазони кутів для суглобів. Таким чином реалізується персоналізація складності під конкретного користувача.

Таким чином, серверна частина виконує роль центрального модуля прийняття рішень, який: агрегує дані з поточних та історичних сесій, виконує аналіз прогресу пацієнта, застосовує алгоритм адаптації складності, та забезпечує збереження повної історії взаємодії користувача із системою.

4.4 Анімація

Unity надає вбудовані засоби для створення анімацій за допомогою компонента Animation та вікна Animation (рисунок 4.3). Для створення анімації необхідно обрати відповідний об'єкт сцени та додати до нього анімаційний кліп. Далі зміна параметрів об'єкта (позиції, обертання, масштабу тощо) здійснюється покадрово шляхом встановлення ключових кадрів (keyframes) на часовій шкалі (timeline). Редагування анімації може виконуватися як через числові значення у

режимі Dopesheet, так і за допомогою інструментів трансформації (Move Tool, Rotate Tool), що дозволяє більш інтуїтивно змінювати положення об'єкта у просторі. Ключові кадри та анімаційні події (events) можуть бути додані в будь-який момент часу на таймлайні, що забезпечує гнучкість у налаштуванні поведінки об'єкта. Додатково, вкладка Curves дозволяє редагувати криві інтерполяції між ключовими кадрами, що дає можливість контролювати швидкість та плавність анімації. Зокрема, за допомогою налаштування кривих можна досягти більш природних переходів між рухами або реалізувати циклічні анімації без різких стрибків.

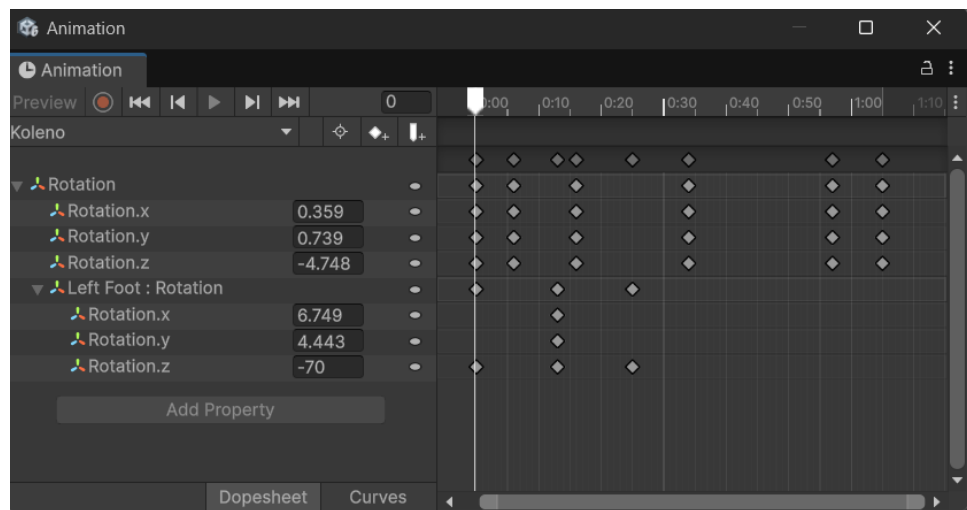


Рисунок 4.3 – Вікно Animation в Unity

Крім того, за допомогою компонента Animator у Unity можна створювати графи переходів між анімаціями. Для цього визначаються стани (states), що відповідають окремим анімаціям, умови переходів між ними (transitions), а також параметри, значення яких можуть змінюватися програмно. На рисунку 4.4 наведено приклад такого графа. Використання Animator Controller дозволяє керувати відтворенням анімацій залежно від контексту та логіки додатку. Зокрема, це дає змогу активувати анімацію лише у потрібний момент або автоматично змінювати її при виконанні певних умов. Додатково, система підтримує використання шарів (layers), що дозволяє незалежно анімувати різні частини моделі. Наприклад, персонаж може одночасно виконувати рух рукою та згинати ногу. Для плавного

переходу між подібними анімаціями застосовується механізм Blend Tree. Він дає можливість інтерполювати між кількома анімаційними станами залежно від значення параметрів, забезпечуючи природний перехід, наприклад, від ходьби до бігу.



Рисунок 4.4 – Вікно Animator в Unity

Існують спеціалізовані сервіси з готовими анімаціями, такі як ActorCore та Міхамо (на рисунку 4.5), які дозволяють не лише імпортувати анімації у власний проєкт, але й використовувати готові 3D-моделі персонажів. Дані платформи містять як безкоштовний, так і платний контент, що значно спрощує процес створення анімованих сцен.

Проте в межах даного проєкту було прийнято рішення відмовитися від використання готових анімацій на користь розробки власних, зважаючи на такі причини:

- 1) відсутність спеціалізованих реабілітаційних вправ, погоджених із профільними спеціалістами та адаптованих під потреби користувачів системи;
- 2) самостійне створення анімацій забезпечує глибше розуміння принципів побудови рухів, роботи з ригінгом, анімаційними станами та інтеграцією моделей у

VR-середовище, а також демонструє практичні навички роботи з 3D-графікою та анімацією;

3) власні анімації дозволяють точніше контролювати амплітуду, швидкість та правильність виконання рухів, що є особливо важливим для реабілітаційних вправ;

4) використання власноруч створених анімацій забезпечує кращу адаптацію системи до алгоритмів оцінки руху, реалізованих у додатку, оскільки всі фази вправ можуть бути налаштовані відповідно до логіки аналізу виконання.

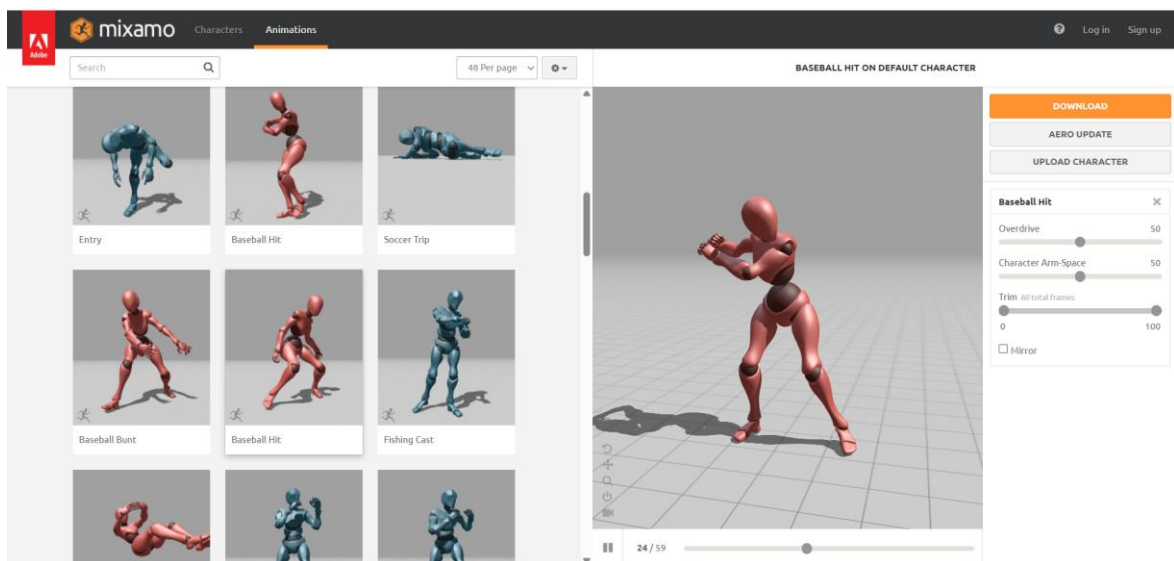


Рисунок 4.5 – Веб сторінка Міхамо

Для кращого розуміння анатомічної будови людського тіла було використано застосунок Anatomy Learning, доступний для платформ App Store, Google Play, а також у вигляді вебзастосунку (зображено на рисунку 4.6). Використання даного інструменту дозволило детальніше вивчити природну анатомію людських рук та особливості руху суглобів, що в подальшому було враховано під час створення анімацій для розроблюваного додатка.

Застосунок надає можливість взаємодіяти з різними частинами людського тіла та досліджувати їх у кількох режимах відображення, зокрема у вигляді кісткової системи, м'язового шару та інших анатомічних структур. Це дало змогу краще зрозуміти механіку рухів і точніше відтворити їх у VR-середовищі.

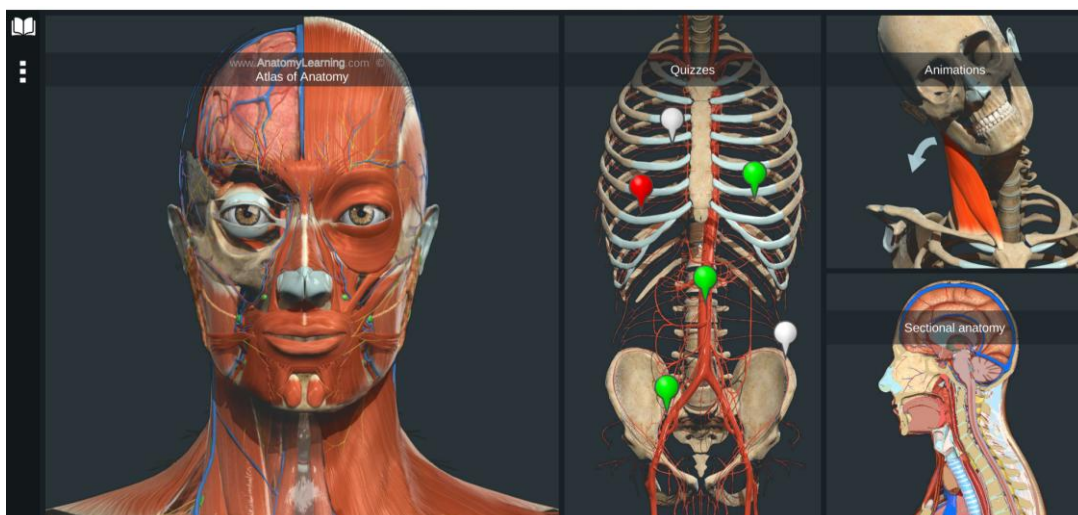


Рисунок 4.6 – Веб сторінка Anatomy Learning

4.5 Створення UI елементів

Unity надає доступ до офіційної платформи Unity Asset Store, яка дозволяє знаходити безкоштовні та комерційні 2D- і 3D-ресурси для використання у власних проєктах. На сторінці кожного пакета міститься інформація про автора, розмір файлів, сумісність із версіями Unity Editor та умови використання. Використання таких публічно доступних асетів дозволяє суттєво пришвидшити процес розробки. Оскільки серед основних вимог до даного дипломного проєкту не передбачалося створення власних 3D-моделей, більшість тривимірних об'єктів було імпортовано саме з Asset Store. Водночас із 2D-елементами ситуація виявилася менш однозначною. Хоча значна частина іконок також була взята з Asset Store, у процесі розробки виникла потреба у додаткових графічних елементах, які були відсутні у наявних наборах. У зв'язку з цим було прийнято рішення створити власні іконки, зберігаючи при цьому стилістику вже використаних ресурсів. Розробка іконок здійснювалася у середовищі Figma (рисунок 4.7). Для цього використовувалися інструменти Pen та Bend, за допомогою яких відтворювалися контури базових елементів. Надалі ці елементи комбінувалися шляхом накладання шарів, що дозволило досягти візуальної цілісності та наближеності до оригінального стилю.

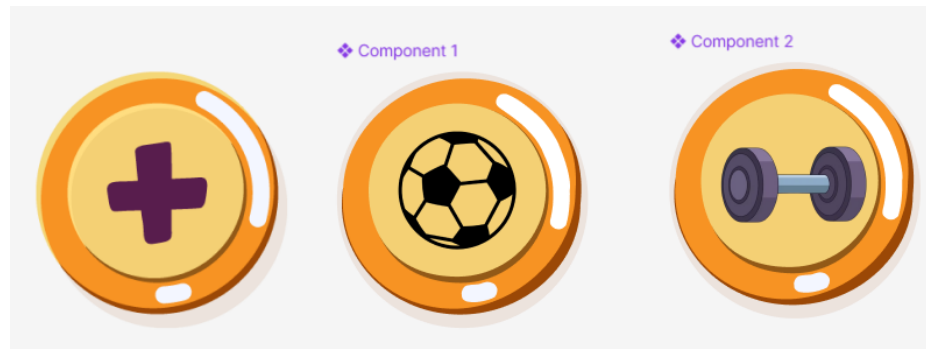


Рисунок 4.7 – Дизайн 2D елементів в Figma

Після створення зображення імпортувалися до Unity та налаштовувалися як тип Sprite (2D and UI). Далі у вікні Inspector відповідні зображення призначалися у властивість Source Image для UI-елементів, зокрема кнопок.

Unity також надає вбудований інструмент для візуального створення шейдерів — Shader Graph. Він дозволяє формувати матеріали шляхом поєднання вузлів (nodes), що описують математичні операції та властивості поверхні. У процесі розробки ігрового середовища виникла потреба у специфічних візуальних елементах, які були відсутні серед безкоштовних ресурсів Unity Asset Store. У зв'язку з цим було прийнято рішення реалізувати власний шейдер для зони активації геймплею. Розроблений Shader Graph (рисунок 4.8) складається з набору вузлів, зокрема: Position, Subtract, Power, Saturate та інших, які використовуються для обробки координат, формування градієнтів і контролю інтенсивності ефекту.

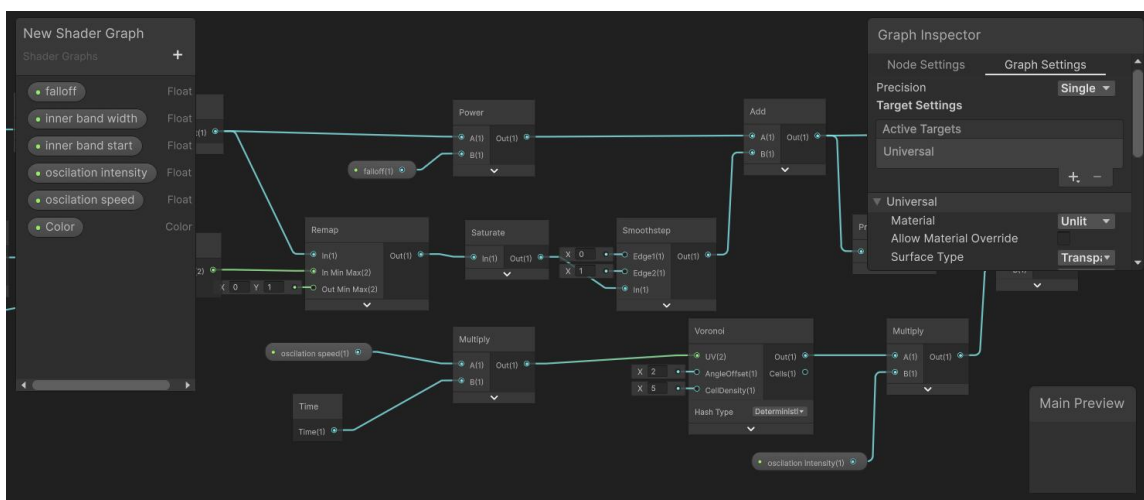


Рисунок 4.8 – Вікно Shader Graph в Unity

Наприклад, для створення власного range indicator було використано Blender. У режимі редагування базовий циліндр було модифіковано шляхом видалення верхньої та нижньої граней із збереженням лише бокової поверхні, що дозволило сформувати необхідний візуальний елемент. Після цього модель була експортована у форматі .fbx та імпортована в Unity. Для відображення ефекту розсіювання в просторі від периметра об'єкта було застосовано попередньо створений матеріал із використанням Shader Graph, що забезпечило більш наочну та динамічну візуалізацію області дії індикатора.

Також Blender було використано для редагування готової 3D-моделі людини з метою видалення верхньої частини тіла. У режимі Edit Mode було виділено полігони, що належать нижнім кінцівкам, після чого за допомогою функції Selection їх було відокремлено в окремий об'єкт (зображено на рисунку 4.9). Надалі непотрібну частину моделі було видалено, а змінену модель експортовано назад у форматі .fbx.

Необхідність такого редагування була зумовлена специфікою розроблюваного додатка, який орієнтований на реабілітацію нижніх кінцівок. Модель повністю повторює рухи користувача, тому використання лише нижньої частини тіла дозволило уникнути зайвих візуальних елементів, які могли б ускладнювати сприйняття рухів та заважати під час виконання вправ.



Рисунок 4.9 – Редагування моделі в Blender

Редагування моделі було виконано саме в Blender, оскільки спроба видалити верхню частину безпосередньо в Unity призводила до пошкодження структури моделі через наявні зв'язки між body, armature та joints.

4.6 Алгоритми

Під час процесу розробки алгоритмів оцінки правильності виконання вправи і адаптації складності на основі індивідуального прогресу пацієнта, постало питання про необхідність пошуку вичерпного датасету. Проте виникли наступні проблеми:

- 1) мала кількість публічнодоступних датасетів з реабілітаційними вправами;
- 2) датасети мають занадто мало даних;
- 3) відсутність маркування правильних і неправильних виконань вправ;
- 4) відсутність метаданих, неможливість інтерпретувати інформацію надану в датасеті;
- 5) переважаюча кількість сирих даних з сенсорів, для попередньої обробки яких здадобиться надто багато ресурсів.

Важливо зазначити, що існують публічнодоступні датасети кінематики рухів нижніх кінцівок здорових людей. Такі набори даних зазвичай застосовуються для навчання моделей відтворювати природні та плавні людські рухи. Проте їх використання в межах даного додатку є недоцільним, оскільки вони не мають необхідної медичної валідації, а також не враховують специфіку рухових обмежень пацієнтів із травмами.

Існує поширена практика, за якої розробники в межах досліджень або впровадження реабілітаційних додатків використовують для навчання моделей машинного навчання дані, зібрані за допомогою власних сенсорів у межах їхньої системи. Це дозволяє забезпечити повну відповідність між апаратним забезпеченням, використаним під час формування датасету, та гарнітурою, що застосовується при експлуатації системи кінцевими користувачами. Проте через

обмеженість ресурсів такий підхід не розглядався для реалізації у межах даного додатку.

Виходячи з усього вищеперечисленого в даному пункті, було прийнято рішення відмовитись від імплементації алгоритму машинного навчання і створити власний.

На рисунку 4.10 описаний алгоритм адаптації складності вправ, він включає в себе такі основні процеси як: розрахунок балу поточної сесії користувача, аналіз історії прогресу пацієнта, оцінка емоційного стану користувача і підрахунок фінальної оцінки.

Оцінка емоційного стану користувача включає в себе міні опитування, яке фокусується на власній суб'єктивній оцінці пацієнта про власний стан. Ці питання допомагають врахувати показники, які неможливо вирахувати апаратним забезпеченням, такі як: рівень болю, рівень втоми і рівень зацікавленості.

Аналіз історії прогресу пацієнта відбувається за допомогою отримання даних за його минулі сесії, записані в базі даних, сортуючи їх від старіших до найновіших, щоб врахувати фактор тенденції покращення чи навпаки погіршення таких показників як точність і плавність.

До оцінки поточної сесії входить також фактор завершеності. Користувач має можливість достроково припинити виконання вправи, не досягнувши необхідної кількості повторів, що суттєво знижує підсумковий бал. При цьому можливість продовження вправи після повного виходу із сесії не передбачена. Таке рішення обумовлене необхідністю забезпечення однакових умов для всіх пацієнтів, оскільки нестабільне або переривчасте виконання може спотворювати результати оцінювання. Кожна реабілітаційна вправа передбачає оптимальний час виконання без значних перерв, тому суттєве відхилення фактичного часу від норми повинно враховуватися при формуванні підсумкової оцінки. Зокрема, надто швидке виконання може призводити до перевантаження та ускладнення травм, тоді як надто повільне - знижує ефективність реабілітації та не забезпечує досягнення бажаного результату. Даний всепоглинаючий підхід оцінки, який охоплює різні

сфери показників пацієнта, сприяє більш зваженій фінальній оцінці.

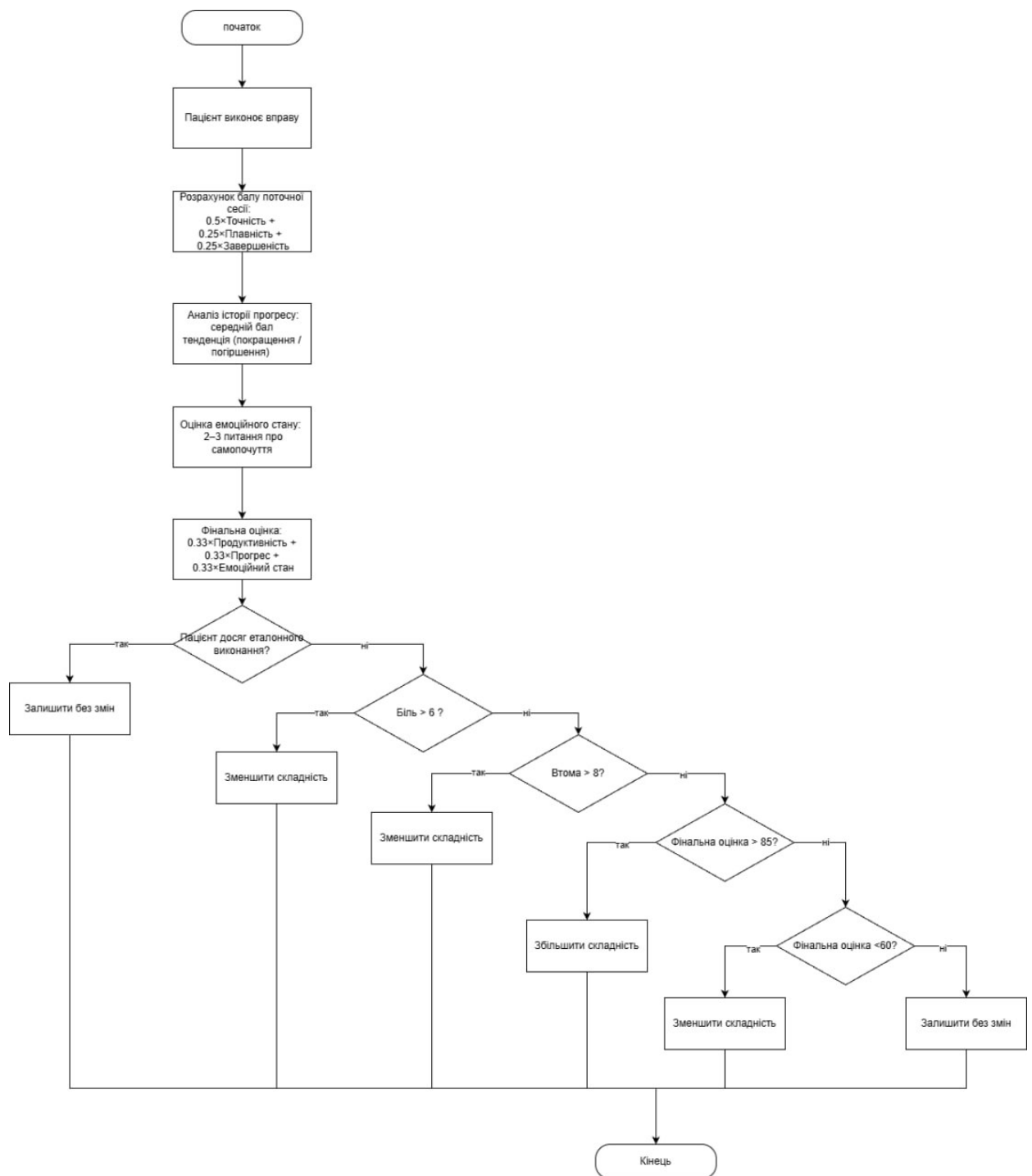


Рисунок 4.10 – Алгоритм адаптації складності вправ

На рисунку 4.11 представлено IDEF0-діаграму рівня A-0, що описує процес формування оцінки поточної сесії. Вона демонструє основні вхідні дані, керуючі

впливи, механізми та результати роботи підсистеми оцінювання виконання реабілітаційних вправ.

Оцінка точності виконання вправи здійснюється шляхом порівняння фактичного значення кута суглоба із цільовим значенням, яке визначається як середина допустимого діапазону.

Похибка визначається як модуль різниці між фактичним та цільовим значенням і нормалізується відносно допустимого відхилення (tolerance). Отримане значення інвертується для формування оцінки точності, де нульова похибка відповідає максимальному значенню.

Для забезпечення коректності результату використовується обмеження значення у діапазоні [0;1], після чого обчислюється середнє значення для всіх задіяних суглобів та переводиться у відсоткову шкалу.

Оцінка плавності руху (smoothness) базується на аналізі зміни швидкості руху суглоба між послідовними кадрами. Для цього обчислюється різниця між швидкостями (jerk), що характеризує різкість руху.

Чим менша зміна швидкості, тим більш плавним вважається рух. Середнє значення jerk використовується для обчислення узагальненої метрики плавності за формулою 4.1

$$\text{smoothness} = \frac{1}{1 + \text{avgJerk}}, \quad (4.1)$$

де avgJerk – середня зміна швидкості між кадрами

Підсумкова оцінка визначається як середнє значення плавності для всіх задіяних суглобів.

Важливо зазначити, що показник плавності обчислюється протягом усього виконання вправи, оскільки він відображає не окрему позу, а характер руху в динаміці. На відміну від метрик точності пози, які оцінюють відповідність еталонному положенню у визначеній фазі руху, показник плавності дозволяє аналізувати якість моторного контролю в реальному часі.

Даний показник може бути використаний для виявлення таких факторів, як тремор, м'язова втома або порушення координації рухів, оскільки їх наявність призводить до збільшення різких змін швидкості руху (jerk) та, відповідно, зниження значення smoothness.

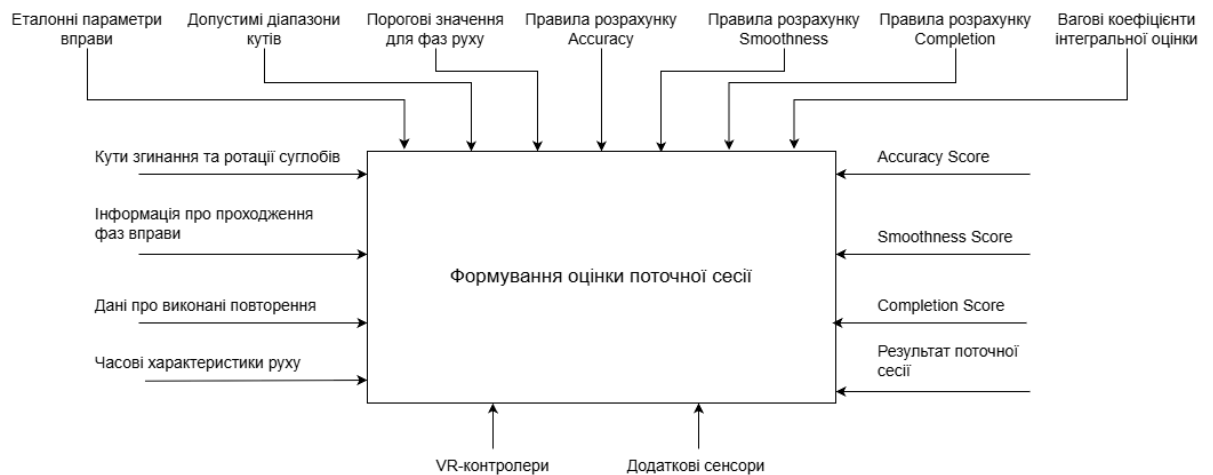


Рисунок 4.11 – IDEF0 діаграма рівня А-0

Більшість реальних реабілітаційних вправ не обмежуються одним типом руху. Наприклад, під час відведення ноги пацієнт може одночасно виконувати її ротацію або змінювати кут згинання в колінному суглобі. Тому підтримка комбінованих рухів дозволить наблизити вправи у системі до тих, що використовуються у клінічній практиці. На даний момент реалізовані вправи поділяються на дві основні категорії: вправи на ротацію та вправи на згинання суглобів. Проте в реабілітаційній практиці широко використовуються комплексні вправи, які поєднують обидва типи рухів. Для підтримки таких сценаріїв доцільно розширити механізм перевірки правильності виконання вправ шляхом додавання відповідної логіки до класу JointCondition. У такому випадку під час кожної фази виконання вправи буде здійснюватися перевірка необхідного суглоба відповідно до типу руху (ExerciseType), визначеного для поточної фази (PhaseType). Це дозволить коректно оцінювати вправи, що містять як ротаційні рухи, так і згинання суглобів у межах одного сценарію виконання, а також підвищить гнучкість і масштабованість

системи при додаванні нових типів вправ. Така модифікація архітектури дозволить додавати нові вправи без суттєвих змін у логіці оцінювання. Для створення нової вправи буде достатньо описати її фази та відповідні умови перевірки суглобів.

4.7 Тестування

Тестування програмної системи було проведено із використанням вбудованих засобів Unity, а також фреймворку xUnit, який застосовувався для тестування серверної частини на платформі .NET (через NuGet-пакети).

У Unity було реалізовано як runtime-тести, що виконуються під час роботи додатка, так і edit-time тести, які дозволяють перевіряти функціональність поза ігровим процесом. Окремо було створено mock-вправи, призначені для перевірки коректності роботи алгоритмів оцінювання рухів користувача. Дані вправи є тестовими сценаріями, які дозволяють відтворювати контрольовані рухи для перевірки різних умов роботи системи, зокрема правильного, частково правильного та некоректного виконання рухових патернів.

Дані mock-вправи прикріплювалися до 3D-моделі, яка імітує рухи пацієнта, що дозволило візуально контролювати процес виконання вправ у реальному часі. Це забезпечило можливість детальної перевірки коректності розрахунку кутів у суглобах, визначення фаз руху, а також загальної логіки алгоритму оцінювання.

Крім того, такий підхід дозволив виявити потенційні помилки у взаємодії між компонентами системи на ранніх етапах розробки та перевірити стабільність роботи алгоритмів при зміні вхідних параметрів.

Основна увага була зосереджена на тестуванні класу AdaptationEngine, який відповідає за розрахунок інтегрального показника виконання вправ та прийняття рішення щодо зміни складності тренування (збільшення, зменшення або стабілізація рівня).

У межах тестового модуля було реалізовано набір unit-тестів, кожен з яких перевіряє окремий сценарій роботи алгоритму. Тести побудовані за принципом

Arrange–Act–Assert, що забезпечує чітку структуру перевірки: підготовка вхідних даних, виконання методу та перевірка результату.

Зокрема, було протестовано наступні сценарії:

- 1) коректну роботу алгоритму при середніх значеннях параметрів, де очікується стабільний стан системи;
- 2) реакцію системи на підвищений рівень болю користувача, що має призводити до зменшення навантаження;
- 3) ситуацію високої ефективності виконання вправ, де система повинна ініціювати підвищення складності;
- 4) випадки низьких показників виконання, що також призводять до зниження рівня складності;
- 5) обмеження вихідного значення score у межах діапазону від 0 до 100;
- 6) контроль нижньої межі значення score для запобігання виходу за допустимі межі.

4.8 Впровадження

Система планується бути розгорнутою у локальному мережевому середовищі (LAN), що забезпечує стабільну взаємодію між її компонентами та мінімізує затримки під час обміну даними. Такий підхід є особливо важливим для VR-застосунків, оскільки дозволяє забезпечити своєчасну обробку інформації про рухи користувача та швидке отримання результатів аналізу.

Клієнтська частина системи реалізована у вигляді VR-додатка. Додаток запускається на персональному комп'ютері користувача, до якого підключається VR-шолом та додаткові засоби відстеження рухів. Саме клієнтська частина відповідає за відображення віртуального середовища, взаємодію користувача з об'єктами сцени, збір даних про виконання вправ та передачу цих даних на сервер.

Серверна частина реалізована у вигляді Web API та виконується на окремому сервері в межах локальної мережі. Для зберігання інформації планується, що база

даних PostgreSQL, може бути розгорнута як на тому самому сервері, що й серверна частина.

У якості цільової платформи обрано архітектуру PC VR, за якої основні обчислення виконуються на персональному комп'ютері, а VR-шолом використовується для відображення віртуального середовища та взаємодії з ним. Такий підхід було обрано через вищу обчислювальну потужність персональних комп'ютерів порівняно з автономними (standalone) VR-пристроями, що є важливим для забезпечення стабільної роботи системи, обробки даних від додаткових сенсорів, виконання алгоритмів оцінювання рухів та подальшої адаптації складності вправ. Крім того, використання PC VR спрощує інтеграцію із серверною частиною та додатковим обладнанням, яке може застосовуватися в реабілітаційному процесі.

Висновки до розділу 4

У межах даного розділу було описано процес програмної реалізації VR-додатка для реабілітації нижніх кінцівок. Представлено архітектурну структуру системи, що базується на трирівневій моделі, та логіку взаємодії між її компонентами. Описано реалізацію бази даних, серверної частини та ключових алгоритмів, які забезпечують аналіз і оцінку рухів користувача.

Окрему увагу приділено алгоритмам обробки даних руху, що дозволяють визначати правильність виконання вправ та формувати кількісну оцінку якості їх виконання. Також розглянуто створення UI-елементів та анімацій, які забезпечують інтерактивну взаємодію користувача з системою у VR-середовищі.

Проведене тестування підтвердило працездатність системи, коректність реалізованих функцій та стабільність роботи основних модулів.

5 РОБОТА КОРИСТУВАЧА ІЗ ПРОГРАМНОЮ СИСТЕМОЮ

Оскільки однією з ключових задач під час розробки було досягнення балансу між мінімалістичним користувацьким інтерфейсом та максимально інтерактивними VR-середовищами, особливу увагу приділено спрощенню взаємодії користувача із системою. Зокрема, інтерфейс спроектовано таким чином, щоб мінімізувати кількість зайвих дій, модальних вікон та додаткових підтверджень, які можуть відволікати користувача від основного процесу реабілітації.

Основний акцент зроблено на швидкому доступі до функціоналу та інтуїтивно зрозумілій навігації, тоді як VR-середовища забезпечують більш глибоке занурення у процес виконання вправ. Такий підхід дозволяє поєднати простоту взаємодії з високим рівнем залученості користувача, що є особливо важливим для ефективності реабілітаційного процесу.

5.1 Віртуальні середовища системи

При вході в додаток користувач потрапляє до стартового середовища, створеного засобами Unity, де може взаємодіяти з об'єктами за допомогою VR-контролерів. Сцена містить базові елементи користувацького інтерфейсу, зокрема: слайдери, випадаючі меню, чекбокси, прокручувані списки та модальні вікна. Крім того, у середовищі представлені об'єкти з різними типами взаємодії: відстеженням кінематичних рухів, миттєвими діями, рухами з урахуванням швидкості, а також взаємодією за допомогою погляду (наведення, вибір, керування та автоматичне скасування вибору), рисунок 5.1.

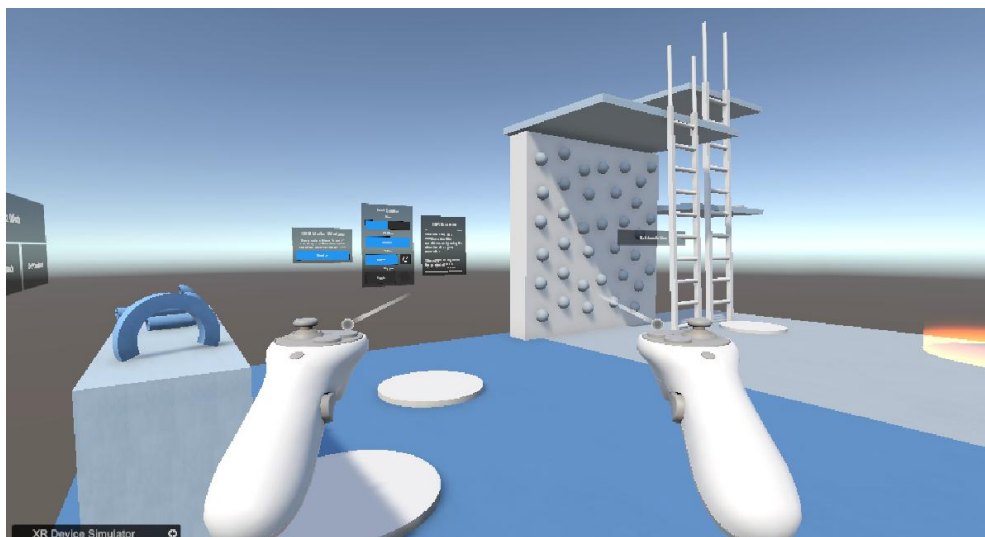


Рисунок 5.1 – Вигляд стартового середовища

Така організація взаємодії дозволяє користувачам, навіть без попереднього досвіду роботи з VR, швидко опанувати основні механіки та підготуватися до виконання реабілітаційних вправ.

Коли користувач готовий до початку вправ він може за допомогою меню обрати необхідне середовище (спортивний зал чи футбольне поле), після чого він буде автоматично телепортований на точку спауну даного рівня, де він може почати вправу. Для початку заняття було передбачено зону активації, яка візуально нагадує помаранчеве коло на підлозі. Зайшовши у цю зону користувач побачить стартове меню (Рисунок 5.2), де за допомогою іконки «Старт» він може почати завдання. Після цього буде автоматично активовано анімацію моделі асистента, який буде показувати необхідний рух.

Середовище «Спортзал» розроблене для користувачів, які віддають перевагу формальним заняттям, а не гейміфікованим рівням. Ця сценка є найбільш наближеною до процесу реальної реабілітації з лікарем. У сценці присутня моделька асистента, а середовище облаштоване як звичайний спортзал з фізичним обладнанням: гантелі, турнік та інші.



Рисунок 5.2 – Вигляд середовища «Спортзал»

Зверху екрана пацієнт може бачити власний прогрес виконання і кількість здійснених повторів, крім того при вдалому виконанні повтору користувач супроводжуватиме текст успішності на екрані. При неправильному виконанні повтор не буде зараховано і текст на екрані повідомить користувача про це.

У той же час віртуальне середовище «Футбольне поле» було розроблене для користувачів, які надають перевагу більш ігровому та динамічному формату реабілітації (рисунок 5.3). Дане середовище містить спеціально адаптований набір вправ для нижніх кінцівок, зокрема ударах ногою по м'ячу. Під час виконання вправ користувач має можливість взаємодіяти з віртуальним футбольним м'ячем та забивати його у ворота, що створює додатковий рівень залучення та мотивації до проходження реабілітації.

Використання елементів гейміфікації дозволяє зробити процес виконання вправ менш монотонним та підвищує зацікавленість користувача у регулярному проходженні тренувань.



Рисунок 5.3 – Вигляд середовища «Футбольне поле»

У майбутніх модифікаціях системи планується подальше розширення функціональності даного середовища, зокрема додавання нових рівнів та механік взаємодії. Серед запланованих функцій є реалізація передачі м'яча іншому гравцю, виконання вправ на точність удару, а також створення більш складних сценаріїв рухової активності, що дозволить урізноманітнити реабілітаційний процес та забезпечити поступове підвищення рівня складності вправ.

5.2 Користувацький інтерфейс системи

За потреби, користувач може поставити на паузу виконання вправи, нажавши на іконку «Пауза», яка знаходиться в правому верхньому куті стартового меню. Після чого з'явиться меню паузи (рисунок 5.4).

В меню паузи користувач може продовжити виконання вправи нажавши іконку «Старт», прогрес, який був досягнутий до цього буде збережено. Натиснувши на іконку «Перезапуск», користувач може почати вправу заново, скинувши повністю минулий прогрес. Натиснувши на іконку «Меню» користувача перенаправить на вікно «Рівні» (рисунок 5.5) з вибором необхідної вправи, якщо користувач хоче змінити поточне середовище, чи завдання.

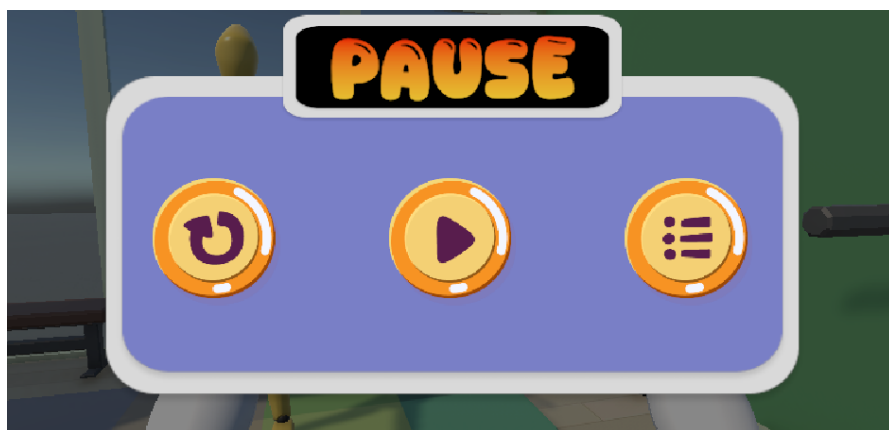


Рисунок 5.4 – Вікно паузи

Вікно «Рівні» має також має іконку «Назад», яка знаходиться у верхньому лівому кутку, яка перенаправить користувача назад до меню «Пауза», у разі випадкового натиску.



Рисунок 5.5 – Вікно сцен

Після успішного проходження вправи користувач повинен пройти коротке опитування про його емоційний стан (рисунок 5.6). Оцінка болю і втоми оцінюється за 10-ти бальною шкалою, за допомогою слайдера, який користувач може перетягувати VR контролерами. Щоб оцінити настрої користувачу лише необхідно натиснути на емоджі, який на його думку є найбільш релевантним.

Після натискання кнопки «Submit» всі результати записуються в базу даних і користувач може або виконувати дану вправу далі з адаптованою складністю, або

переключити на іншу за допомогою меню.



Рисунок 5.6 – Вікно опитування

Всі спливаючі вікна зроблені в одному візуальному стилі для кращої навігації і візуального сприйняття картинки. UI елементи мають мінімалістичний стиль, через те, що вся навігація відбувається за допомогою VR контролерів, тому всі меню, кнопки і іконки повинні бути доволі великого розміру і не перенавантаженими.

Висновки до розділу 5

У даному розділі було розглянуто особливості взаємодії користувача з розробленою програмною системою. Описано основні етапи роботи з VR-додатком, включаючи вибір вправ, взаємодію з віртуальним середовищем, отримання візуального зворотного зв'язку та проходження реабілітаційних сесій. Також було продемонстровано різні ігрові середовища та механіки, що використовуються для підвищення мотивації та залученості користувача до процесу реабілітації.

ВИСНОВКИ

У процесі виконання завдань, поставлених для реалізації програмного модуля адаптації складності реабілітаційних вправ на основі прогресу пацієнта, було розроблено VR-додаток із алгоритмом оцінки правильності виконання вправ. Запропонований алгоритм поєднує декілька параметрів виконання вправ, що забезпечує масштабованість та гнучкість системи для подальшого розширення.

Під час розробки було реалізовано всі основні функціональні вимоги. За допомогою Unity створено інтерактивні VR-середовища. Також із використанням PostgreSQL реалізовано базу даних для зберігання інформації про процес реабілітації пацієнтів, їхній прогрес та показники точності виконання вправ.

Окрім цього, було проведено ознайомлення з вбудованими інструментами Unity, що забезпечують створення та керування анімаціями. На основі цих інструментів було реалізовано анімацію модельки віртуального асистента, який демонструє еталонне виконання реабілітаційних вправ. Це дозволяє користувачу наочно орієнтуватися у правильній техніці виконання рухів та підвищує ефективність навчального процесу.

Розроблений програмний продукт є не лише прикладом сучасного VR-рішення у сфері реабілітації, але й системою, що базується на медичних рекомендаціях та має потенціал до подальшого вдосконалення. Його метою є підтримка процесу відновлення пацієнтів після різних типів травм, включаючи побутові ушкодження, травми внаслідок нещасних випадків та бойових дій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Baron L., Chheang V., Chaudhari A., Liaqat A., Chandrasekaran A., Wang Y. Virtual Therapy Exergame for Upper Extremity Rehabilitation Using Smart Wearable Sensors. Proceedings of the 8th ACM/IEEE International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE '23). Orlando, FL, USA, 2023. URL: <https://doi.org/10.1145/3580252.3586975>
2. Zhang F., Xu L., Zhou Y., Zhang N., Liu Z. Research of Lower Limb Rehabilitation System with Multisensory Feedback Based on Virtual-Real Coordination. In: Ying F. et al. (Eds.) International Symposium on World Ecological Design. IOS Press, 2024. URL: <https://doi.org/10.3233/FAIA240099>
3. Longo U.G., Carnevale A., Andreoli F., Mannocchi I., Bravi M., Hadj Sassi M.S., Santacaterina F., Carli M., Schena E., Papalia R. Immersive Virtual Reality for Shoulder Rehabilitation: Evaluation of a Physical Therapy Program Executed with Oculus Quest 2. BMC Musculoskeletal Disorders. 2023. Vol. 24. Article 859. URL: <https://doi.org/10.1186/s12891-023-06861-5>
4. Fok Kow F.-C.-S., Sinha A.K., Ming Z.J., Songyu B., Tan J.J.K., Jeffrey H.Y.J., Mihaleva G., Magnenat Thalmann N., Cai Y. MIDAS: Multi-sensorial Immersive Dynamic Autonomous System Improves Motivation of Stroke-Affected Patients for Hand Rehabilitation. 2021. [URL:10.36227/techrxiv.17006641](https://doi.org/10.36227/techrxiv.17006641)
5. Csikszentmihalyi M. Flow: The Psychology of Optimal Experience. New York: Harper & Row, 1990. 303 p.
6. Fisher N., Kulshreshth A.K. Exploring Dynamic Difficulty Adjustment Methods for Video Games. Virtual Worlds. 2024. Vol. 3, No. 2. P. 230–255. URL: <https://doi.org/10.3390/virtualworlds3020012>
7. Unity: Develop, Deploy, and Grow | The World's Leading Game Engine. Unity. URL: <https://unity.com/> .
8. Unity Documentation. Unity Documentation. URL: <https://docs.unity.com/en-us> .

9. PostgreSQL: About. PostgreSQL: The world's most advanced open source database. URL: <https://www.postgresql.org/about/> .
10. C# - a modern, open-source programming language | .NET. Microsoft. URL: <https://dotnet.microsoft.com/en-us/languages/csharp> .
11. Scripting in Unity for experienced C# & C++ programmers. Unity. URL: <https://unity.com/how-to/programming-unity> .
12. Understanding the architecture of a 3-tier application. vFunction. URL: <https://vfunction.com/blog/3-tier-application/> .
13. What is Object-Relational Mapping (ORM)? - Object-Relational Mapping Explained - AWS. Amazon Web Services, Inc. URL: <https://aws.amazon.com/what-is/object-relational-mapping/> .

ДОДАТОК А Лістинг розробленої системи

ExerciseType.cs

```
using UnityEngine;

public enum ExerciseType
{
    JointAngle,
    Rotation
}
```

JointCondition.cs

```
using UnityEngine;

[System.Serializable]
public class JointCondition
{
    public JointType jointType;
    public PhaseType phase;

    public float minAngle;
    public float maxAngle;

    public float tolerance;
}
```

JointType.cs

```
using UnityEngine;

public enum JointType
{
    Hip,
    Knee,
    Ankle,
    Toes
}
```

PhaseType.cs

```
using UnityEngine;

public enum PhaseType
{
    Start,
    Moving,
    Peak,
    Return
}
```

ExerciseState.cs

```
public enum ExerciseState
{
    Idle,
    Training,
    Finished
}
```

Exercise.cs

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

[CreateAssetMenu(fileName = "NewExercise", menuName = "Rehab/Exercise")]
public class Exercise : ScriptableObject
{
    public string exerciseName;
    public ExerciseType type;

    public int targetReps;
    public float allowedDeviation;

    public List<JointCondition> requiredJoints;
}
```

ZoneTrigger.cs

```
using UnityEngine;

public class ZoneTrigger : MonoBehaviour
{
    public FootExerciseSystemManager exerciseSystem;
    public GameObject zoneObject;
    public Exercise currentExercise;

    public Animator animator;

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            exerciseSystem.currentZone = zoneObject;
            exerciseSystem.OnPlayerEnterZone();
            exerciseSystem.animator = animator;
            exerciseSystem.currentExercise = currentExercise;
        }
    }

    private void OnTriggerExit(Collider other)
```

```

    {
        if (other.CompareTag("Player"))
        {
            exerciseSystem.OnPlayerExitZone();
        }
    }
}

```

UIManager.cs

```

using UnityEngine;
using TMPPro;
using UnityEngine.UI;
using System.Collections;
public class UIManager : MonoBehaviour
{
    public static UIManager Instance;

    public GameObject mainCanvas;
    public GameObject pauseCanvas;
    public GameObject sceneCanvas;
    public GameObject pollCanvas;
    public GameObject startButton;
    public Image repsCalculatorProgress;
    public Slider progressBar;
    public TMP_Text feedbackText;
    public TMP_Text repsCalculatorText;
    public Slider painSlider;
    public Slider fatigueSlider;
    public int selectedInterestLevel;
    public CanvasGroup tutorialText;
    public bool isSceneActivated;
    public GameObject returnButton;

    void Awake()
    {
        Instance = this;
    }

    void Start()
    {
        ShowTutorialMessage(
            "Welcome! Try moving around.\nWhen you are ready, proceed to the start area."
        );
    }

    public void ShowTutorialMessage(string message)
    {

```

```

StopAllCoroutines();

tutorialText.GetComponentInChildren<TMPPro.TMP_Text>().text = message;

StartCoroutine(ShowAndFade());
}

IEnumerator ShowAndFade()
{
    tutorialText.gameObject.SetActive(true);
    tutorialText.alpha = 1;

    yield return new WaitForSeconds(4f);

    yield return StartCoroutine(FadeOut(tutorialText));
}

IEnumerator FadeOut(CanvasGroup canvasGroup)
{
    float duration = 1.5f;
    float time = 0;

    while (time < duration)
    {
        canvasGroup.alpha = 1 - (time / duration);
        time += Time.deltaTime;
        yield return null;
    }

    canvasGroup.alpha = 0;
    canvasGroup.gameObject.SetActive(false);
}

public void ShowStartButton()
{
    if (isSceneActivated)
    {
        mainCanvas.SetActive(true);
        startButton.SetActive(true);
    }
    else
    {
        sceneCanvas.SetActive(true);
        isSceneActivated = true;
    }
}

public void HideAll()

```

```

{
    mainCanvas.SetActive(false);
    sceneCanvas.SetActive(false);
    pauseCanvas.SetActive(false);
    pollCanvas.SetActive(false);
}

public void SetFeedback(string text)
{
    feedbackText.text = text;
}

public void SetProgress(float value, float currentReps)
{
    progressBar.value = value;
    repsCalculatorProgress.fillAmount = value;
    repsCalculatorText.text = currentReps.ToString("0");
}

public void OpenSceneMenu()
{
    pauseCanvas.SetActive(false);
    sceneCanvas.SetActive(true);
}

public void BackToPause()
{
    sceneCanvas.SetActive(false);
    pauseCanvas.SetActive(true);
}

public void Submit()
{
    FootExerciseSystemManager manager =
        FindObjectOfType<FootExerciseSystemManager>();

    manager.painLevel = Mathf.RoundToInt(painSlider.value);
    manager.fatigueLevel = Mathf.RoundToInt(fatigueSlider.value);
    manager.interestLevel = selectedInterestLevel;

    pollCanvas.SetActive(false);
    manager.StartCoroutine(manager.GetAdaptation(manager.currentScore));
}

public void SelectInterestLevel(int value)
{
    selectedInterestLevel = value;
}

```

```
}
```

FootExerciseSystemManager.cs

```
void EvaluateJointAngle()
{
    totalFrames++;

    bool startPhaseCorrect = true;
    bool peakPhaseCorrect = true;

    foreach (var joint in currentExercise.requiredJoints)
    {
        float angle = GetJointAngle(joint.jointType);
        jointAngleHistory[joint.jointType].Add(angle);

        bool insideRange =
            angle >= (joint.minAngle - joint.tolerance) &&
            angle <= (joint.maxAngle + joint.tolerance);

        switch (joint.phase)
        {
            case PhaseType.Start:
                if (!insideRange)
                    startPhaseCorrect = false;
                break;

            case PhaseType.Peak:
                if (!insideRange)
                    peakPhaseCorrect = false;
                break;
        }
    }

    bool isInStart = startPhaseCorrect;
    bool isInPeak = peakPhaseCorrect;

    if (isInPeak && !reachedPeak)
    {
        reachedPeak = true;

        float repAccuracy = CalculateAccuracy();
        totalAccuracy += repAccuracy;
    }

    if (isInStart)
    {
        startStableTime += Time.deltaTime;
    }
}
```

```

    }
    else
    {
        startStableTime = 0f;
    }

    if (reachedPeak && startStableTime >= requiredStableTime)
    {
        currentReps++;

        progress = Mathf.Clamp01(
            (float)currentReps / currentExercise.targetReps
        );

        float smoothness = CalculateSmoothness();
        totalSmoothness += smoothness;

        UIManager.Instance?.SetProgress(progress, currentReps);
        UIManager.Instance?.SetFeedback("Good repetition!");

        reachedPeak = false;
        reachedStartPosition = false;
        startStableTime = 0f;

        foreach (var key in jointAngleHistory.Keys)
        {
            jointAngleHistory[key].Clear();
        }
    }

    if (isInStart)
    {
        reachedStartPosition = true;
    }

    if (currentReps >= currentExercise.targetReps)
    {
        FinishExercise();
    }
}

void EvaluateRotation()
{
    if (state != ExerciseState.Training)
        return;

    totalFrames++;

```

```

bool startPhaseCorrect = true;
bool peakPhaseCorrect = true;

foreach (var joint in currentExercise.requiredJoints)
{
    float angle = GetRotationAngle(joint.jointType);

    Debug.Log($" {joint.jointType} | Phase: {joint.phase} | Angle: {angle} | Min: {joint.minAngle} | Max:
{joint.maxAngle}");

    bool insideRange =
        angle >= (joint.minAngle - joint.tolerance) &&
        angle <= (joint.maxAngle + joint.tolerance);

    switch (joint.phase)
    {
        case PhaseType.Start:
            if (!insideRange)
                startPhaseCorrect = false;
            break;

        case PhaseType.Peak:
            if (!insideRange)
                peakPhaseCorrect = false;
            break;
    }
}

bool isInStart = startPhaseCorrect;
bool isInPeak = peakPhaseCorrect;

// Досягнення Peak
if (isInPeak && !reachedPeak && reachedStartPosition)
{
    reachedPeak = true;

    float repAccuracy = CalculateAccuracy();
    Debug.Log($"repAccuracy = {repAccuracy}");
    totalAccuracy += repAccuracy;

    UIManager.Instance?.SetFeedback("Peak reached");
}

// Накопичення стабільності Start
if (isInStart)
    startStableTime += Time.deltaTime;
else
    startStableTime = 0f;

```

```

// REP (Peak - стабільний Start)
if (reachedPeak && startStableTime >= requiredStableTime)
{
    currentReps++;

    float progress = Mathf.Clamp01(
        (float)currentReps / currentExercise.targetReps
    );

    float smoothness = CalculateSmoothness();
    totalSmoothness += smoothness;

    UIManager.Instance?.SetProgress(progress, currentReps);
    UIManager.Instance?.SetFeedback("Good repetition!");

    // reset
    reachedPeak = false;
    reachedStartPosition = false;
    startStableTime = 0f;

    foreach (var key in jointAngleHistory.Keys)
    {
        jointAngleHistory[key].Clear();
    }
}

// Фіксуємо що ми були у Start
if (isInStart)
{
    reachedStartPosition = true;
}

// завершення
if (currentReps >= currentExercise.targetReps)
{
    FinishExercise();
}
}

private float GetAngle(JointType jointType)
{
    switch (currentExercise.type)
    {
        case ExerciseType.Rotation:
            return GetRotationAngle(jointType);

        case ExerciseType.JointAngle:

```

```

        return GetJointAngle(jointType);

    default:
        return 0f;
    }
}

private float GetJointAngle(JointType jointType)
{
    Transform targetJoint = null;

    switch (jointType)
    {
        case JointType.Hip:
            targetJoint = hip;
            break;

        case JointType.Knee:
            targetJoint = knee;
            break;

        case JointType.Ankle:
            targetJoint = ankle;
            break;

        default:
            return 0f;
    }

    if (targetJoint == null)
        return 0f;

    Transform parent = targetJoint.parent;
    Transform child = targetJoint.childCount > 0 ? targetJoint.GetChild(0) : null;

    if (parent == null || child == null)
        return 0f;

    Vector3 dir1 = parent.position - targetJoint.position;
    Vector3 dir2 = child.position - targetJoint.position;

    float angle = Vector3.Angle(dir1, dir2);

    return angle;
}

private float GetRotationAngle(JointType jointType)
{

```

```

Transform targetJoint = GetJointTransform(jointType);

if (targetJoint == null)
    return 0f;

// нормалізація [-180, 180]
float angle = Mathf.DeltaAngle(0f, targetJoint.localEulerAngles.y);

return angle;
}

public float CalculateAccuracy()
{
    float total = 0f;
    int count = 0;

    foreach (var joint in currentExercise.requiredJoints)
    {
        if (joint.phase != PhaseType.Peak) continue;

        float angle = GetJointAngle(joint.jointType);
        float target = (joint.minAngle + joint.maxAngle) / 2f;

        float error = Mathf.Abs(angle - target);
        float maxError = joint.tolerance;

        float accuracy = Mathf.Clamp01(1f - error / maxError);
        total += accuracy;
        count++;
    }

    return count > 0 ? (total / count) * 100f : 0f;
}

float CalculateSmoothness(List<float> angles)
{
    if (angles.Count < 3)
        return 1f;

    float jerkSum = 0f;
    int samples = 0;

    for (int i = 2; i < angles.Count; i++)
    {
        float v1 = angles[i] - angles[i - 1];
        float v2 = angles[i - 1] - angles[i - 2];

        float jerk = Mathf.Abs(v2 - v1);
    }
}

```

```

        jerkSum += jerk;
        samples++;
    }

    float avgJerk = jerkSum / samples;

    float smoothness = 1f / (1f + avgJerk);

    return Mathf.Clamp01(smoothness);
}

float CalculateSmoothness()
{
    float total = 0f;
    int count = 0;

    foreach (var kvp in jointAngleHistory)
    {
        float smoothness = CalculateSmoothness(kvp.Value);
        // Debug.Log($"Joint {kvp.Key} smoothness = {smoothness}");
        total += smoothness;
        count++;
    }

    float avg = count > 0 ? total / count : 0f;

    // Debug.Log($"AVG smoothness = {avg}");

    return avg;
}

```

EnvironmentManager.cs

```

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

public class EnvironmentManager : MonoBehaviour
{
    public FootExerciseSystemManager exerciseSystem;

    public GameObject[] environments;
    public Transform[] spawnPoints;

    public Transform player;

    private int currentIndex = -1;

```

```

public VRBodyFollow bodyFollow;
public LegsFollow legsFollow;

public Transform xrOrigin; // XR Origin
public Transform head; // Main Camera

public void ShowEnvironment(int index)
{
    Debug.Log($"ShowEnvironment called: {index}");

    for (int i = 0; i < environments.Length; i++)
        environments[i].SetActive(false);

    environments[index].SetActive(true);

    Vector3 offset = xrOrigin.position - head.position;
    offset.y = 0; // щоб не ламати висоту

    xrOrigin.position = spawnPoints[index].position + offset;
    exerciseSystem.ResetExercise();
    UIManager.Instance.HideAll();

    exerciseSystem.currentZone.SetActive(true);
    Debug.Log($"Activated: {environments[index].name}");

    UIManager.Instance.ShowTutorialMessage(
        "When you are ready, proceed to the start area.\n\nThe assistant will show you the correct movement"
    );

    // if (exerciseSystem.currentZone != null)
    //     exerciseSystem.currentZone.SetActive(true);

}

public void HideAll()
{
    foreach (var env in environments)
    {
        env.SetActive(false);
    }

    currentIndex = -1;
}
}

```



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

Програмний модуль адаптації складності реабілітаційних вправ на основі прогресу пацієнта

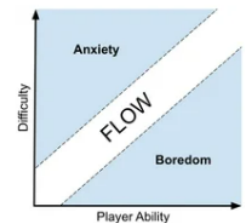
виконала: студентка групи ТВ-23 Корж.І.В.

керівник: посада Дацюк Ольга Анатолівна, старший викладач

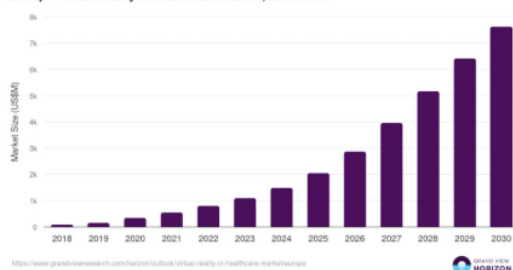
2026

Актуальність теми

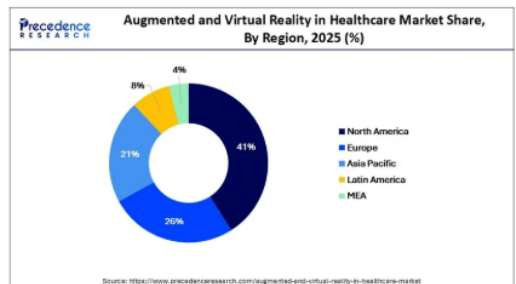
1. Зростання використання VR у медицині та реабілітації
2. Підвищення мотивації пацієнтів через інтерактивність
3. Можливість дистанційної реабілітації
4. Адаптивне налаштування складності вправ



Europe virtual reality in healthcare market, 2018-2030



<https://www.grandviewresearch.com/industry-analysis/virtual-reality-in-healthcare-market>



Source: <https://www.precedenceresearch.com/augmented-and-virtual-reality-in-healthcare-market>



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Постановка задачі

Мета роботи:

Розробка програмного модуля, який забезпечує відстеження рухів пацієнта за допомогою VR-контролерів та додаткових сенсорів, а також автоматичну адаптацію складності реабілітаційних вправ відповідно до індивідуального прогресу користувача.

Задачі:

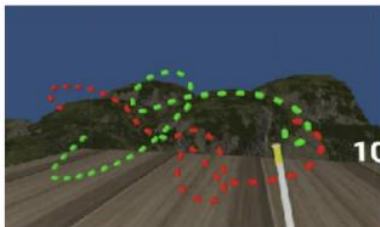
1. Аналіз предметної області
2. Аналіз наявних систем
3. Визначення вимог
4. Проектування архітектури
5. Реалізація системи
6. Тестування та налагодження



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

3/17

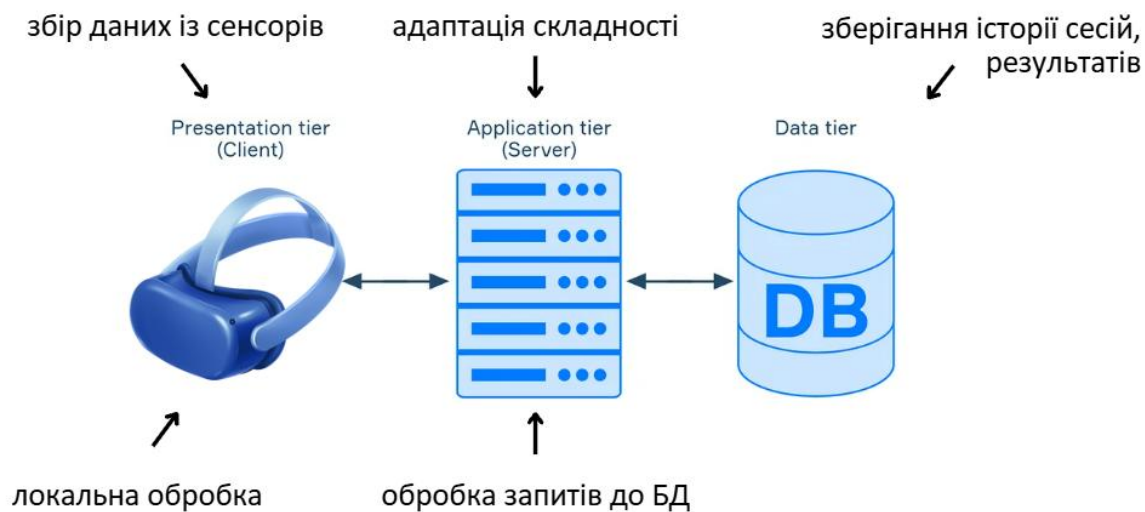
Аналіз предметної області



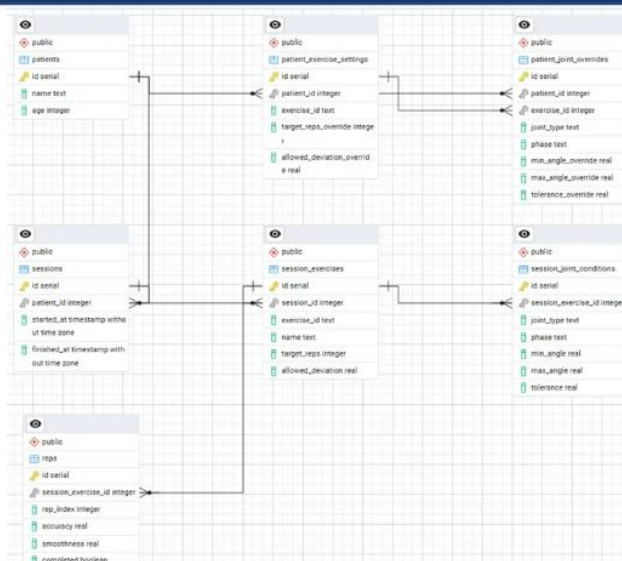
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

4/17

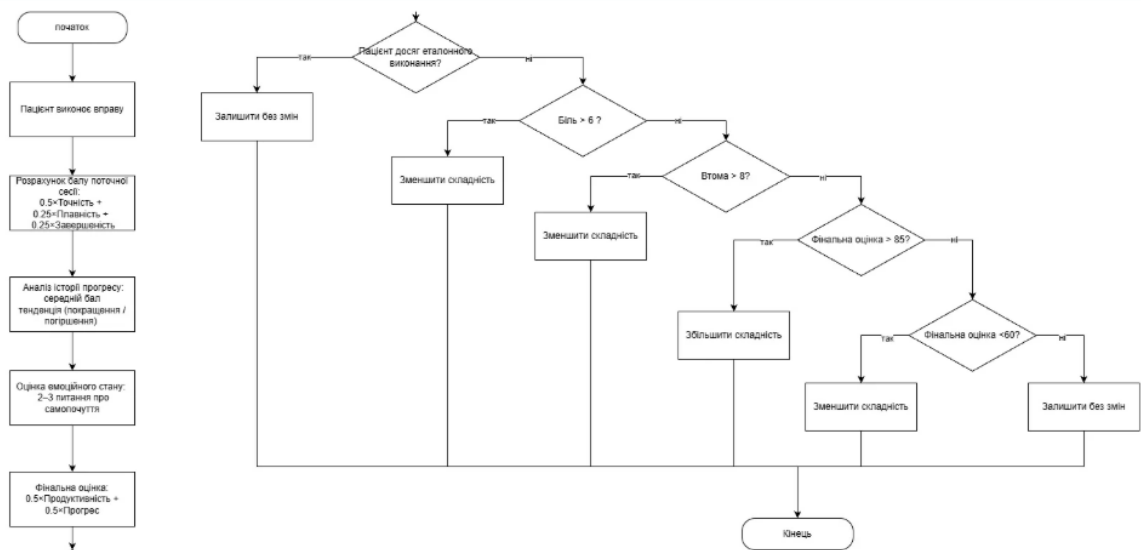
Архітектура системи



Модель бази даних



Блок схема алгоритму адаптації вправ



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

7/17

Діаграма IDEF0



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

8/17

Опис алгоритму

CalculateAccuracy
↓
Для кожного суглоба
↓
 $\text{target} = (\text{min} + \text{max}) / 2$
↓
 $\text{error} = |\text{angle} - \text{target}|$
↓
Clamp [0,1]
↓
Середнє значення
↓
 $\times 100\%$
↓
RETURN

CalculateSmoothness
↓
Для кожного суглоба
↓
Для кожної точки (i):
↓
 $v1 = \text{angle}[i] - \text{angle}[i-1]$
 $v2 = \text{angle}[i-1] - \text{angle}[i-2]$
↓
 $\text{jerk} = |v2 - v1|$
↓
 $\text{avgJerk} = \text{середнє}$
↓
 $\text{smoothness} = 1 / (1 + \text{avgJerk})$
↓
Середнє по всіх суглобах
↓
RETURN

v - velocity
(швидкість)
jerk - зміна швидкості
між кадрами



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

9/17

Засоби розробки



Unity



PostgreSQL



Blender



VS Code



Entity Framework Core



Figma



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

10/17

Віртуальні середовища системи

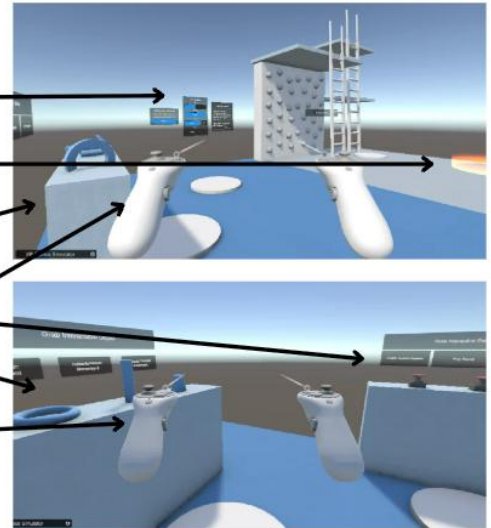
Стартове середовище

базові елементи користувацького інтерфейсу

зона активації

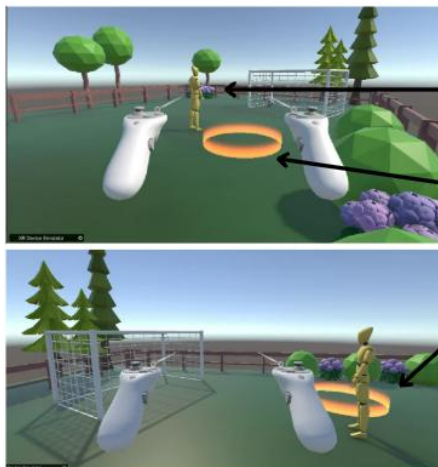
об'єкти з різними
типами взаємодії

VR-контролери



Віртуальні середовища системи

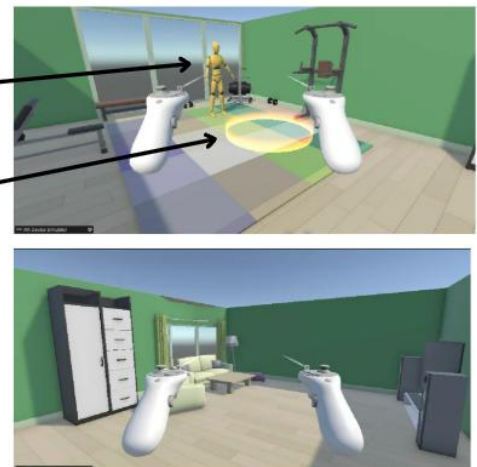
Середовище «Футбольне поле»



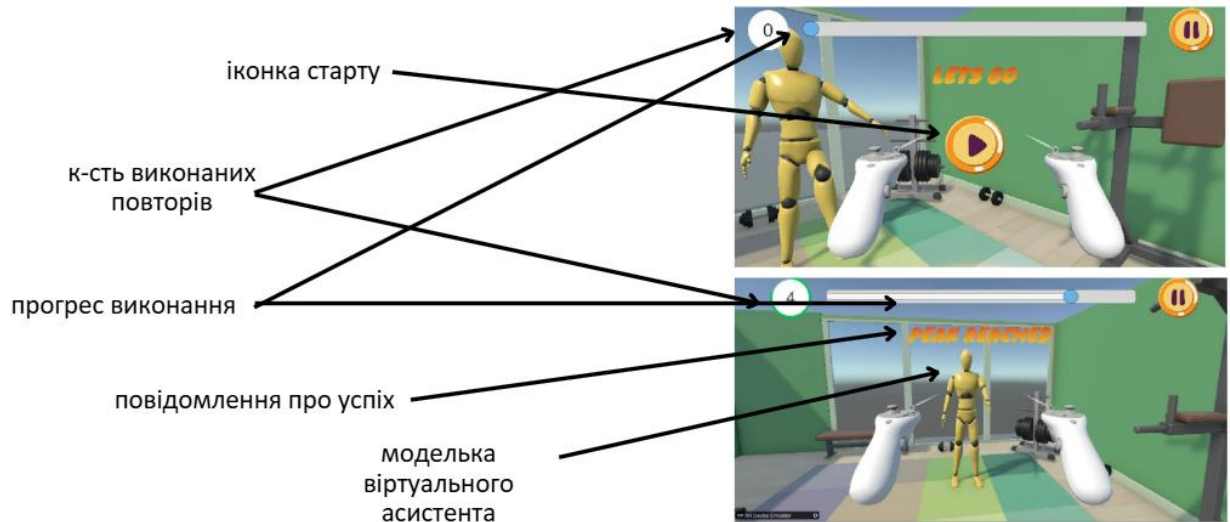
моделька
віртуального
асистента

зона активації

Середовище «Спортзал»



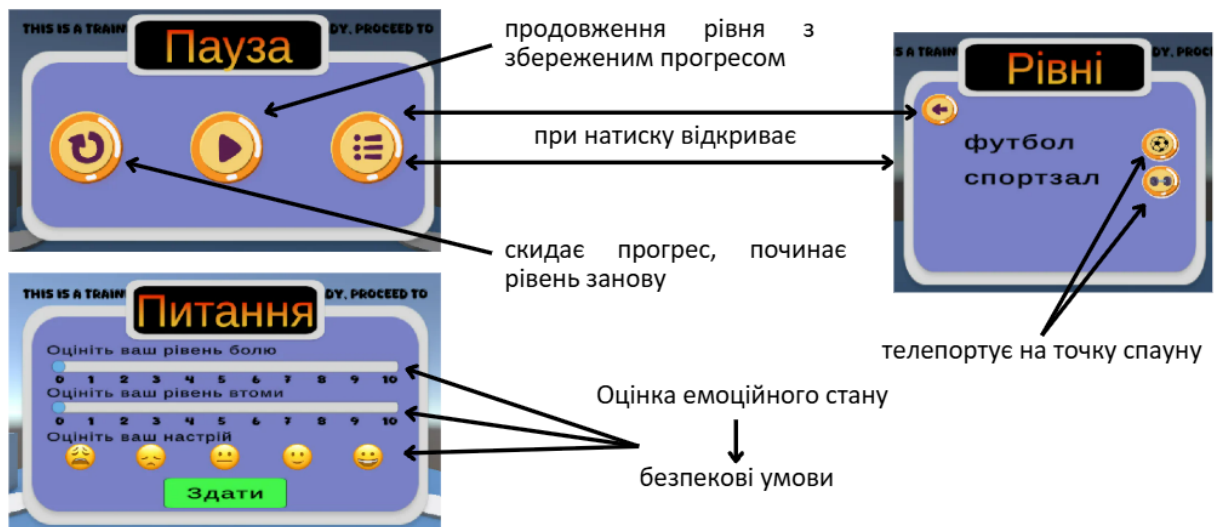
Віртуальні середовища системи



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

13/17

Користувацький інтерфейс системи



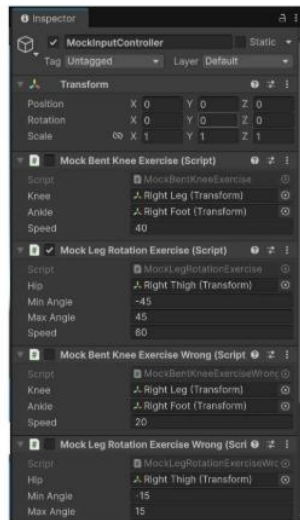
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

14/17

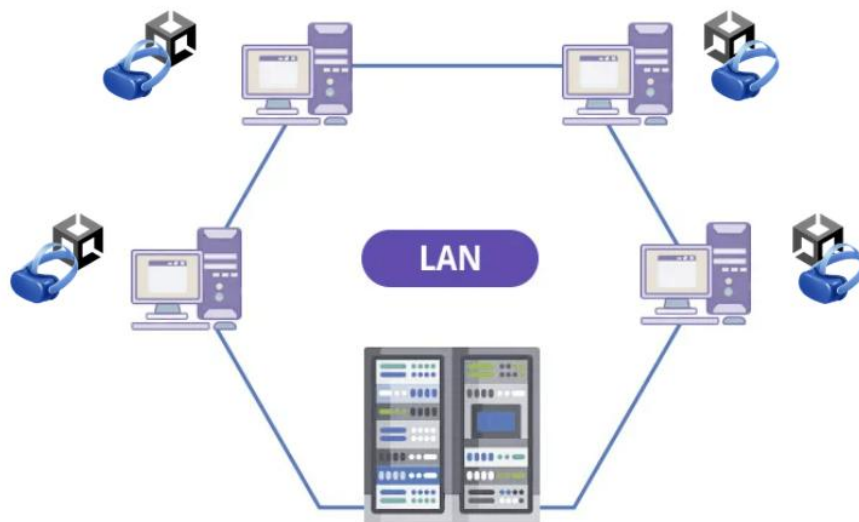
Тестування системи

1. Мок-вправи прикріплювалися до 3D-моделі, яка імітує рухи пацієнта

2. Unit-тест для беку



Впровадження та супровід



Висновки

Розроблено VR-додаток з автоматичною адаптацією складності реабілітаційних вправ відповідно до індивідуального прогресу користувача

Відповідно до завдання виконано наступне:

1. Проведено аналіз предметної області
2. Проведено аналіз наявних систем
3. Визначено вимоги системи
4. Зпроектовано архітектуру
5. Реалізовано систему
6. Протестовано та налагоджено систему

